First edition (March 2001)

This edition applies to Version 2 Release 1 of CICS Transaction Server for z/OS, program number 5697-E93, and to all subsequent versions, releases, and modifications until otherwise indicated in new editions. Make sure you are using the correct edition for the level of the product.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address given below.

At the back of this publication is a page entitled “Sending your comments to IBM”. If you want to make comments, but the methods described are not available to you, please address them to:

IBM United Kingdom Laboratories, Information Development,
Mail Point 095, Hursley Park, Winchester, Hampshire, England, SO21 2JN.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© Copyright International Business Machines Corporation 1989, 2001. All rights reserved.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
Contents

Preface ............................................ xxv
What this book is about ....................... xxv
Who this book is for ........................... xxv
What you need to know to use this book ..... xxv
Notes on terminology .......................... xxv
Book structure .................................. xxvi

Summary of changes ............................ xxvii
Changes for this CICS Transaction Server for
z/OS, Version 2 Release 1 edition ............ xxvii
Changes for the CICS Transaction Server for
OS/390 Version 1 Release 3 edition .......... xxvii
Changes for the CICS/ESA 4.1 edition ...... xxvii
Changes for the CICS/ESA 3.3 edition .... xxviii

Part 1. Introduction ............................ 1

Chapter 1. CICS domains ..................... 3
Domain gates ..................................... 6
Functions provided by gates .................. 6
Specific gates, generic and call-back gates. 6
Domain call formats ............................. 7
Ownership of formats .......................... 7
Tokens ........................................... 8
Responses ....................................... 8

Chapter 2. Application domain ............. 9

Part 2. CICS components ........................ 11

Chapter 3. Added gates domain (AP) ....... 13
AP domain’s new gates ......................... 13
  APTC gate, OPEN function .................. 13
  APTC gate, SEND function .................. 14
  APTC gate, RECEIVE function .............. 14
  APTC gate, CLOSE function ................. 15
  APTC gate, ISSUE_NOTIFY function ....... 15
  APTC gate, LISTEN function ............... 16
  APTC gate, CANCEL function ............... 16
  APTC gate, SET_SESSION function ....... 16
  APTC gate, EXTRACT_PROCESS function ... 17
  APID gate, QUERY_NETNAME function ..... 17
  APID gate, PROFILE function .............. 18
Modules .......................................... 18
Exits ............................................ 19
Trace ............................................ 19

Chapter 4. Application domain (AP) ...... 21
Application domain’s specific gates ......... 21
  ABAB gate, CREATE_ABEND_RECORD function .... 21
  ABAB gate, UPDATE_ABEND_RECORD function .... 23
  ABAB gate, START_ABEND function .......... 27
  ABAB gate, INQUIRE_ABEND_RECORD function .. 28
  ABAB gate, TAKE_TRANSACTION_DUMP function .. 29
  APAP gate, TRANSFER_SIT function .......... 30
  APJX gate, INVOKE_USER_EXIT function ...... 30
  APID gate, PROFILE function .............. 31
  APID gate, QUERY_NETNAME function ....... 31
  APIQ gate, INQ_APPLICATION_DATA function .. 32
  APJC gate, WRITE_JOURNAL_DATA function ... 33
  APLH gate, ESTABLISH_LANGUAGE function ... 33
  APLH gate, START_PROGRAM function ....... 34
  APLH gate, NOTIFY_REFRESH function ....... 35
  APLI gate, ESTABLISH_LANGUAGE function .... 36
  APLI gate, START_PROGRAM function ....... 37
  APLJ gate, ESTABLISH_LANGUAGE function .... 38
  APLJ gate, START_PROGRAM function ....... 39
  APMR gate, TRANSACTION_INITIALIZATION function ... 40
  APMR gate, TRANSACTION_TERMINATION function ..... 41
  APMR gate, INQUIRE function ............... 41
  APRT gate, ROUTE_TRANSACTION function ...... 42
  APTC gate, CANCEL function ............... 43
  APTC gate, CLOSE function ................. 43
  APTC gate, CLOSE function ................. 44
  APTC gate, EXTRACT_PROCESS function ....... 44
  APTC gate, ISSUE_NOTIFY function ......... 45
  APTC gate, LISTEN function ............... 45
  APTC gate, OPEN function .................. 46
  APTC gate, RECEIVE function .............. 46
  APTC gate, SEND function .................. 47
  APTC gate, SET_SESSION function .......... 47
  APTD gate, WRITE_TRANSIENT_DATA function .. 48
  APTD gate, READ_TRANSIENT_DATA function ... 49
  APTD gate, DELETE_TRANSIENT_DATA function .... 49
  APTD gate, RESET_TRIGGER_LEVEL function ... 50
  APTD gate, INITIALISE_TRANSIENT_DATA function .... 51
  APXM gate, TRANSACTION_INITIALIZATION function .... 51
  APXM gate, RMI_START_OF_TASK function .... 51
  APXM gate, TRANSACTION_TERMINATION function .... 52
  BRAT gate, ATTACH function ............... 52
  BRIQ gate, INQUIRE_CONTEXT function ...... 53
  ICXM gate, BIND_FACILITY function ....... 54
  ICXM gate, RELEASE_FACILITY function .... 54
  ICXM gate, INQUIRE_FACILITY function .... 54
  TFXM gate, INIT_XM_CLIENT function ....... 55
  TFXM gate, BIND_XM_CLIENT function ....... 55
  MRXM gate, INIT_XM_CLIENT function ....... 55
  MRXM gate, BIND_XM_CLIENT function ....... 56
  62XM gate, INIT_XM_CLIENT function ....... 56
  62XM gate, BIND_XM_CLIENT function ....... 57

© Copyright IBM Corp. 1989, 2001
Chapter 5. AP domain initialization program

Chapter 6. AP domain KC subcomponent

Chapter 7. AP domain termination program.

Chapter 8. Autoinstall for terminals, consoles and APPC connections
Chapter 9. Autoinstall terminal model manager .................. 127
Functions provided by the autoinstall terminal model manager. .......... 127
   AIIN format, START_INIT function .............................. 127
   AIIN format, COMPLETE_INIT function ......................... 127
   AIIQ format, LOCATE_TERM_MODEL function .................... 128
   AIIQ format, UNLOCK_TERM_MODEL function ...................... 128
   AIIQ format, INQUIRE_TERM_MODEL function ..................... 129
   AIIQ format, START_BROWSE function ........................... 129
   AIIQ format, GET_NEXT function ................................ 129
   AIIQ format, END_BROWSE function ............................... 130
   AITM format, ADD_REPL_TERM_MODEL function .................... 130
   AITM format, DELETE_TERM_MODEL function ...................... 131

Modules ........................................... 131
Exits .............................................. 132
Trace .............................................. 132

Chapter 10. Basic mapping support .......................... 133
Design overview .................................................. 133
Message routing ................................................... 135
Terminal paging .................................................... 135
Device independence ............................................. 135
Control blocks .................................................... 136
Modules ....................................................... 139
   DFHDSB (data stream build) .................................. 141
   DFHIIP (non-3270 input mapping) .............................. 143
   DFHMCP (mapping control program) ............................ 143
   DFHML1 (LU1 printer with extended attributes mapping) .... 147
   DFHM32 (3270 mapping) ....................................... 149
   DFHPBP (page and text build) ................................ 151
   DFHPHP (partition handling program) ......................... 153
   DFHRLR (route list resolution program) ....................... 156
   DFTPPP (terminal page processor) ............................. 158
   DFHTPQ (undelivered messages cleanup program) ............. 160
   DFHTPR (terminal page retrieval program) .................... 162
   DFHTPS (terminal page scheduling program) ................... 165
Copy books .............................................. 165
Exits .............................................. 165
Trace .............................................. 165

Chapter 11. Builders ........................................ 167
Design overview .................................................. 167
   What is a builder (DFHBS*)? ................................... 168
   Builder parameter set (BPS) .................................... 168
   TCTTE creation and deletion .................................. 169
   Component overview .......................................... 169
   DFHZCQ and TCTTE generation ................................. 170
   Patterns, hierarchies, nodes, and builders .................... 176
   The DELETE process .......................................... 180
   Completing the process description ......................... 185

Chapter 12. Built-in functions ................................ 201
Design overview .................................................. 201
   Field edit (DEEDIT) ........................................... 201
   Phonetic conversion .......................................... 201
Modules ....................................................... 201
Exits .............................................. 202
Trace .............................................. 202

Chapter 13. Business Application Manager domain (BAM) ........ 203
Business application manager domain’s specific gate .................. 203
   BATT gate, ADD_REPLACE_PROCESSTYPE function .......... 204
   BATT gate, INQUIRE_PROCESSTYPE function ................. 205
   BATT gate, START_BROWSE_PROCESSTYPE function .......... 206
   BATT gate, GET_NEXT_PROCESSTYPE function ............... 206
   BATT gate, END_BROWSE_PROCESSTYPE function ............ 206
   BATT gate, SET_PROCESSTYPE function ...................... 207
   BATT gate, DISCARD_PROCESSTYPE function ................. 207
   BATT gate, COMMIT_PROCESSTYPE_TABLE function .......... 208
   BAXM gate, INIT_ACTIVITY_REQUEST function ............... 208
   BAXM gate, BIND_ACTIVITY_REQUEST function ............... 208
   BAPR gate, ADD_PROCESS function ......................... 209
   BAPR gate, RUN_PROCESS function ............................. 209
   BAPR gate, LINK_PROCESS function ............................ 210
   BAPR gate, ACQUIRE_PROCESS function ........................ 211
   BAPR gate, CANCEL_PROCESS function ....................... 211
   BAPR gate, SUSPEND_PROCESS function ...................... 211
   BAPR gate, RESUME_PROCESS function ........................ 212
Chapter 15. CICS catalog domains
(GC/LC) ............................................. 249
CICS catalog domains’ specific gate .................. 249
CCC gate, ADD function .............................. 250
CCC gate, DELETE function ........................... 250
CCC gate, GET function .............................. 250
CCC gate, WRITE function ............................ 251
CCC gate, GET_UPDATE function .................... 251
CCC gate, PUT_REPLACE function .................... 252
CCC gate, START_BROWSE function ............... 252
CCC gate, GET_NEXT function ....................... 253
CCC gate, END_BROWSE function .................... 253
CCC gate, TYPE_PURGE function ..................... 253
CCC gate, START_WRITE function .................... 254
CCC gate, WRITE_PURGE function .................... 254
CCC gate, END_WRITE function ....................... 255
CICS catalog domains’ generic gate .................. 255
Modules ............................................. 256
Exits ............................................... 256
Trace ............................................... 256

Chapter 16. CICS-C/370 interface .................... 257
Design overview .................................... 257
Establishing the connection .......................... 259
Storage for the transaction .......................... 259
Storage acquisition .................................. 260
Control blocks ...................................... 260
Modules ............................................. 260
Exits ............................................... 260
Trace ............................................... 260
External interfaces .................................. 261
  CICS-C/370 interface parameter lists ............. 261

Chapter 17. CICS-DB2 Attachment Facility .............. 265
Design overview .................................... 265
CICS Initialization .................................. 266
CICS-DB2 Attachment startup ....................... 267
CICS-DB2 Attachment shutdown ...................... 268
CICS-DB2 mainline processing ....................... 268
Control blocks ..................................... 271
  DFHD2SS (CICS-DB2 static storage) .............. 271
  DFHD2GLB (CICS-DB2 Global block) ............... 271
  DFHD2ENT (CICS-DB2 DB2ENTRY block) .......... 272
  DFHD2TRN (CICS-DB2 DB2TRAN block) .......... 272
  DFHD2CSB (CICS-DB2 subtask block) .......... 272
  DFHD2GWA (CICS-DB2 global work area) ........ 272
  DFHD2LOT (CICS-DB2 life of task block) ....... 272
Modules ............................................. 272
Exits ............................................... 273
Trace ............................................... 273
Statistics .......................................... 273

Chapter 18. Command interpreter ...................... 275
Design overview .................................... 275
Modules ............................................. 275
Exits ............................................... 275
Trace ............................................... 275
Chapter 19. CSD utility program
(DFHCSUDP) ........................................ 277
  Design overview .................................. 277
  Modules ............................................. 278
  Exits ................................................. 278
  Trace ................................................. 279
  Statistics .......................................... 279

Chapter 20. Database control (DBCTL) ..... 281
  Design overview .................................. 281
  The connection process ............................ 282
  The interface layer ................................ 287
  DBCTL system definition ........................... 290
  DBCTL PSB scheduling .............................. 290
  Database calls ...................................... 290
  DBCTL PSB termination ............................. 290
  System termination ................................ 290
  Control blocks .................................... 291
  Modules ............................................. 291
  Exits ................................................. 292

Chapter 21. Data interchange program .... 293
  Design overview .................................. 293
  Modules ............................................. 294
  Exits ................................................. 294
  Trace ................................................. 294

Chapter 22. Directory manager
domain (DD) ........................................... 295
  Directory manager domain’s specific gates .... 295
    DDDI gate, CREATEDIRECTORY function .... 295
    DDDI gate, ADD_ENTRY function .............. 296
    DDDI gate, DELETE_ENTRY function .......... 296
    DDDI gate, REPLACE_DATA function .......... 297
    DDLO gate, LOCATE function .................. 297
    DDBR gate, START_BROWSE function .......... 298
    DDBR gate, GET_NEXT_ENTRY function ....... 298
    DDBR gate, END_BROWSE function ............ 299
  Directory manager domain’s generic gates ... 299
  Exits ................................................. 300
  Trace ................................................. 300

Chapter 23. Dispatcher domain (DS) ...... 301
  Dispatcher domain’s specific gates ............. 301
    DSAT gate, ATTACH function .................. 302
    DSAT gate, CHANGE_MODE function .......... 303
    DSAT gate, CLEAR_MATCH function .......... 305
    DSAT gate, NOTIFY_DELETE_TCB function .... 305
    DSAT gate, CHANGE_PRIORITY function ...... 305
    DSAT gate, SET_PRIORITY function .......... 305
    DSAT gate, CANCEL_TASK function ........... 306
    DSAT gate, FREE_SUBSPACE_TCBS function ... 306
    DSAT gate, DELETE_SUBSPACE_TCBS function ... 307
    DSAT gate, TCB_POOL_MANAGEMENT function . 307
    DSBR gate, START_BROWSE function .......... 308
    DSBR gate, END_BROWSE function .......... 308
    DSBR gate, GET_NEXT function ............... 308
    DSBR gate, INQUIRE_TASK function .......... 310
    DSBR gate, SET_TASK function ............... 311
    DSBR gate, INQUIRE_TCB function ............ 311
    DSBR gate, SET_TCB function ................. 311
    DSIT gate, INQUIRE_DISPATCHER function ... 312
    DSIT gate, SET_DISPATCHER function ....... 312
    DSIT gate, ACTIVATE_MODE function ......... 313
    DSIT gate, ADD_TCB function ................. 314
    DSIT gate, DELETE_TCB function .......... 315
    DSIT gate, DELETE_OPEN_TCB function ....... 315
    DSIT gate, DELETE_ALL_OPEN_TCB function ... 316
    DSIT gate, FREE_TCB function ............... 316
    DSIT gate, PROCESS_DEAD_TCBS function ..... 316
    DSSR gate, ADD_SUSPEND function ............ 317
    DSSR gate, DELETE_SUSPEND function ....... 317
    DSSR gate, INQUIRE_SUSPEND_TOKEN function . 317
    DSSR gate, SUSPEND function ................. 318
    DSSR gate, RESUME function .................. 319
    DSSR gate, WAIT_MVS function ............... 319
    DSSR gate, WAIT_OLDW function .............. 321
    DSSR gate, WAIT_OLDC function ............. 322
  Dispatcher domain’s generic gates .......... 323
  Dispatcher domain’s generic formats ....... 324
    DSAT format, TASK_REPLY function .......... 325
    DSAT format, PURGE_INHIBIT_QUERY function . 325
  Modules ............................................. 326
  Exits ................................................. 326
  Trace ................................................. 326

Chapter 24. Distributed program link .... 327
  Modules ............................................. 329
  Exits ................................................. 329
  Trace ................................................. 329

Chapter 25. Distributed transaction
processing .......................................... 331
  Design overview .................................. 331
  Distributed transaction processing with MRO
  and LU6.1 .......................................... 331
  Mapped and unmapped conversations (LU6.2) ... 332
  Modules ............................................. 333
    DFHEGL ............................................ 333
    DFHETC and DFHETL ............................... 333
    D FHZARL .......................................... 334
    D FHZARM .......................................... 336
    D FHZARQ .......................................... 337
    D FHZARR .......................................... 338
    D FHZERH .......................................... 340
    D FHZISP .......................................... 340
    D FHZSTAP ......................................... 340
  Exits ................................................. 341
  Trace ................................................. 341

Chapter 26. DL/I database support ....... 343
  Design overview .................................. 343
  The router component (DFHDLI). ................ 344
  Control blocks .................................... 345
    DL/I interface block (DIB) .................... 345
    DL/I interface parameter list (DLP) ........ 345

Contents vii
Chapter 27. Document Handler domain (DH) ........................................ 349

Document Handler domain’s specific gates .................................. 349
  DHDH gate, CREATE_DOCUMENT function .................................. 350
  DHDH gate, INSERT_DATA function ......................................... 351
  DHDH gate, INSERT_BOOKMARK function .............................. 352
  DHDH gate, REPLACE_DATA function ..................................... 353
  DHDH gate, DELETE_DOCUMENT function ............................... 354
  DHDH gate, DELETE_DATA function ..................................... 355
  DHDH gate, DELETE_BOOKMARK function ............................ 356
  DHDH gate, RETRIEVE_WITH_CTLINFO function ...................... 356
  DHDH gate, RETRIEVE_WITHOUT_CTLINFO function ............... 357
  DHDH gate, INQUIRE_DOCUMENT function ............................ 357
  DHDH gate, SET_SYMBOL_VALUE_BY_API function ...................... 358
  DHDH gate, SET_SYMBOL_VALUE_BY_SSI function .................. 358
  DHDH gate, ADD_SYMBOL_LIST function ................................ 359
  DHDH gate, EXPORT_SYMBOL_LIST function ....................... 359
  DHDH gate, IMPORT_SYMBOL_LIST function .......................... 360
  DHTM gate, INITIALIZE_DOCTEMPLATES function .................. 360
  DHTM gate, ADD_REPLACE_DOCTEMPLATE function .................. 360
  DHTM gate, READ_TEMPLATE function ................................ 361
  DHTM gate, INQUIRE_DOCTEMPLATE function ............................ 362
  DHTM gate, INQUIRE_TEMPLATE_STATUS function .................. 362
  DHTM gate, DELETE_DOCTEMPLATE function ............................ 363
  DHTM gate, START_BROWSE function .................................. 363
  DHTM gate, GET_NEXT function ...................................... 364
  DHTM gate, END_BROWSE function .................................. 364
  DHRP gate, RECOVER_DEFINITIONS function ............................ 365

Document Handler domain’s generic gates ................................ 365
  Modules ........................................................................ 366
  Exits ......................................................................... 368
  Trace ........................................................................ 368

Chapter 28. Domain manager domain (DM) .................................. 369

Domain manager domain’s specific gates .................................. 369
  DMEN gate, LISTEN function ........................................... 370
  DMEN gate, DELETE function ........................................... 370
  DMEN gate, NOTIFY_SMSVSAMOPERATIONAL function .............. 371
  DMDM gate, ADD_DOMAIN function ..................................... 371
  DMDM gate, QUIESCE_SYSTEM function .............................. 372
  DMDM gate, SET_PHASE function ...................................... 372
  DMDM gate, WAIT_PHASE function .................................... 373
  DMIQ gate, START_BROWSE function .................................. 374
  DMIQ gate, GET_NEXT function ........................................ 374
  DMIQ gate, END_BROWSE function ..................................... 375
  DMIQ gate, INQ_DOMAIN_BY_NAME function ......................... 375
  DMIQ gate, INQ_DOMAIN_BY_TOKEN function ......................... 376
  DMIQ gate, INQ_DOMAIN_BY_ID function .............................. 376
  Domain manager domain’s generic gates ................................ 377
  Domain manager domain’s generic formats ........................... 377
    DMDM format, PRE.Initialize function .............................. 378
    DMDM format, INITIALIZE_DOMAIN function .................. 378
    DMDM format, QUIESCE_DOMAIN function ....................... 379
    DMDM format, TERMINATE_DOMAIN function ..................... 379
  Modules ........................................................................ 380
  Exits ......................................................................... 380
  Trace ........................................................................ 380

Chapter 29. Dump domain (DU) .................................................. 381

Design overview ........................................................................ 381
Dump domain’s specific gates .................................................. 382
  DUDT gate, ADD_TRAN_DUMPCODE function ......................... 382
  DUDT gate, DELETE_TRAN_DUMPCODE function ..................... 384
  DUDT gate, INQUIRE_TRAN_DUMPCODE function ..................... 385
  DUDT gate, STARTBR_TRAN_DUMPCODE function ................... 386
  DUDT gate, GETNEXT_TRAN_DUMPCODE function .................... 387
  DUDT gate, ENDBR_TRAN_DUMPCODE function ....................... 388
  DUDT gate, ADD_SYSTEM_DUMPCODE function ....................... 389
  DUDT gate, DELETE_SYSTEM_DUMPCODE function .................... 390
  DUDT gate, INQUIRE_SYSTEM_DUMPCODE function .................. 391
  DUDT gate, SET_SYSTEM_DUMPCODE function ....................... 392
  DUDT gate, STARTBR_SYSTEM_DUMPCODE function .................. 393
  DUDT gate, GETNEXT_SYSTEM_DUMPCODE function .................. 394
  DUDT gate, ENDBR_SYSTEM_DUMPCODE function ..................... 395
  DUDU gate, TRANSACTION_DUMP function ............................ 397
  DUDU gate, SYSTEM_DUMP function .................................... 400
  DUSR gate, CROSS_SYSTEM_DUMP_AVAIL function .................. 402
  DUSR gate, DUMPSD_OPEN function ..................................... 403
  DUSR gate, DUMPSD_CLOSE function .................................... 403
  DUSR gate, DUMPSD_SWITCH function .................................. 403
  DUSR gate, INQUIRE_CURRENT_DUMPS function ..................... 405
  DUSR gate, INQUIRE_DUMPSD_OPEN_STATUS function ............. 405
  DUSR gate, INQUIRE_DUMPSD_AUTOSWITCH function ............... 405
  DUSR gate, SET_DUMPSD_AUTOSWITCH function ..................... 405
<table>
<thead>
<tr>
<th>Chapter 30. Dump utility program (DFHDU530)</th>
<th>419</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design overview</td>
<td>419</td>
</tr>
<tr>
<td>Data sets</td>
<td>419</td>
</tr>
<tr>
<td>Processing</td>
<td>420</td>
</tr>
<tr>
<td>Modules</td>
<td>421</td>
</tr>
<tr>
<td>Copy books</td>
<td>421</td>
</tr>
<tr>
<td>Exits</td>
<td>421</td>
</tr>
<tr>
<td>Trace</td>
<td>421</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 31. Dynamic allocation sample program (IBM 3270 only)</th>
<th>423</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design overview</td>
<td>423</td>
</tr>
<tr>
<td>Control blocks</td>
<td>424</td>
</tr>
<tr>
<td>Modules</td>
<td>424</td>
</tr>
<tr>
<td>Exits</td>
<td>425</td>
</tr>
<tr>
<td>Trace</td>
<td>425</td>
</tr>
<tr>
<td>External interfaces</td>
<td>425</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 32. Enqueue Domain (NQ)</th>
<th>427</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enqueue domain's specific gates</td>
<td>428</td>
</tr>
<tr>
<td>NQED gate, CREATE_ENQUEUE_POOL function</td>
<td>428</td>
</tr>
<tr>
<td>NQED gate, DEACTIVATE function</td>
<td>431</td>
</tr>
<tr>
<td>NQED gate, REACQUIRE_ENQUEUE function</td>
<td>431</td>
</tr>
<tr>
<td>NQED gate, SET_QNNAME_LIST function</td>
<td>433</td>
</tr>
<tr>
<td>NQED gate, ENQUEUE function</td>
<td>433</td>
</tr>
<tr>
<td>NQED gate, DEQUEUE function</td>
<td>435</td>
</tr>
<tr>
<td>NQIB gate, ENQUEUE_ENQUEUE function</td>
<td>436</td>
</tr>
<tr>
<td>NQIB gate, START_BROWSE_ENQUEUE function</td>
<td>438</td>
</tr>
<tr>
<td>NQIB gate, GET_NEXT_ENQUEUE function</td>
<td>439</td>
</tr>
<tr>
<td>NQIB gate, END_BROWSE_ENQUEUE function</td>
<td>440</td>
</tr>
<tr>
<td>NQRN gate, ENQUEUE function</td>
<td>441</td>
</tr>
<tr>
<td>NQRN gate, ADD_REPLACE_ENQMODEL function</td>
<td>442</td>
</tr>
<tr>
<td>NQRN gate, DISCARD_ENQMODEL function</td>
<td>443</td>
</tr>
<tr>
<td>NQRN gate, INQUIRE_ENQMODEL function</td>
<td>444</td>
</tr>
<tr>
<td>NQRN gate, SET_ENQMODEL function</td>
<td>445</td>
</tr>
<tr>
<td>Enqueue domain's generic gates</td>
<td>447</td>
</tr>
<tr>
<td>Modules</td>
<td>448</td>
</tr>
<tr>
<td>Exits</td>
<td>448</td>
</tr>
<tr>
<td>Trace</td>
<td>448</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 33. Enterprise Java domain (EJ)</th>
<th>451</th>
</tr>
</thead>
<tbody>
<tr>
<td>EJ domain's specific gates</td>
<td>451</td>
</tr>
<tr>
<td>EJBB gate, START_BROWSE Function</td>
<td>452</td>
</tr>
<tr>
<td>EJBB gate, GET_NEXT Function</td>
<td>453</td>
</tr>
<tr>
<td>EJBB gate, END_BROWSE Function</td>
<td>454</td>
</tr>
<tr>
<td>EJBG gate, ADD_BEAN function</td>
<td>454</td>
</tr>
<tr>
<td>EJBG gate, CONFIRM_ALL_BEANS Function</td>
<td>455</td>
</tr>
<tr>
<td>EJBG gate, DELETE_ALL_BEANS Function</td>
<td>456</td>
</tr>
<tr>
<td>EJBG gate, DELETE_BEAN Function</td>
<td>456</td>
</tr>
<tr>
<td>EJBG gate, GET_BEAN_DD Function</td>
<td>457</td>
</tr>
<tr>
<td>EJBG gate, INQUIRE_BEAN Function</td>
<td>458</td>
</tr>
<tr>
<td>EJBG gate, START_BROWSE function</td>
<td>458</td>
</tr>
<tr>
<td>EJBG gate, GET_NEXT function</td>
<td>459</td>
</tr>
<tr>
<td>EJCB gate, END_BROWSE function</td>
<td>460</td>
</tr>
<tr>
<td>EJCG gate, ACTION_CORBASERVER function</td>
<td>461</td>
</tr>
<tr>
<td>EJCG gate, ADD_CORBASERVER function</td>
<td>461</td>
</tr>
<tr>
<td>EJCG gate, AMEND_CORBASERVER function</td>
<td>462</td>
</tr>
<tr>
<td>EJCG gate, DELETE_CORBASERVER function</td>
<td>463</td>
</tr>
<tr>
<td>EJCG gate, INQUIRE_CORBASERVER function</td>
<td>463</td>
</tr>
<tr>
<td>EJCG gate, RESOLVE_CORBASERVER function</td>
<td>464</td>
</tr>
<tr>
<td>EJCG gate, SET_ALL_STATE function</td>
<td>465</td>
</tr>
<tr>
<td>EJCG gate, WAIT_FOR_CORBASERVER function</td>
<td>465</td>
</tr>
<tr>
<td>EJCP gate, DISCARD_DJAR function</td>
<td>466</td>
</tr>
<tr>
<td>EJCP gate, INSTALL_DJAR function</td>
<td>466</td>
</tr>
<tr>
<td>EJCP gate, PRE_INSTALL_DJAR function</td>
<td>467</td>
</tr>
<tr>
<td>EJCP gate, PUBLISH_CORBASERVER function</td>
<td>467</td>
</tr>
<tr>
<td>EJCP gate, PUBLISH_DJAR function</td>
<td>468</td>
</tr>
<tr>
<td>EJCP gate, RETRACT_CORBASERVER function</td>
<td>468</td>
</tr>
<tr>
<td>EJCP gate, RETRACT_DJAR function</td>
<td>469</td>
</tr>
<tr>
<td>EJDB gate, START_BROWSE function</td>
<td>469</td>
</tr>
<tr>
<td>EJDB gate, GET_NEXT function</td>
<td>470</td>
</tr>
<tr>
<td>EJDB gate, END_BROWSE function</td>
<td>471</td>
</tr>
<tr>
<td>EJCG gate, ACTION_DJAR function</td>
<td>471</td>
</tr>
<tr>
<td>EJCG gate, ADD_DJAR function</td>
<td>472</td>
</tr>
<tr>
<td>EJCG gate, AMEND_DJAR function</td>
<td>472</td>
</tr>
<tr>
<td>EJCG gate, COUNT_FOR_CS function</td>
<td>473</td>
</tr>
<tr>
<td>EJCG gate, DELETE_DJAR function</td>
<td>474</td>
</tr>
<tr>
<td>EJCG gate, DELETE_ALL_DJARS function</td>
<td>475</td>
</tr>
<tr>
<td>EJCG gate, INQUIRE_DJAR function</td>
<td>475</td>
</tr>
<tr>
<td>EJCG gate, SET_ALL_STATE function</td>
<td>476</td>
</tr>
<tr>
<td>EJCG gate, WAIT_FOR_DJAR function</td>
<td>476</td>
</tr>
<tr>
<td>EJCG gate, WAIT_FOR_USABLE_DJARS function</td>
<td>476</td>
</tr>
<tr>
<td>EJDI gate, ADD_ENTRY Function</td>
<td>477</td>
</tr>
<tr>
<td>EJDI gate, INITIALIZE Function</td>
<td>478</td>
</tr>
<tr>
<td>EJDI gate, LOOKUP_ENTRY Function</td>
<td>478</td>
</tr>
<tr>
<td>EJDI gate, REMOVE_ENTRY Function</td>
<td>479</td>
</tr>
<tr>
<td>EJDU gate, DUMP_DATA Function</td>
<td>479</td>
</tr>
</tbody>
</table>
### Chapter 34. Event manager domain (EM)

<table>
<thead>
<tr>
<th>Event manager domain’s specific gates</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>EJDU gate, DUMP_STACK Function</td>
<td>480</td>
</tr>
<tr>
<td>EJDU gate, INQUIRE_TRACE_FLAGS Function</td>
<td>480</td>
</tr>
<tr>
<td>EJGE gate, INITIALIZE Function</td>
<td>481</td>
</tr>
<tr>
<td>EJGE gate, QUIESCE Function</td>
<td>481</td>
</tr>
<tr>
<td>EJGE gate, TERMINATE Function</td>
<td>482</td>
</tr>
<tr>
<td>EJIO gate, RESOLVE Function</td>
<td>482</td>
</tr>
<tr>
<td>EJIO gate, RESOLVE_CServers Function</td>
<td>483</td>
</tr>
<tr>
<td>EJIO gate, RESOLVE_DJARS Function</td>
<td>483</td>
</tr>
<tr>
<td>EJIO gate, ADD_METHOD Function</td>
<td>484</td>
</tr>
<tr>
<td>EJIO gate, ADD_BEAN Function</td>
<td>484</td>
</tr>
<tr>
<td>EJIO gate, ADD_METHOD_INFO Function</td>
<td>485</td>
</tr>
<tr>
<td>EJIO gate, INQUIRE_OBJECT Function</td>
<td>487</td>
</tr>
<tr>
<td>EJIO gate, INITIALIZE Function</td>
<td>486</td>
</tr>
<tr>
<td>EJIO gate, RESOLVE_STORE Function</td>
<td>488</td>
</tr>
<tr>
<td>EJIO gate, RESOLVE_Stores Function</td>
<td>488</td>
</tr>
<tr>
<td>EJIO gate, RETRIEVE_STATISTICS Function</td>
<td>489</td>
</tr>
<tr>
<td>EJIO gate, START_BROWSE_OBJECT Function</td>
<td>490</td>
</tr>
<tr>
<td>EJIO gate, END_BROWSE_OBJECT Function</td>
<td>486</td>
</tr>
<tr>
<td>EJIO gate, GET_NEXT_OBJECT Function</td>
<td>487</td>
</tr>
<tr>
<td>EJIO gate, INQUIRE_OBJECT Function</td>
<td>488</td>
</tr>
<tr>
<td>EJIO gate, INQUIRE_METHOD Function</td>
<td>485</td>
</tr>
<tr>
<td>EJIO gate, INITIALIZE Function</td>
<td>486</td>
</tr>
<tr>
<td>EJIO gate, RESOLVE_STORE Function</td>
<td>493</td>
</tr>
<tr>
<td>EJIO gate, STORE_OBJECT Function</td>
<td>493</td>
</tr>
<tr>
<td>EJ domain’s generic gates</td>
<td>494</td>
</tr>
</tbody>
</table>

**Modules**

<table>
<thead>
<tr>
<th>Exit</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>EJdomain’s generic gate</td>
<td>495</td>
</tr>
</tbody>
</table>

**Trace**

<table>
<thead>
<tr>
<th>Exit</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>EJdomain’s generic gate</td>
<td>495</td>
</tr>
</tbody>
</table>

---

**Chapter 35. EXEC interface**

<table>
<thead>
<tr>
<th>Design overview</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control blocks</td>
<td>511</td>
</tr>
</tbody>
</table>

**Modules**

<table>
<thead>
<tr>
<th>Exit</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHEIP</td>
<td>523</td>
</tr>
<tr>
<td>DFHEIP</td>
<td>524</td>
</tr>
<tr>
<td>Method of calling processor modules</td>
<td>524</td>
</tr>
</tbody>
</table>

**Trace**

<table>
<thead>
<tr>
<th>Exit</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHEIP</td>
<td>525</td>
</tr>
</tbody>
</table>

---

**Chapter 36. Execution diagnostic facility (EDF)**

<table>
<thead>
<tr>
<th>Design overview</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control blocks</td>
<td>527</td>
</tr>
</tbody>
</table>

**Modules**

<table>
<thead>
<tr>
<th>Exit</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>CEBR transaction (DFHEDFBR)</td>
<td>527</td>
</tr>
<tr>
<td>EDF display (DFHEDFD)</td>
<td>527</td>
</tr>
<tr>
<td>EDF map set (DFHEDFM)</td>
<td>528</td>
</tr>
<tr>
<td>EDF control program (DFHEDFP)</td>
<td>528</td>
</tr>
<tr>
<td>EDF response table (DFHEDFR)</td>
<td>529</td>
</tr>
<tr>
<td>EDF task switch program (DFHEDFX)</td>
<td>529</td>
</tr>
</tbody>
</table>

**Trace**

<table>
<thead>
<tr>
<th>Exit</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>CEBR transaction (DFHEDFBR)</td>
<td>530</td>
</tr>
<tr>
<td>EDF display (DFHEDFD)</td>
<td>525</td>
</tr>
<tr>
<td>EDF map set (DFHEDFM)</td>
<td>528</td>
</tr>
<tr>
<td>EDF control program (DFHEDFP)</td>
<td>528</td>
</tr>
<tr>
<td>EDF response table (DFHEDFR)</td>
<td>529</td>
</tr>
<tr>
<td>EDF task switch program (DFHEDFX)</td>
<td>529</td>
</tr>
</tbody>
</table>

---

**Chapter 37. Extended recovery facility (XRF)**

<table>
<thead>
<tr>
<th>Design overview</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control blocks</td>
<td>531</td>
</tr>
</tbody>
</table>

**Modules**

<table>
<thead>
<tr>
<th>Exit</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>CEFPF</td>
<td>527</td>
</tr>
<tr>
<td>EDF control program (DFHEDFP)</td>
<td>528</td>
</tr>
<tr>
<td>EDF response table (DFHEDFR)</td>
<td>529</td>
</tr>
<tr>
<td>EDF task switch program (DFHEDFX)</td>
<td>529</td>
</tr>
</tbody>
</table>

---

**Chapter 38. External CICS interface**

<table>
<thead>
<tr>
<th>Design overview</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control blocks</td>
<td>533</td>
</tr>
</tbody>
</table>

**Modules**

<table>
<thead>
<tr>
<th>Exit</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>CEFPF</td>
<td>533</td>
</tr>
<tr>
<td>EDF control program (DFHEDFP)</td>
<td>533</td>
</tr>
<tr>
<td>EDF response table (DFHEDFR)</td>
<td>533</td>
</tr>
<tr>
<td>EDF task switch program (DFHEDFX)</td>
<td>533</td>
</tr>
</tbody>
</table>

**Trace**

<table>
<thead>
<tr>
<th>Exit</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>CEFPF</td>
<td>536</td>
</tr>
<tr>
<td>EDF control program (DFHEDFP)</td>
<td>536</td>
</tr>
<tr>
<td>EDF response table (DFHEDFR)</td>
<td>536</td>
</tr>
<tr>
<td>EDF task switch program (DFHEDFX)</td>
<td>536</td>
</tr>
</tbody>
</table>

---

**Chapter 39. Field engineering program**

<table>
<thead>
<tr>
<th>Design overview</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control blocks</td>
<td>537</td>
</tr>
</tbody>
</table>

**Trace**

<table>
<thead>
<tr>
<th>Exit</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>CEFPF</td>
<td>537</td>
</tr>
<tr>
<td>EDF control program (DFHEDFP)</td>
<td>537</td>
</tr>
<tr>
<td>EDF response table (DFHEDFR)</td>
<td>537</td>
</tr>
<tr>
<td>EDF task switch program (DFHEDFX)</td>
<td>537</td>
</tr>
</tbody>
</table>

---

**Chapter 40. File control**

<table>
<thead>
<tr>
<th>Design overview</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concurrency control</td>
<td>539</td>
</tr>
</tbody>
</table>

**Deblocking services for BDAM data sets**

<table>
<thead>
<tr>
<th>Exit</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>CEFPF</td>
<td>539</td>
</tr>
<tr>
<td>EDF control program (DFHEDFP)</td>
<td>539</td>
</tr>
<tr>
<td>EDF response table (DFHEDFR)</td>
<td>539</td>
</tr>
<tr>
<td>EDF task switch program (DFHEDFX)</td>
<td>539</td>
</tr>
</tbody>
</table>

**Sequential retrieval**

<table>
<thead>
<tr>
<th>Exit</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>CEFPF</td>
<td>540</td>
</tr>
<tr>
<td>EDF control program (DFHEDFP)</td>
<td>540</td>
</tr>
<tr>
<td>EDF response table (DFHEDFR)</td>
<td>540</td>
</tr>
<tr>
<td>EDF task switch program (DFHEDFX)</td>
<td>540</td>
</tr>
</tbody>
</table>

**Read Integrity**

<table>
<thead>
<tr>
<th>Exit</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>CEFPF</td>
<td>541</td>
</tr>
<tr>
<td>EDF control program (DFHEDFP)</td>
<td>541</td>
</tr>
<tr>
<td>EDF response table (DFHEDFR)</td>
<td>541</td>
</tr>
<tr>
<td>EDF task switch program (DFHEDFX)</td>
<td>541</td>
</tr>
</tbody>
</table>

**Backout logging**

<table>
<thead>
<tr>
<th>Exit</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>CEFPF</td>
<td>541</td>
</tr>
<tr>
<td>EDF control program (DFHEDFP)</td>
<td>541</td>
</tr>
<tr>
<td>EDF response table (DFHEDFR)</td>
<td>541</td>
</tr>
<tr>
<td>EDF task switch program (DFHEDFX)</td>
<td>541</td>
</tr>
</tbody>
</table>

**Forward Recovery Logging**

<table>
<thead>
<tr>
<th>Exit</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>CEFPF</td>
<td>542</td>
</tr>
<tr>
<td>EDF control program (DFHEDFP)</td>
<td>542</td>
</tr>
<tr>
<td>EDF response table (DFHEDFR)</td>
<td>542</td>
</tr>
<tr>
<td>EDF task switch program (DFHEDFX)</td>
<td>542</td>
</tr>
</tbody>
</table>

**Automatic journaling and logging**

<table>
<thead>
<tr>
<th>Exit</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>CEFPF</td>
<td>542</td>
</tr>
<tr>
<td>EDF control program (DFHEDFP)</td>
<td>542</td>
</tr>
<tr>
<td>EDF response table (DFHEDFR)</td>
<td>542</td>
</tr>
<tr>
<td>EDF task switch program (DFHEDFX)</td>
<td>542</td>
</tr>
</tbody>
</table>

**Use of concurrent tasks**

<table>
<thead>
<tr>
<th>Exit</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>CEFPF</td>
<td>542</td>
</tr>
<tr>
<td>EDF control program (DFHEDFP)</td>
<td>542</td>
</tr>
<tr>
<td>EDF response table (DFHEDFR)</td>
<td>542</td>
</tr>
<tr>
<td>EDF task switch program (DFHEDFX)</td>
<td>542</td>
</tr>
</tbody>
</table>

**Shared Data table services**

<table>
<thead>
<tr>
<th>Exit</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>CEFPF</td>
<td>542</td>
</tr>
<tr>
<td>EDF control program (DFHEDFP)</td>
<td>542</td>
</tr>
<tr>
<td>EDF response table (DFHEDFR)</td>
<td>542</td>
</tr>
<tr>
<td>EDF task switch program (DFHEDFX)</td>
<td>542</td>
</tr>
</tbody>
</table>

**Coupling facility data tables server**

<table>
<thead>
<tr>
<th>Exit</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>CEFPF</td>
<td>542</td>
</tr>
<tr>
<td>EDF control program (DFHEDFP)</td>
<td>542</td>
</tr>
<tr>
<td>EDF response table (DFHEDFR)</td>
<td>542</td>
</tr>
<tr>
<td>EDF task switch program (DFHEDFX)</td>
<td>542</td>
</tr>
</tbody>
</table>

**How CICS processes file control requests**

<table>
<thead>
<tr>
<th>Exit</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>CEFPF</td>
<td>542</td>
</tr>
<tr>
<td>EDF control program (DFHEDFP)</td>
<td>542</td>
</tr>
<tr>
<td>EDF response table (DFHEDFR)</td>
<td>542</td>
</tr>
<tr>
<td>EDF task switch program (DFHEDFX)</td>
<td>542</td>
</tr>
<tr>
<td>Module</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>------</td>
</tr>
<tr>
<td>ONEY</td>
<td>563</td>
</tr>
<tr>
<td>Control blocks</td>
<td>552</td>
</tr>
<tr>
<td>Access method control block (ACB)</td>
<td>553</td>
</tr>
<tr>
<td>Data control block (DCB)</td>
<td>554</td>
</tr>
<tr>
<td>Data set name block (DSNB)</td>
<td>555</td>
</tr>
<tr>
<td>File browse work area (FBWA)</td>
<td>555</td>
</tr>
<tr>
<td>File control static storage (FC static)</td>
<td>556</td>
</tr>
<tr>
<td>File control quiesce receive element (FCQRE)</td>
<td>556</td>
</tr>
<tr>
<td>File control quiesce send element (FCQSE)</td>
<td>556</td>
</tr>
<tr>
<td>File control coupling facility data table pool element (FCPE)</td>
<td>557</td>
</tr>
<tr>
<td>File control coupling facility data table pool wait element (FCPW)</td>
<td>557</td>
</tr>
<tr>
<td>File control table entry (FCCTE)</td>
<td>558</td>
</tr>
<tr>
<td>File control table entry (FCPW)</td>
<td>558</td>
</tr>
<tr>
<td>File control coupling facility data tables UOW pool block (FCUP)</td>
<td>558</td>
</tr>
<tr>
<td>File input/output area (FIOA)</td>
<td>559</td>
</tr>
<tr>
<td>File lasting access block (FLAB)</td>
<td>559</td>
</tr>
<tr>
<td>File control locks locator blocks (FLLBs)</td>
<td>560</td>
</tr>
<tr>
<td>File request anchor block (FRAB)</td>
<td>560</td>
</tr>
<tr>
<td>File request thread elements (FRTEs)</td>
<td>561</td>
</tr>
<tr>
<td>Keypoint list element (KPLE)</td>
<td>562</td>
</tr>
<tr>
<td>Shared resources control (SHRCTL) block</td>
<td>562</td>
</tr>
<tr>
<td>VSAM work area (VSWA)</td>
<td>563</td>
</tr>
<tr>
<td>Modules</td>
<td>563</td>
</tr>
<tr>
<td>DFHEIFC (file control EXEC interface module)</td>
<td>565</td>
</tr>
<tr>
<td>DFHCAT (file control catalog manager)</td>
<td>566</td>
</tr>
<tr>
<td>DFHCBD (file control BDAM request processor)</td>
<td>567</td>
</tr>
<tr>
<td>DFHFCCA (file control RLS control ACB manager)</td>
<td>568</td>
</tr>
<tr>
<td>DFHFCDL (file control CFDT load program)</td>
<td>569</td>
</tr>
<tr>
<td>DFHCBDN (file control DSN block manager)</td>
<td>569</td>
</tr>
<tr>
<td>DFHFCDO (file control CFDT open/close program)</td>
<td>571</td>
</tr>
<tr>
<td>DFHFCDR (file control CFDT request processor)</td>
<td>571</td>
</tr>
<tr>
<td>DFHFCDTS (file control shared data table request program)</td>
<td>571</td>
</tr>
<tr>
<td>DFHFCDTX (file control shared data table function ship program)</td>
<td>571</td>
</tr>
<tr>
<td>DFHFCDU (file control CFDT UOW calls program)</td>
<td>572</td>
</tr>
<tr>
<td>DFHFCDW (file control CFDT RMC program)</td>
<td>572</td>
</tr>
<tr>
<td>DFHFCDY (file control CFDT resynchronization program)</td>
<td>572</td>
</tr>
<tr>
<td>DFHFCES (file control ENF servicer)</td>
<td>572</td>
</tr>
<tr>
<td>DFHFCFL (file control FRAB and FLAB processor)</td>
<td>572</td>
</tr>
<tr>
<td>DFHFCFR (file control file request handler)</td>
<td>572</td>
</tr>
<tr>
<td>DFHFCFS (file control file state program)</td>
<td>574</td>
</tr>
<tr>
<td>DFHFCIN1 (file control initialization program 1)</td>
<td>577</td>
</tr>
<tr>
<td>DFHFCIN2 (file control initialization program 2)</td>
<td>578</td>
</tr>
<tr>
<td>DFHFCIR (file control initialize recovery)</td>
<td>578</td>
</tr>
<tr>
<td>DFHFCI (file control shared resources pool processor)</td>
<td>579</td>
</tr>
<tr>
<td>DFHFCII (file control log failures handler)</td>
<td>580</td>
</tr>
<tr>
<td>DFHFCII (file control logging and journaling program)</td>
<td>580</td>
</tr>
<tr>
<td>DFHFCMT (file control table manager)</td>
<td>580</td>
</tr>
<tr>
<td>DFHFCN (file control open/close program)</td>
<td>583</td>
</tr>
<tr>
<td>DFHFCNG (file control non-RLS lock handler)</td>
<td>587</td>
</tr>
<tr>
<td>DFHFCNR (file control offsite recovery completion)</td>
<td>587</td>
</tr>
<tr>
<td>DFHFCQI (file control RLS quiesce initiation)</td>
<td>588</td>
</tr>
<tr>
<td>DFHFCQR (file control quiesce receive transaction)</td>
<td>588</td>
</tr>
<tr>
<td>DFHFCQS (file control RLS quiesce send transaction)</td>
<td>588</td>
</tr>
<tr>
<td>DFHFCQT (file control RLS quiesce common system transaction)</td>
<td>588</td>
</tr>
<tr>
<td>DFHFCQU (file control RLS quiesce processor)</td>
<td>588</td>
</tr>
<tr>
<td>DFHFCQX (file control RLS quiesce exit)</td>
<td>589</td>
</tr>
<tr>
<td>DFHFCRC (file control recovery control program)</td>
<td>589</td>
</tr>
<tr>
<td>DFHFCRD (file control RLS cleanup transaction)</td>
<td>591</td>
</tr>
<tr>
<td>DFHFCRF (file control function shipping interface module)</td>
<td>591</td>
</tr>
<tr>
<td>DFHFCRL (file control share control block manager)</td>
<td>592</td>
</tr>
<tr>
<td>DFHFCRO (file control RLS open/close program)</td>
<td>593</td>
</tr>
<tr>
<td>DFHFCRP (file control restart program)</td>
<td>593</td>
</tr>
<tr>
<td>DFHFCRR (file control RLS restart)</td>
<td>595</td>
</tr>
<tr>
<td>DFHFCRS (file control RLS record management processor)</td>
<td>596</td>
</tr>
<tr>
<td>DFHFCRV (file control RLS VSAM interface processor)</td>
<td>596</td>
</tr>
<tr>
<td>DFHFCSD (file control shutdown program)</td>
<td>596</td>
</tr>
<tr>
<td>DFHFCST (file control statistics program)</td>
<td>597</td>
</tr>
<tr>
<td>DFHFCVR (file control VSAM interface program)</td>
<td>598</td>
</tr>
<tr>
<td>DFHFCVS (file control VSAM request processor)</td>
<td>599</td>
</tr>
</tbody>
</table>

Parameter lists | 600  |
| FCIA CHECK function | 600  |
| FCIA COLD_START_RLS function | 601  |
| FCIA DRAIN_CONTROL_ACB function | 601  |
| FCIA INQUIRE_RECOVERY function | 602  |
| FCIA LOST_LOCKS_COMPLETE function | 602  |
| FCIA QUESIE_COMPLETE function | 603  |
| FCIA QUESIE_REQUEST function | 604  |
| FCIA REGISTER_CONTROL_ACB function | 605  |
| FCIA RELEASE_LOCKS function | 605  |
| FCIA RESET_NONRLS_BATCH function | 606  |
| FCIA RETAIN_DATASET_LOCKS function | 606  |
| FCIA RETAIN_UOW_LOCKS function. | 607  |
| FCIA UNREGISTER_CONTROL_ACB function | 607  |
| FCCI INQUIRE function. | 608  |
| FCCR POINT function | 608  |
| FCCR HIGHEST function | 609  |
| FCCR READ function | 609  |
| FCCR READ_DELETE function | 611  |
| FCCR UNLOCK function | 611  |
| FCCR LOAD function | 612  |
| FCCR WRITE function | 612  |
| FCCR REWRITE function | 613  |
| FCCR DELETE function | 615  |
| FCCR DELETE_MULTIPLE function. | 616  |
| FCCT OPEN function | 617  |
| FCCT CLOSE function | 619  |
Chapter 42. Function shipping ........ 673
Design overview ................ 673
Application programming functions with CICS function shipping .... 674
Local and remote names .......... 674
Mirror transactions .......... 674
Initialization of CICS for CICS function shipping .......... 675
Communication with a remote system ................. 675
Protocols .................. 676
CICS function shipping environment .......... 676
CICS function shipping—handling of EXEC ............ 679
CICS commands .......... 679
CICS function shipping—handling of DL/I requests .......... 684
Terminal control support for CICS function shipping .......... 686
NOCHECK option function handling .......... 688
Exits .................. 689
Trace .................. 689

Chapter 43. “Good morning” message program .......... 691
Design overview ................ 691
Modules .................. 691
Exits .................. 691
Trace .................. 691

Chapter 44. IIOP domain (II) ........ 693
IIOP domain’s specific gates .......... 693
IILS gate, ADD_LOGICAL_SERVER function .......... 693
IILS gate, DELETE_LOGICAL_SERVER function .......... 694
IILS gate, PUBLISH_LOGICAL_SERVER function .......... 694
IILS gate, RETRACT_LOGICAL_SERVER function .......... 694
IIMM gate, ADD_REPLACE_RQMODEL function .......... 695
IIMM gate, DELETE_RQMODEL function .......... 696
IIMM gate, COMMIT_RQMODELS function .......... 696
IIRH gate, FIND_REQUEST_STREAM function .......... 697
IIRH gate, PARSE function .......... 697
IIRP gate, GET_INITIAL_DATA function .......... 698
IIRP gate, RECEIVE_REQUEST function .......... 699
IIRP gate, INVOKE function .......... 700
IIRP gate, RECEIVE_REPLY function .......... 700

xii  CICS Transaction Server: CICS Diagnosis Reference
Chapter 45. Interregion communication (IRC) ........ 709
Design overview ......................................... 709
Control blocks ........................................... 710
  Terminal control layer .................................. 710
  DFHIR layer ............................................ 712
  Terminal control layer and DFHIR layer .............. 714
MRO ECB summary ......................................... 715
Modules ................................................... 715
  Interregion programs .................................. 716
  CICS address space modules ............................ 717
Exits ................................................................ 720
Trace ......................................................... 720

Chapter 46. Intersystem communication (ISC) .......... 721

Chapter 47. Interval control ................................. 723
Design overview ........................................... 723
Time of day .................................................. 723
  Time-dependent task synchronization ..................... 723
  Automatic time-ordered transaction initiation ......... 723
  Time-of-day control ..................................... 723
Control blocks ............................................. 724
Modules ...................................................... 724
Exits .................................................................. 725
Trace ........................................................... 725

Chapter 48. JVM domain (SJ) ............................... 727
JVM domain’s specific gates ............................... 727
  SJNI gate, INVOKE_JAVA_PROGRAM function .... 727
  SJS gate, INQUIRE_JVM_POOL function ............ 728
  SJS gate, SET_JVM_POOL function .................. 728
Modules ....................................................... 728
Exits .................................................................. 729
Trace ........................................................... 729

Chapter 49. Kernel domain (KE) ......................... 731
Kernel domain’s specific gates ......................... 731
  KEAR gate, DEREREGISTER function .................. 733
  KEAR gate, READY function ............................ 733
  KEAR gate, REGISTER function ....................... 733
  KEAR gate, WAITPRED function ....................... 733
  KEDD gate, ADD_DOMAIN function ................... 734
KEDD gate, INQUIRE_DOMAIN_BY_TOKEN function .... 734
KEDD gate, INQUIRE_DOMAIN_BY_NAME function ........ 735
KEDD gate, SET_ANCHOR function ...................... 735
KEDD gate, INQUIRE_ANCHOR function ................ 735
KEDD gate, ADD_GATE function .......................... 736
KEDD gate, DELETE_GATE function ..................... 736
KEDD gate, INQUIRE_GLOBAL_TRACE function ....... 737
KEDD gate, SET_GLOBAL_TRACE function ............. 737
KEDD gate, INQUIRE_DOMAIN_TRACE function ........ 738
KEDD gate, SET_DOMAIN_TRACE function .............. 739
KEDD gate, PERFORM_SYSTEM_ACTION function ....... 739
KEDD gate, SET_TRAP_OFF function .................... 740
KEDD gate, SET_TRAP_ON function ..................... 740
KEDD gate, SET_DEFAULT_RECOVERY function .......... 741
KEDS gate, ABNORMALLY_TERMINATE_TASK function . 741
KEDS gate, CREATE_TASK function .................... 741
KEDS gate, CREATE_TCB function ....................... 742
KEDS gate, DETACH_TERMINATED_OWNTCBS function .... 743
KEDS gate, END_TASK function ......................... 743
KEDS gate, FREE_TCBS function .......................... 744
KEDS gate, PUSH_TASK function ......................... 744
KEDS gate, POP_TASK function ........................... 744
KEDS gate, READ_TIME function ......................... 744
KEDS gate, RESET_TIME function ....................... 745
KEDS gate, STOP_RUNAWAY_TIMER function ........... 745
KEDS gate, START_RUNAWAY_TIMER function ........... 745
KEDS gate, RESTORE_STIMER function ................. 746
KEDS gate, SEND_DEFERRED_ABEND function ........... 746
KEDS gate, START_PURGE_PROTECTION function ........ 746
KEDS gate, STOP_PURGE_PROTECTION function ........ 747
KEDS gate, PROCESS_KETA_ERROR function ............ 747
KEDG gate, INQUIRE_KERNEL function .................... 747
KEDG gate, SET_KERNEL function ........................ 748
KETI gate, RESET_LOCAL_TIME function .............. 749
KETI gate, REQUEST_NOTIFY_OF_A_RESET function .... 749
KETI gate, CONVERT_TO_DECIMAL_TIME function ...... 750
KETI gate, CONVERT_TO_STCK_FORMAT function ....... 751
KETI gate, TRANSACTION_INITIALISATION function .... 751
KETI gate, CONVERT_TO_STCK_FORMAT function ....... 751
KEXM gate, TRANSACTION_INITIALISATION function .... 752
Kernel domain’s generic formats ...................... 752
KEDS format, TASK_REPLY function ................. 752

Contents xiii
Chapter 53. Master terminal program 817
Design overview .......................... 817
Modules .................................. 817
Exits ..................................... 817
Trace ..................................... 817

Chapter 54. Message domain (ME) 819
Message domain’s specific gates ........ 819
  MEBM gate, RETRIEVE_MESSAGE function ... 819
  MEBM gate, INQUIRE_MESSAGE_LENGTH function ... 820
  MEBM gate, INQUIRE_MESSAGE_DEFINITION function ... 820
  MEME gate, SEND_MESSAGE function ... 821
  MEME gate, CONVERSE function ... 822
  MEME gate, RETRIEVE_MESSAGE function ... 823
  MEME gate, INQUIRE_MESSAGE_LENGTH function ... 824
  MEME gate, VALIDATE_LANGUAGE_CODE function ... 825
  MEME gate, VALIDATE_LANGUAGE_SUFFIX function ... 825
  MEME gate, INQUIRE_MESSAGE function ... 826
  MNSR gate, SET_MESSAGE_OPTIONS function ... 827
Message domain’s generic gate ........ 828
Modules .................................. 828
Exits ..................................... 829
Trace ..................................... 829

Chapter 55. Message generation program 831
Design overview ......................... 831
Modules .................................. 831
Exits ..................................... 831
Trace ..................................... 832

Chapter 56. Message switching. 833
Design overview ......................... 833
Control blocks .......................... 834
Modules .................................. 834
Exits ..................................... 835
Trace ..................................... 835
External interfaces ..................... 835

Chapter 57. Monitoring domain (MN) 837
Monitoring domain’s specific gates .... 837
  MNMN gate, EXCEPTION_DATA_PUT function ... 837
  MNMN gate, PERFORMANCE_DATA_PUT function ... 838
  MNMN gate, INQUIRE_MONITORING_DATA function ... 838
  MNMR gate, MONITOR function ... 839
  MNSR gate, SET_MCT_SUFFIX function ... 839
  MNSR gate, SET_MONITORING function ... 840
  MNSR gate, INQ_MONITORING function ... 841
  MNXM gate, TRANSACTION_INITIALIZATION function ... 842
  MNXM gate, TRANSACTION_TERMINATION function ... 843
Monitoring domain’s generic gate .... 843

Chapter 58. Multiregion operation (MRO) 847

Chapter 59. Node abnormal condition program 849
Design overview ......................... 849
Control blocks .......................... 852
Modules .................................. 853
Exits ..................................... 853
Trace ..................................... 853
Statistics ............................... 853

Chapter 60. Node error program 855
Design overview ......................... 855
Modules .................................. 855
Exits ..................................... 855
Trace ..................................... 856

Chapter 61. Object Transaction Service domain (OT) 857
Object Transaction Service domain’s specific gates 857
  OTTR gate, IMPORT_TRAN function ... 857
  OTTR gate, BEGIN_TRAN function ... 858
  OTTR gate, COMMIT_ONE_PHASE function ... 859
  OTTR gate, PREPARE function ... 859
  OTTR gate, COMMIT function ... 859
  OTTR gate, SET_ROLLBACK_ONLY function ... 860
  OTSU gate, ADD_SUBORDINATE function ... 860
  OTSU gate, SET_VOTE function ... 860
  OTSU gate, FORGET function ... 861
  OTSU gate, RESYNC function ... 861
  OTCO gate, SET_COORDINATOR function ... 862
  OTCO gate, FORGET function ... 862
Modules .................................. 863
Exits ..................................... 863
Trace ..................................... 863

Chapter 62. Parameter manager domain (PA) 865
Parameter manager domain’s specific gate. 865
  PAGP gate, FORCE_START function ... 865
  PAGP gate, GET_PARAMETERS function ... 866
  PAGP gate, INQUIRE_START function ... 866
Parameter manager domain’s generic gate. 866
Modules .................................. 867
Exits ..................................... 867
Trace ..................................... 867

Chapter 63. Partner resource manager 869
Functions provided by the partner resource manager 869
  PRCM format, INQUIRE_PARTNER function ... 869
  PRCM format, START_PARTNER_BROWSE function ... 870
  PRCM format, GET_NEXT_PARTNER function ... 870
Chapter 64. Program control ..... 875
Design overview ..... 875
Services in response to requests ..... 875
Modules ..... 875
DFHEPC ..... 875
Exits ..... 878
Trace ..... 878

Chapter 65. Program error program 879
Design overview ..... 879
Control blocks ..... 879
Modules ..... 879
Exits ..... 879
Trace ..... 879

Chapter 66. Program manager domain (PG) ..... 881
Program manager domain's specific gates ..... 881
PGAQ gate, INQUIRE_AUTOINSTALL function ..... 882
PGAQ gate, SET_AUTOINSTALL function ..... 883
PGDD gate, DEFINE_PROGRAM function ..... 883
PGDD gate, DELETE_PROGRAM function ..... 885
PGEx gate, INITIALIZE_EXIT function ..... 886
PGEx gate, TERMINATE_EXIT function ..... 887
PGHm gate, SET_CONDITIONS function ..... 887
PGHm gate, INQ_CONDITIONS function ..... 888
PGHm gate, IGNORE_CONDITIONS function ..... 889
PGHm gate, SET_AID function ..... 889
PGHm gate, INQ_AID function ..... 890
PGHm gate, SET_ABEND function ..... 891
PGHm gate, INQ_ABEND function ..... 892
PGHm gate, PUSH_HANDLE function ..... 893
PGHm gate, POP_HANDLE function ..... 893
PGHm gate, FREE_HANDLE_TABLES function ..... 894
PGHm gate, CLEAR_LABELS function ..... 894
PGIs gate, INQUIRE_PROGRAM function ..... 894
PGIs gate, INQUIRE_CURRENT_PROGRAM function ..... 897
PGIs gate, SET_PROGRAM function ..... 900
PGIs gate, START_BROWSE_PROGRAM function ..... 901
PGIs gate, GET_NEXT_PROGRAM function ..... 902
PGIs gate, END_BROWSE_PROGRAM function ..... 904
PGIs gate, REFRESH_PROGRAM function ..... 905
PGLD gate, LOAD_EXEC function ..... 906
PGLD gate, LOAD function ..... 906
PGLD gate, RELEASE_EXEC function ..... 907
PGLD gate, RELEASE function ..... 907
Pgle gate, LINK_EXEC function ..... 908
PGLK gate, LINK function ..... 909
PGLK gate, LINK_PLT function ..... 910
PGLU gate, LINK_URM function ..... 910
Ppg gate, INITIAL_LINK function ..... 911
PGRE gate, PREPARE_RETURN function ..... 912
PGXe gate, PREPARE_XCTL_EXEC function ..... 912
PGXM gate, INITIALIZE_TRANSACTION function ..... 913
PGXM gate, TERMINATE_TRANSACTION function ..... 913
Program manager domain's generic gates ..... 914
Initialize domain ..... 914
Quiesce domain ..... 914
Terminate domain ..... 915
Modules ..... 915
Exits ..... 916
Trace ..... 916

Chapter 67. Program preparation utilities ..... 917
Design overview ..... 917
Modules ..... 917
Exits ..... 917
Trace ..... 917

Chapter 68. Recovery Manager Domain (RM). ..... 919
Recovery Manager Domain's specific gates ..... 919
RMUW gate, CREATE_UOW function ..... 920
RMUW gate, INQUIRE_UOW_ID function ..... 921
RMUW gate, INQUIRE_UOW_TOKEN function ..... 921
RMUW INQUIRE_UOW function ..... 922
RMUW gate, SET_UOW function ..... 924
RMUW gate, COMMIT_UOW function ..... 924
RMUW gate, FORCE_UOW function ..... 925
RMUW gate, START_UOW_BROWSE function ..... 925
RMUW gate, GET_NEXT_UOW function ..... 926
RMUW gate, END_UOW_BROWSE function ..... 928
RMUW gate, BACKOUT_UOW function ..... 928
RMUW gate, BIND_UOW_TO_TXN function ..... 929
RMUW gate, REATTACH_REPLY function ..... 929
RMLn gate, ADD_LINK function ..... 929
RMLn gate, DELETE_LINK function ..... 931
RMLn gate, INQUIRE_LINK function ..... 931
RMLn gate, SET_LINK function ..... 933
RMLn gate, ISSUE_PREPARE function ..... 935
RMLn gate, INBOUND_FLOW function ..... 935
RMLn gate, INITIATE_RECOVERY function ..... 936
RMLn gate, SET_RECOVERY_STATUS function ..... 937
RMLn gate, REPORT_RECOVERY_STATUS function ..... 938
RMLn gate, TERMINATE_RECOVERY function ..... 939
RMLn gate, SET_MARK function ..... 939
RMLn gate, START_LINK_BROWSE function ..... 940
RMLn gate, GET_NEXT_LINK function ..... 941
RMLn gate, END_LINK_BROWSE function ..... 943
RMNM gate, INQUIRE_LOGNAME function ..... 943
RMNM gate, SET_LOGNAME function ..... 944
Chapter 70. Remote DL/I. 975
  Design overview 975
  System definition 975
  DL/I PSB scheduling 975
  Database calls 975
  DL/I PSB termination 975
  Control blocks 975

Chapter 71. RequestStream domain (RZ) 979
  RequestStream domain’s specific gates 979
    RZSO gate, CREATE function 979
    RZSO gate, SEND_REQUEST function 981
    RZSO gate, RECEIVE_REPLY function 981
    RZSO gate, LEAVE function 982
    RZSO gate, JOIN function 983
    RZSO gate, IS_ID_LOCAL function 984
    RZTA gate, RECEIVE_REQUEST function 984
    RZTA gate, SEND_REPLY function 985
    RZTA gate, GET_SERVER_DATA function 986
    RZTA gate, GET_PUBLIC_ID function 987
    RZTA gate, GET_CURRENT function 987
    RZTA gate, SET_EXIT_PROGRAM function 989
    RZRJ gate, PERFORM_JOIN function 989
  RequestStream domain’s generic gates 990
  RequestStream domain formats 990
    LSTN gate, REGISTER function 990
    LSTN gate, LISTEN function 991
    LSTN gate, CANCEL function 992
    LSTN gate, DEREGISTER function 993
    NOTI gate, NOTIFY function 993
  Modules 994
  Exits 995
  Trace 995

Chapter 72. Resource definition online (RDO) 997
  Design overview 997
  Modules 997
  Exits 999
  Trace 1000

Chapter 73. SAA Communications and Resource Recovery interfaces. 1001
  Design overview 1002
    The SAA Communications interface 1002
    The SAA Resource Recovery interface 1003
  Functions provided by the CPI component 1004
    CPIN format, START_INIT function 1004
    CPIN format, COMPLETE_INIT function 1005
    CPS format, SYNCPOINT_REQUEST function 1005
  Modules 1005
  Exits 1006
  Trace 1006

Chapter 74. Scheduler Services domain (SH) 1007
Chapter 75. Security manager domain

Security manager domain’s specific gates.

XSAD gate, ADD_USER_WITH_PASSWORD function.
XSAD gate, ADD_USER_WITHOUT_PASSWORD function.
XSAD gate, DELETE_USER_SECURITY function.
XSAD gate, INQUIRE_USER_ATTRIBUTES function.
XSAD gate, VALIDATE_USERID function.
XSFL gate, FLATTEN_USER_SECURITY function.
XSFL gate, UNFLATTEN_USER_SECURITY function.
XSIS gate, INQUIRE_REGION_USER function.
XSIS gate, INQUIRE_SECURITY_DOMAIN_PARMS function.
XSIS gate, SET_SECURITY_DOMAIN_PARMS function.
XSIS gate, SET_NETWORK_IDENTIFIER function.
XSLU gate, GENERATE_APPC_BIND function.
XSLU gate, GENERATE_APPC_RESPONSE function.
XSLU gate, VALIDATE_APPC_RESPONSE function.
XSPW gate, CREATE_PASSTICKET function.
XSPW gate, INQUIRE_PASSWORD_DATA function.
XSPW gate, UPDATE_PASSWORD_DATA function.
XRSC gate, CHECK_CICS_RESOURCE function.
XRSC gate, CHECK_CICS_COMMAND function.
XRSC gate, CHECK_SURROGATE_USER function.
XSRC gate, CHECK_NON_CICS_RESOURCE function.
XSRC gate, REBUILD_RESOURCE_CLASSES function.
XSXM gate, ADD_TRANSACTION_SECURITY function.
XSXM gate, DEL_TRANSACTION_SECURITY function.
Security manager domain’s generic gates.

Modules.

Exits.

Trace.

Chapter 76. Sign-on component

Sign-on component’s subroutines.

DFHSNAS subroutine, SIGNON_ATI_SESSION function.
DFHSNAS subroutine, SIGNOFF_ATI_SESSION function.
DFHSNPU subroutine, SIGNON_PRESET_USERID function.
DFHSNPU subroutine, SIGNOFF_PRESET_USERID function.
DFHSNSG subroutine, SIGNOFF_SURROGATE function.
DFHSNSU subroutine, SIGNON_SESSION_USERID function.
DFHSNSU subroutine, SIGNOFF_SESSION_USERID function.
DFHSNTU subroutine, SIGNON_TERMINAL_USER function.
DFHSNTU subroutine, SIGNOFF_TERMINAL_USER function.
DFHSNSU subroutine, SIGNON_ATTACH_HEADER function.
DFHSNSU subroutine, SIGNOFF_ATTACH_HEADER function.

Modules.

Exits.

Trace.

Chapter 77. Socket domain (SO)

Socket domain’s specific gates.

SOCK gate, SEND function.
SOCK gate, SEND_SSL_DATA function.
SOCK gate, RECEIVE function.
SOCK gate, RECEIVE_SSL_DATA function.
SOCK gate, CLOSE function.
SOCK gate, LISTEN function.
SORD gate, REGISTER function.
SORD gate, DEREISTER function.
SORD gate, IMMCLOSE function.
SOIS gate, SET_PARAMETERS function.
SOIS gate, INITIALIZE_ENVIRONMENT function.
SOIS gate, INQUIRE_ENVIRONMENT function.
SOIS gate, SET function.
SOIS gate, INQUIRE_STATISTICS function.
SOIS gate, VERIFY function.
SOIS gate, EXPORT_CERTIFICATE_DATA function .............. 1069
SOIS gate, IMPORT_CERTIFICATE_DATA function .............. 1069
SOIS gate, DELETE_CERTIFICATE_DATA function .............. 1070
SOAD gate, ADD_REPLACE_TCPIPSERVICE function .............. 1070
SOAD gate, DELETE_TCPIPSERVICE function ..................... 1070
SOTB gate, INQUIRE_TCPIPSERVICE function ..................... 1071
SOTB gate, START_BROWSE function .................. 1072
SOTB gate, GET_NEXT function .................. 1073
SOTB gate, END_BROWSE function .................. 1074
SOTB gate, SET_TCPIPSERVICE function .................. 1074
SOSE gate, INITIALIZE_SSL function .................. 1075
SOSE gate, SECURE_SOC_INIT function .................. 1075
SOSE gate, SECURE_SOC_READ function .................. 1076
SOSE gate, SECURE_SOC_WRITE function .................. 1076
SOSE gate, SECURE_SOC_CLOSE function .................. 1077
SOSE gate, SECURE_SOC_RESET function .................. 1077
SOSE gate, TERMINATE_SSL function .................. 1077
SOSE gate, EXPORT_CERTIFICATE_DATA function .............. 1078
SOSE gate, IMPORT_CERTIFICATE_DATA function .............. 1078
SOSE gate, DELETE_CERTIFICATE_DATA function .............. 1079
Socket domain’s generic gates .................................. 1079
Modules .................................. 1080
Exits .................................. 1081

Chapter 78. Statistics domain (ST) 1083
Statistics domain’s specific gate .................................. 1083
STST gate, INQ_STATISTICS_OPTIONS function .............. 1083
STST gate, SET_STATISTICS_OPTIONS function .............. 1084
STST gate, REQUEST_STATISTICS function .............. 1084
STST gate, RECORD_STATISTICS function .................. 1085
STST gate, STATISTICS_COLLECTION function .............. 1085
STST gate, DISABLE_STATISTICS function .................. 1086
Statistics domain’s generic gates .................................. 1087
Statistics domain’s generic format .............................. 1087
STST format, COLLECT_STATISTICS function .............. 1088
STST format, COLLECT_RESOURCE_STATS function .............. 1088
Modules .................................. 1089
Exits .................................. 1089
Trace .................................. 1089

Chapter 79. Statistics utility program (DFHSTUP) 1091
Design overview .................................. 1091
DFHSTUP operation .................................. 1092
Modules .................................. 1093

Chapter 80. Storage control macro-compatibility interface 1095
Design overview .................................. 1095

Chapter 81. Storage manager domain (SM) 1097
Storage manager domain’s specific gates .............. 1097
SMAD gate, ADD_SUBPOOL function .............. 1098
SMAD gate, DELETE_SUBPOOL function .............. 1099
SMAR gate, ALLOCATE_TRANSACTION_STG function .............. 1099
SMAR gate, RELEASE_TRANSACTION_STG function .............. 1100
SMCK gate, CHECK_STORAGE function .............. 1100
SMCK gate, RECOVER_STORAGE function .............. 1101
SMGF gate, GETMAIN function .............. 1101
SMGF gate, FREEMAIN function .............. 1102
SMGF gate, INQUIRE_ELEMENT_LENGTH function .............. 1103
SMMC gate, INQUIRE_ELEMENT_LENGTH function .............. 1104
SMMC gate, INQUIRE_TASK_STORAGE function .............. 1104
SMMC gate, INITIALIZE function .............. 1105
SMMC gate, GETMAIN function .............. 1105
SMMC gate, FREEMAIN function .............. 1106
SMMC gate, FREEMAIN_ALL_TERMINAL function .............. 1107
SMSR gate, INQUIRE_ISOLATION_TOKEN function .............. 1108
SMSR gate, INQUIRE_REENTRANT_PROGRAM function .............. 1108
SMSR gate, INQUIRE_SHORT_ON_STORAGE function .............. 1109
SMSR gate, SET_STORAGE_RECOVERY function .............. 1110
SMSR gate, SET_TRANSACTION_ISOLATION function .............. 1110
SMSR gate, SWITCH_SUBSPACE function .............. 1110
SMSR gate, INQUIRE_DSA_SIZE function .............. 1111
SMSR gate, SET_DSA_LIMIT function .............. 1112
SMSR gate, SET_STORAGE_PROTECT function .............. 1112
SMSR gate, INQUIRE_STORAGE_PROTECT function .............. 1113
SMSR gate, SET_REENTRANT_PROGRAM function .............. 1114
SMSR gate, UPDATE_SUBSPACE_TCB_INFO function .............. 1114
Input parameters .................................. 1114
Output parameters .................................. 1115
Storage manager domain’s generic gates .............. 1115
Storage manager domain’s generic format .............. 1116
Format SMNT, STORAGE_NOTIFY function .............. 1117
Modules .................................. 1117
Exits .................................. 1118
Trace .................................. 1118

Contents xix
Chapter 92. Terminal abnormal condition program .................................. 1191

Chapter 93. Terminal control ............................................................. 1195

Chapter 94. Terminal error program .................................................. 1223

Chapter 95. Timer domain (TI) ......................................................... 1225

Chapter 96. Trace control macro-compatibility interface ...................... 1231

Chapter 97. Trace domain (TR) ...................................................... 1233

Contents  xxi
Chapter 103. Transient data control 1335
Design overview .......................... 1335
Intrapartition queues .......................... 1335
Extrapartition queues .......................... 1336
Indirect queues .......................... 1336
Automatic transaction initiation .......................... 1336
Transient data services .......................... 1337
Transient data .......................... 1337
Modules .................................. 1340
Exits .................................. 1340
Trace .................................. 1340

Chapter 104. User domain 1341
User domain’s specific gates .......................... 1341
USAD gate, ADD_USER_WITH_PASSWORD function .......................... 1342
USAD gate, ADD_USER_WITHOUT_PASSWORD function .......................... 1344
USAD gate, DELETE_USER function .......................... 1345
USAD gate, INQUIRE_USER function .......................... 1346
USAD gate, INQUIRE_DEFAULT_USER function .......................... 1347
USAD gate, VALIDATE_USERID function .......................... 1349
USFL gate, FLATTEN_USER function .......................... 1349
USFL gate, TAKEOVER function .......................... 1350
USFL gate, UNFLATTEN_USER function .......................... 1350
USIS gate, SET_USER_DOMAIN_PARMS function .......................... 1351
USXM gate, ADD_TRANSACTION_USER function .......................... 1351
USXM gate, DELETE_TRANSACTION_USER function .......................... 1352
USXM gate, END_TRANSACTION function .......................... 1352
USXM gate, FLATTEN_TRANSACTION_USER function .......................... 1353
USXM gate, INIT_TRANSACTION_USER function .......................... 1353
USXM gate, INQUIRE_TRANSACTION_USER function .......................... 1354
USXM gate, TERM_TRANSACTION_USER function .......................... 1355
USXM gate, UNFLATTEN_TRANSACTION_USER function .......................... 1356
User domain’s generic gates .......................... 1357
Modules .................................. 1358
Exits .................................. 1358
Trace .................................. 1340

Chapter 105. User exit control. 1359
Design overview .......................... 1359
User exit control modules .......................... 1361
Control blocks .......................... 1362
Modules .................................. 1364
Exits .................................. 1364
Trace .................................. 1364

Chapter 106. VTAM generic resource 1365
Design Overview .......................... 1365
Generic resource and LU6.1/LU6.2 .......................... 1365
LU6.2 GR to GR connections .......................... 1365
LU6.2 GR to non-GR connections .......................... 1366
LU6.1 .......................... 1367
Ending affinities .................................. 1367
Generic resource and ATI .......................... 1367
Modules .................................. 1368
DFHZBLX .................................. 1368
DFHZGCH .......................... 1368
DFHZGIN .................................. 1369
Problem solving for generic resource .......................... 1369
Generic resource status byte (TCTV_GRSTATUS) .......................... 1370
Generic resource flag byte (TCSEI_GR) .......................... 1370
Trace .................................. 1371
Waits .................................. 1371

Chapter 107. VTAM LU6.2 1373
Design overview .......................... 1373
Session management .......................... 1373
LU6.2 session states .......................... 1376
LU6.2 SEND and RECEIVE processing .......................... 1376
Limited resources .......................... 1377
Modules .................................. 1377
DFHZRVL .................................. 1378
DFHZRLP .................................. 1378
DFHZSDL .................................. 1379
DFHZSLX .................................. 1381
DFHZRLX .................................. 1381
DFHCLS3 .................................. 1381
DFHZLXS .................................. 1381
DFHZGCN .................................. 1382
DFHZGCA .................................. 1383
Exits .................................. 1384
Trace .................................. 1384

Chapter 108. VTAM persistent sessions 1385
Design overview .......................... 1385
Persistent Sessions Restart flow .......................... 1386
Modules .................................. 1391
Diagnosing Persistent Sessions Problems .......................... 1393
Persistent Sessions status byte (TCTE_PRSS) .......................... 1396
Bid status byte (TCTE_BID_STATUS) .......................... 1398
Summary of persistent session waits .......................... 1399
VTAM exits .................................. 1399
Trace .................................. 1400
Statistics .................................. 1400
Preface

What this book is about

When the term “CICS” is used without any qualification in this book, it refers to the CICS element of IBM CICS Transaction Server for OS/390.

“MVS” is used for the operating system, which can be either an element of OS/390, or MVS/Enterprise System Architecture System Product (MVS/ESA SP).

This book gives a detailed description of the various components that make up a CICS system. It also provides reference tables of CICS source modules and executable modules.

This book is intended to help you in diagnosing problems with CICS.

Who this book is for

This book provides a basis for communication between the system programmer and the IBM support representative whenever a problem with CICS code is suspected.

What you need to know to use this book

You should have system programming experience and a good working knowledge of CICS and of the functions used in your system to support CICS applications.

Before using this book, you should have read the CICS Problem Determination Guide to learn about the general approach to CICS problem-solving and the procedures to use when diagnosing and reporting system problems. You should already be familiar with the general layout of CICS traces and dumps.

In addition, you may need to refer to the following books in the CICS library while diagnosing what appears to be a system problem:

- The CICS Data Areas manual for details of the layout and contents of CICS data areas
- The CICS Messages and Codes manual for information about the messages and abend codes that can be issued by a running CICS system

Notes on terminology

The following abbreviations are used throughout this book:

<table>
<thead>
<tr>
<th>Term</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>CICS</td>
<td>When used without qualification in the book, refers to the CICS element of IBM CICS Transaction Server for z/OS</td>
</tr>
<tr>
<td>ESA</td>
<td>IBM Enterprise Systems Architecture/370 (ESA/370)</td>
</tr>
<tr>
<td>MVS</td>
<td>The IBM operating system, which can be either an element of OS/390, or MVS/Enterprise System Architecture System Product (MVS/ESA SP)</td>
</tr>
</tbody>
</table>
VTAM
IBM Advanced Communications Function/Virtual Telecommunications Access Method (ACF/VTAM)

VTAM/NCP
IBM Virtual Telecommunications Access Method/Network Control Program (VTAM/NCP)

TCAM
The DCB interface of the IBM Advanced Communications Function/Telecommunications Access Method (ACF/TCAM)

IMS
IMS/ESA

DL/I
The DL/I facilities of IMS/ESA

FEPI
Front End Programming Interface

Book structure

The structure of this book is as follows:

"Part 1. Introduction" on page 1
this gives an introduction to this book and to the structure of CICS.

"Part 2. CICS components" on page 11
this describes the various CICS components (domains and functions).

"Part 3. CICS modules" on page 1417
this lists alphabetically the contents of the CICS distribution material, gives tables showing the link-edit relationships between object modules and load modules in a CICS system, and also describes briefly many of the CICS executable modules.
Summary of changes

This edition of the CICS Diagnosis Reference is based on the CICS Transaction Server for OS/390 Version 1 Release 3 edition. Changes from the CICS Transaction Server for OS/390 Version 1 Release 3 edition are marked by a vertical line to the left of the change.

Changes for this CICS Transaction Server for z/OS, Version 2 Release 1 edition

The main changes for this edition are as follows:

- Trace entries have been moved to the CICS Trace Entries by popular demand.
- New gates have been added to the following domains to support the 3270 bridge:
  - Application domain, starting on page 21
  - Transaction Manager domain, starting on page 1261

Changes for the CICS Transaction Server for OS/390 Version 1 Release 3 edition

The main changes for the previous edition are as follows:

- New chapters on the new domain components of CICS were added:
  - Log manager domain, starting on page 778
  - VTAM generic resources, starting on page 1365
  - Recovery manager domain, starting on page 919
- The file control chapter has been amended to show new function within file control.
- The following chapters have been deleted:
  - Asynchronous processing.
  - Dynamic backout programming.
  - Emergency restart.
  - Journaling chapters and system log/journaling utilities - this is largely replaced by the Log manager domain.
  - Local DL/I.
  - Shared databases.
  - Time of day control.
  - Volume control.

Changes for the CICS/ESA 4.1 edition

The book was enhanced to reflect the changes to the CICS product for CICS/ESA 4.1.

A new part that provides the CICS trace interpretation tables, was moved from the CICS Trace Entries.

New chapters on the new domain components of CICS were added:

- Directory manager domain, starting on page 292
The functions of the terminal allocation program (DFHALP) were replaced by calls to the TFAL gate. For information about the TFAL gate functions, see page 75.

A new chapter about the Kernel sub-component of the application domain, "Chapter 6. AP domain KC subcomponent" on page 107 was added.

The following sections from "Chapter 41. Front end programming interface (FEPI)" on page 657 were moved to the CICS Problem Determination Guide:
- Problem determination
- FEPI waits
- FEPI abends
- Reporting a FEPI problem to IBM

Changes for the CICS/ESA 3.3 edition

This book has been enhanced to reflect the changes to the CICS product for CICS/ESA 3.3.

There is a new section on the distributed program link component of CICS, starting on page 327.

The section on the system recovery component of CICS have been extensively rewritten while including details of the changes to this component in CICS Transaction Server for z/OS, Version 2 Release 1.
Part 1. Introduction

This book describes the functional areas (or components) into which CICS is divided. To understand more about a particular functional area, use the contents list and the index to find the appropriate information.

If you are using this book to diagnose a system problem, to find out whether a function is working as designed, you should also consult the special topic, administration, or programming books in the CICS Transaction Server for z/OS 5.1 library listed at the front of this book.

In this and other CICS books, the word “component” is used in a general way to refer to any unit of code that performs an identifiable set of functions and manages a certain type of data.

Some CICS components are shipped as object code only (OCO). If the component causing a problem is OCO, it is the responsibility of IBM to diagnose the problem further. If the component is not OCO, you can refer to the source code on microfiche, and use the detailed information in this book to identify more specifically the cause of the problem. The “Chapter 112. CICS directory” on page 1419 shows which CICS object modules are regarded as OCO; no source code is available for these modules.
Chapter 1. CICS domains

At the top level, CICS is organized into domains as shown in Figure 1 on page 4. With the exception of the application domain (more about this later), each domain is a single major component of CICS.
Domains never communicate directly with each other. Calls between domains are routed through kernel linkage routines. Calls can be made only to official interfaces to the domains, and they must use the correct protocols.

Figure 1. CICS organization—domains. The domain identifiers are shown in parentheses.
Each domain manages its own data. No domain accesses another domain’s data directly. If a domain needs data belonging to another domain, it must call that domain, and that domain then passes the data back in the caller’s parameter area.

The following table lists the CICS domains alphabetically by domain identifier. For each domain, the table also shows whether or not the domain is OCO, and gives a page reference to the section describing the interfaces to the domain.

<table>
<thead>
<tr>
<th>Domain ID</th>
<th>Domain</th>
<th>OCO?</th>
<th>See page</th>
</tr>
</thead>
<tbody>
<tr>
<td>AP</td>
<td>Application</td>
<td>See note</td>
<td>21</td>
</tr>
<tr>
<td>BAM</td>
<td>Business Application Manager</td>
<td>Yes</td>
<td>206</td>
</tr>
<tr>
<td>DD</td>
<td>Directory manager</td>
<td>Yes</td>
<td>205</td>
</tr>
<tr>
<td>DH</td>
<td>Document handler</td>
<td>Yes</td>
<td>204</td>
</tr>
<tr>
<td>DM</td>
<td>Domain manager</td>
<td>Yes</td>
<td>203</td>
</tr>
<tr>
<td>DS</td>
<td>Dispatcher</td>
<td>Yes</td>
<td>202</td>
</tr>
<tr>
<td>DU</td>
<td>Dump</td>
<td>No</td>
<td>201</td>
</tr>
<tr>
<td>EJ</td>
<td>Enterprise Java</td>
<td>No</td>
<td>195</td>
</tr>
<tr>
<td>EM</td>
<td>Event manager</td>
<td>Yes</td>
<td>194</td>
</tr>
<tr>
<td>EX</td>
<td>External CICS interface</td>
<td>Yes</td>
<td>193</td>
</tr>
<tr>
<td>GC</td>
<td>Global catalog</td>
<td>Yes</td>
<td>192</td>
</tr>
<tr>
<td>II</td>
<td>IIOP</td>
<td>No</td>
<td>191</td>
</tr>
<tr>
<td>KE</td>
<td>Kernel</td>
<td>Yes</td>
<td>190</td>
</tr>
<tr>
<td>LC</td>
<td>Local catalog</td>
<td>Yes</td>
<td>189</td>
</tr>
<tr>
<td>LD</td>
<td>Loader</td>
<td>Yes</td>
<td>188</td>
</tr>
<tr>
<td>LG</td>
<td>Log manager</td>
<td>Yes</td>
<td>187</td>
</tr>
<tr>
<td>LM</td>
<td>Lock manager</td>
<td>Yes</td>
<td>186</td>
</tr>
<tr>
<td>ME</td>
<td>Message</td>
<td>Yes</td>
<td>185</td>
</tr>
<tr>
<td>MN</td>
<td>Monitoring</td>
<td>Yes</td>
<td>184</td>
</tr>
<tr>
<td>NQ</td>
<td>Enqueue</td>
<td>Yes</td>
<td>183</td>
</tr>
<tr>
<td>OT</td>
<td>Object Transmission</td>
<td>No</td>
<td>182</td>
</tr>
<tr>
<td>PA</td>
<td>Parameter manager</td>
<td>Yes</td>
<td>181</td>
</tr>
<tr>
<td>PG</td>
<td>Program manager</td>
<td>Yes</td>
<td>180</td>
</tr>
<tr>
<td>RM</td>
<td>Recovery manager</td>
<td>Yes</td>
<td>179</td>
</tr>
<tr>
<td>RX</td>
<td>Resource recovery service</td>
<td>Yes</td>
<td>178</td>
</tr>
<tr>
<td>RZ</td>
<td>Request Stream</td>
<td>No</td>
<td>177</td>
</tr>
<tr>
<td>SH</td>
<td>Scheduler services</td>
<td>Yes</td>
<td>176</td>
</tr>
<tr>
<td>SJ</td>
<td>JVM Domain</td>
<td>No</td>
<td>175</td>
</tr>
<tr>
<td>SM</td>
<td>Storage manager</td>
<td>Yes</td>
<td>174</td>
</tr>
<tr>
<td>SO</td>
<td>Sockets Domain</td>
<td>No</td>
<td>173</td>
</tr>
<tr>
<td>ST</td>
<td>Statistics</td>
<td>Yes</td>
<td>172</td>
</tr>
<tr>
<td>TD</td>
<td>Transient data</td>
<td>Yes</td>
<td>171</td>
</tr>
<tr>
<td>TI</td>
<td>Timer</td>
<td>Yes</td>
<td>170</td>
</tr>
<tr>
<td>TR</td>
<td>Trace</td>
<td>No</td>
<td>169</td>
</tr>
<tr>
<td>TS</td>
<td>Temporary storage</td>
<td>Yes</td>
<td>168</td>
</tr>
<tr>
<td>US</td>
<td>User</td>
<td>Yes</td>
<td>167</td>
</tr>
<tr>
<td>WB</td>
<td>Web Domain</td>
<td>No</td>
<td>166</td>
</tr>
<tr>
<td>XM</td>
<td>Transaction manager</td>
<td>Yes</td>
<td>165</td>
</tr>
<tr>
<td>XS</td>
<td>Security manager</td>
<td>Yes</td>
<td>164</td>
</tr>
</tbody>
</table>

Note: The application domain is mainly non-OCO, but it contains these OCO components:
- CICS data table services
- RDO for VSAM files and LSR pools
- Some EXEC CICS system programming functions
- Autoinstall terminal model manager
• Partner resource manager
• SAA Communications and Resource Recovery
• Some of the file control functions
• Recovery manager connectors interfaces.

The offline statistics utility program (DFHSTUP) and the system dump formatting routines are also treated as OCO.

**Domain gates**

A domain gate is an entry point or interface to a domain. It can be called by any authorized caller who needs to use some function provided by the domain.

A number of domain functions are available through the exit programming interface (XPI). For details, see the [CICS Customization Guide](#).

In practice, every domain has several gates. Each gate has a 4-character identifier; the first two characters are the identifier of the owning domain, and the second two characters differentiate between the functions of the domain’s gates. Here, for example, are two of the dispatcher (DS) domain’s gates:

```
DSAT
DSSR
```

**Functions provided by gates**

An individual gate can provide many functions. The required function is determined by the parameters included on the call. The DSSR gate of the DS domain, for example, provides all these functions:

- `ADD_SUSPEND`
- `DELETE_SUSPEND`
- `INQUIRE_SUSPEND_TOKEN`
- `RESUME`
- `SUSPEND`
- `WAIT_MVS`
- `WAIT_OLDC`
- `WAIT_OLDW`.

**Specific gates, generic and call-back gates**

It is useful to distinguish between **specific gates**, **generic gates** and **callback gates**:

- A specific gate gives access to a set of functions that are provided by that domain alone. The functions are likely to be requested by many different callers. DS domain, for example, has a specific gate (DSAT) that provides CHANGE_MODE and CHANGE_PRIORITY functions (among other functions). Only the DS domain provides those functions, but they can be requested by many different callers.
- A generic gate gives access to a set of functions that are provided by several domains.

Most domains provide a QUIESCE_DOMAIN function, for example, so that they can be quiesced when CICS is shutting down normally. They each have a generic gate that provides this function. DM domain makes a **generic call** to that gate in any domain that is to be quiesced.
A call-back gate also gives access to a set of functions that can be provided by several domains. Unlike a generic gate where the call is broadcast to all domains that have provided a gate a call-back is restricted to specific domains but uses a format owned by the calling domain.

For example the Recovery Manager calls the domains that have registered an interest in syncpoint processing using the PERFORM_PREPARE function format that it owns.

**Domain call formats**

Any module calling a domain gate must use the correct format for the call. The format represents the parameter list structure. It describes the parameters that must be provided on the call (the input parameters), and the parameters that are returned to the caller when the request has been processed (the output parameters).

For example, Table 1 lists the input and output parameters for the ATTACH function of the DS domain’s DSAT gate.

<table>
<thead>
<tr>
<th>Input parameters</th>
<th>Output parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRIORITY</td>
<td>TASK_TOKEN</td>
</tr>
<tr>
<td>USER_TOKEN</td>
<td>RESPONSE</td>
</tr>
<tr>
<td>[TIMEOUT]</td>
<td>[REASON]</td>
</tr>
<tr>
<td>TYPE</td>
<td></td>
</tr>
<tr>
<td>[MODE]</td>
<td></td>
</tr>
<tr>
<td>[TASK_REPLY_GATE_INDEX]</td>
<td></td>
</tr>
<tr>
<td>[SPECIAL_TYPE]</td>
<td></td>
</tr>
</tbody>
</table>

Parameters not shown in brackets are mandatory, and are always interpreted in trace entries. Parameters shown in brackets are optional, and are in trace entries only if values have been set. An exception to this rule is that, regardless of whether REASON is mandatory or optional for a particular function, its value is included in a trace entry only for a non-'OK' response.

The domain call formats described are in the sections dealing with the domains that own them, as discussed in "Ownership of formats".

**Ownership of formats**

Every format is ‘owned’ by a domain:

- The formats for specific calls are owned by the domain being called. DS domain, for example, owns the format for the CHANGE_MODE and CHANGE_PRIORITY calls. This book uses the term specific format to refer to such formats.

- The formats for generic calls and call-back calls are owned by the calling domain. DM domain, for example, owns the format for calls to (generic) gates providing the QUIESCE_DOMAIN function in other domains. This book uses the term generic format to refer to such formats.
Tokens

Tokens are passed as parameters on many domain calls. They identify uniquely objects that are operands of domain functions.

Here are some examples:

**TASK_TOKEN**
uniquely identifies a task to be used as the operand of a function.

**DOMAIN_TOKEN**
uniquely identifies a domain to be used as the operand of a function.

**SUSPEND_TOKEN**
uniquely identifies a task for the purpose of a suspend or resume dialog.

Responses

On all domain calls, one of the output parameters is the domain’s response to the call. It can have any one of these values:

**OK**
When a domain call succeeds, a response of ‘OK’ is given and the REASON code is not set. The requested function has been completed successfully.

**EXCEPTION**
Processing of the function could not be completed for the reason specified in the REASON field. The domain state remains unchanged if such an error occurs.

**DISASTER**
The domain could not complete the request because of some irrecoverable system problem. If there is a major error in the domain, this is reported.

**INVALID**
The parameter list is not valid. If a call is used incorrectly, this is reported.

**KERNERROR**
The kernel was unable to call the required function gate.

**PURGED**
A purge has been requested for the task making the domain call.
Chapter 2. Application domain

Application programs are run in the application (AP) domain, which contains several major components, as shown in Figure 2 on page 10.

Most application domain CICS functions are either provided by modules that are part of the CICS nucleus, that is to say they are an integral part of the system and are loaded at system initialization time, or they are system application programs, which are loaded as needed in the same way as user application programs.
Figure 2. AP domain—major components
Part 2. CICS components

This part describes the various components of a CICS system—the domains into which CICS is organized, and the functions within these domains. Offline utilities, such as the statistics utility program, are also covered.

Apart from the application domain and the two catalog domains, each domain has one section describing it. The application domain consists of so many components that each component is described in a separate section, except for the two catalog domains that are described in the same section.

Sections are ordered alphabetically by component name for quick reference.
Chapter 3. Added gates domain (AP)

The APTC interface in AP domain is called by the transport layer in the RZ domain - requeststreams. It issues the following calls:

- OPEN
- SEND
- RECEIVE
- CLOSE
- ISSUE_NOTIFY
- LISTEN
- CANCEL
- SET_SESSION
- EXTRACT_PROCESS

The APID interface inquires on data in the AP domain. It issues the following calls:

- PROFILE

AP domain’s new gates

Table 2 summarizes the AP domain’s new specific gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and whether or not the functions are available through the exit programming interface (XPI).

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>XPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>APTC</td>
<td>AP 4900</td>
<td>OPEN</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>AP 4901</td>
<td>SEND</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RECEIVE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CLOSE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ISSUE_NOTIFY</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LISTEN</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CANCEL</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SET_SESSION</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EXTRACT_PROCESS</td>
<td>NO</td>
</tr>
<tr>
<td>APID</td>
<td>AP 092A</td>
<td>PROFILE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>AP 092B</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

APTC gate, OPEN function

The OPEN function of the APTC gate is used to allocate a session to the specified AOR.

Input parameters

- **sysid**: Name of the AOR.
- **tranid**: Transaction name to be attached in the AOR.
- **netname**: The netname / applid of the AOR.
- **queue**: The queue option specified by the routing program.
Added gates domain (AP)

**Output parameters**

- **error code**
  code passed back from allocate procedure.

- **RESPONSE**
  is the domain’s response to the call. It can have any of these values:
  OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

- **[REASON]**
  is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID.
  Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>OPEN_ERROR</td>
</tr>
</tbody>
</table>

**APTC gate, SEND function**

The SEND function of the APTC gate is used to send the request to the AOR.

**Input parameters**

- **token**
  Token for the session TCTTE.

- **send_block**
  Block data with length and send data pointer.

- **prefix_area**
  Requeststreams information

- **last**
  An indicator to say if whether this is the last flow.

**Output parameters**

- **RESPONSE**
  is the domain’s response to the call. It can have any of these values:
  OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

- **[REASON]**
  is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID.
  Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>TOKEN_UNKNOWN</td>
</tr>
</tbody>
</table>

**APTC gate, RECEIVE function**

The RECEIVE function of the APTC gate is used to receive data.

**Input parameters**

- **token**
  Token for the session TCTTE.

- **receive_buffer**
  A buffer into which the reply is to be placed.

**Output parameters**

- **last**
  An indicator to say if whether this is the last flow.
RESPONSE
is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID.
Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>TOKEN_UNKNOWN RECEIVE_BUFFER_TOO_SMALL TC_ERROR NO_TCTTE</td>
</tr>
</tbody>
</table>

**APTC gate, CLOSE function**
The CLOSE function of the APTC gate is used in cleanup.

**Input parameters**
- **token**  Token for the session TCTTE.

**Output parameters**

RESPONSE
is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID.
Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>TOKEN_UNKNOWN TC_ERROR</td>
</tr>
</tbody>
</table>

**APTC gate, ISSUE_NOTIFY function**
The ISSUE_NOTIFY function of the APTC gate is used to inform the program DFHAPTC that a request has arrived.

**Input parameters**
- **token**  Token for the session TCTTE.
- **user_token**  Token supplied by the person who is to be notified.
- **action**  The action that should be taken.

**Output parameters**

RESPONSE
is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID.
Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>TOKEN_UNKNOWN ISSUE_NOTIFY_ERROR ACTION_UNKNOWN</td>
</tr>
</tbody>
</table>
**APTC gate, LISTEN function**

The LISTEN function of the APTC gate is used to update the TCTTE with the user token.

**Input parameters**

- **token**  
  Token for the session TCTTE.

- **user_token**  
  A token supplied by requeststreams.

**Output parameters**

- **RESPONSE**  
  is the domain’s response to the call. It can have any of these values
  
  OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

- **[REASON]**  
  is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>TOKEN_UNKNOWN TC_ERROR</td>
</tr>
</tbody>
</table>

**APTC gate, CANCEL function**

The CANCEL function invalidates the listening function.

**Input parameters**

- **token**  
  Token for the session TCTTE.

**Output parameters**

- **RESPONSE**  
  is the domain’s response to the call. It can have any of these values
  
  OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

- **[REASON]**  
  is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>TOKEN_UNKNOWN TC_ERROR</td>
</tr>
</tbody>
</table>

**APTC gate, SET_SESSION function**

The SET_SESSION function of the APTC gate is used to set the recovery status of the session TCTTE.

**Input parameters**

- **token**  
  Token for the session TCTTE.

- **recovery_status**  
  Indicates if recovery is necessary.
**Output parameters**

**RESPONSE**

is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**[REASON]**

is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>TOKEN_UNKNOWN TC_ERROR</td>
</tr>
</tbody>
</table>

**APTC gate, EXTRACT_PROCESS function**

The EXTRACT_PROCESS function of the APTC gate extracts information for

**Input parameters**

None  No input parameters.

**Output parameters**

**convid**

Conversation id - session tcte termid.

**process_name**

name of process to be invoked.

**synlevel**

synlevel of conversation.

**pipdata**

Not applicable

**pipdata length**

Not applicable

**RESPONSE**

is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**[REASON]**

is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>TOKEN_UNKNOWN TC_ERROR</td>
</tr>
</tbody>
</table>

**APID gate, QUERY_NETNAME function**

The QUERY_NETNAME function of the APID gate extracts from the AP domain the netname for a given sysid.

**Input parameters**

**sysid**  Name of sysid.
Added gates domain (AP)

**Output parameters**

- **netname**
  Value of netname for the given sysid.

**RESPONSE**

- is the domain’s response to the call. It can have any of these values
  OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**[REASON]**

- is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID.
  Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>NOT_FOUND TM_LOCATE_FAILED</td>
</tr>
</tbody>
</table>

**APID gate, PROFILE function**

The PROFILE function of the APID gate extracts from the AP domain profile information for timeout.

**Input parameters**

- **name**
  Name of profile.

**Output parameters**

- **rtimeout**
  Read timeout value.

**RESPONSE**

- is the domain’s response to the call. It can have any of these values
  OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**[REASON]**

- is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID.
  Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>NOT_FOUND TM_LOCATE_FAILED</td>
</tr>
</tbody>
</table>

**Modules**

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHAPTC</td>
<td>AP domain module DFHAPTC provides a common mechanism for issuing TC macro calls. code.</td>
</tr>
<tr>
<td></td>
<td>OPEN</td>
</tr>
<tr>
<td></td>
<td>SEND</td>
</tr>
<tr>
<td></td>
<td>RECEIVE</td>
</tr>
<tr>
<td></td>
<td>CLOSE</td>
</tr>
<tr>
<td></td>
<td>ISSUE_NOTIFY</td>
</tr>
<tr>
<td></td>
<td>CANCEL</td>
</tr>
<tr>
<td></td>
<td>SET_SESSION</td>
</tr>
<tr>
<td></td>
<td>EXTRACT_PROCESS</td>
</tr>
<tr>
<td>Module</td>
<td>Function</td>
</tr>
<tr>
<td>---------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>DFHAPID</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>PROFILE</td>
</tr>
</tbody>
</table>

**Exits**

NONE

**Trace**

The point IDs have just been added onto the usual AP ones.
Chapter 4. Application domain (AP)

The principal components of the application domain are described under "Chapter 2. Application domain" on page 9.

Application domain’s specific gates

Table 3 summarizes the application domain’s specific gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and whether or not the functions are available through the exit programming interface (XPI).

Table 3. Application domain’s specific gates

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>XPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABAB</td>
<td>AP 0741</td>
<td>CREATE_ABEND_RECORD</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>AP 0742</td>
<td>UPDATE_ABEND_RECORD</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>START_ABEND</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INQUIRE_ABEND_RECORD</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TAKE_TRANSACTION_DUMP</td>
<td></td>
</tr>
<tr>
<td>APAP</td>
<td>AP 0910</td>
<td>TRANSFER_SIT</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>AP 0911</td>
<td></td>
<td></td>
</tr>
<tr>
<td>APEX</td>
<td>AP 0510</td>
<td>INVOKE_USER_EXIT</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>AP 0515</td>
<td></td>
<td></td>
</tr>
<tr>
<td>APID</td>
<td>AP 092A</td>
<td>PROFILE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>AP 092B</td>
<td>QUERY_NETNAME</td>
<td></td>
</tr>
<tr>
<td>APIQ</td>
<td>AP 0920</td>
<td>INQ_APPLICATION_DATA</td>
<td>YES</td>
</tr>
<tr>
<td></td>
<td>AP 0921</td>
<td></td>
<td></td>
</tr>
<tr>
<td>APJC</td>
<td>AP F900</td>
<td>WRITE_JOURNAL_DATA</td>
<td>YES</td>
</tr>
<tr>
<td></td>
<td>AP F901</td>
<td></td>
<td></td>
</tr>
<tr>
<td>APLH</td>
<td>AP 19A0</td>
<td>ESTABLISH_LANGUAGE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>AP 19A1</td>
<td>NOTIFY_REFRESH</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>START_PROGRAM</td>
<td>NO</td>
</tr>
<tr>
<td>APLI</td>
<td>AP 1940</td>
<td>ESTABLISH_LANGUAGE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>AP 1941</td>
<td>START_PROGRAM</td>
<td>NO</td>
</tr>
<tr>
<td>APLJ</td>
<td>AP 1960</td>
<td>ESTABLISH_LANGUAGE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>AP 1961</td>
<td>START_PROGRAM</td>
<td>NO</td>
</tr>
<tr>
<td>APRM</td>
<td>AP 05A0</td>
<td>TRANSACTION_INITIALIZATION</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>AP 05A1</td>
<td>TRANSACTION_TERMINATION</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INQUIRE</td>
<td>NO</td>
</tr>
<tr>
<td>APRT</td>
<td>AP 1900</td>
<td>ROUTE_TRANSACTION</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>AP 1901</td>
<td></td>
<td></td>
</tr>
<tr>
<td>APTC</td>
<td>AP 4900</td>
<td>CANCEL</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>AP 4901</td>
<td>CLOSE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EXTRACT_PROCESS</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ISSUE_NOTIFY</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LISTEN</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OPEN</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RECEIVE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SEND</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SET_SESSION</td>
<td>NO</td>
</tr>
</tbody>
</table>
## Application domain (AP)

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>XPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>APTD</td>
<td>AP F600</td>
<td>WRITE_TRANSIENT_DATA</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>AP F601</td>
<td>READ_TRANSIENT_DATA</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>DELETE_TRANSIENT_DATA</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>INITIALIZE_TRANSIENT_DATA</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>RESET_TRIGGER_LEVEL</td>
<td></td>
</tr>
<tr>
<td>AP XM</td>
<td>AP 0590</td>
<td>TRANSACTION_INITIALIZATION</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>AP 0591</td>
<td>RMI_START_OF_TASK</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TRANSACTION_TERMINATION</td>
<td>NO</td>
</tr>
<tr>
<td>BRAT</td>
<td>AP 2800</td>
<td>ATTACH</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>AP 2801</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BRFM</td>
<td>AP 2140</td>
<td>Subroutine for bridge facility allocation/deletion.</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>AP 2141</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BRIC</td>
<td>AP 2166</td>
<td>Subroutine interfacing interval control EXEC commands and the bridge exit.</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>AP 2167</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BRIQ</td>
<td>AP 2820</td>
<td>INQUIRE_CONTEXT</td>
<td>NO</td>
</tr>
<tr>
<td>BRMS</td>
<td>AP 2160</td>
<td>Subroutine interfacing BMS EXEC commands and the bridge exit.</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>AP 2161</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BRRM</td>
<td>AP 2840</td>
<td>RMRO callback for PREPARE and COMMIT</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>AP 2841</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BRSP</td>
<td>AP 216C</td>
<td>Subroutine interfacing syncpoint requests and the bridge exit.</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>AP 216D</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BRTC</td>
<td>AP 2163</td>
<td>Subroutine interfacing terminal control EXEC commands and the bridge exit.</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>AP 2164</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BRXM</td>
<td>AP 2860</td>
<td>XMAC callback for INIT_XM_CLIENT and BIND_XM_CLIENT</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>AP 2861</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ICXM</td>
<td>AP 05C0</td>
<td>BIND_FACILITY, RELEASE_FACILITY, INQUIRE_FACILITY</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>AP 05C1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RTSU</td>
<td>AP 1910</td>
<td>COMMIT_SURROGATE, FREE_SURROGATE, GET_RECOVERY_STATUS, PREPARE_SURROGATE, RESET_SURROGATE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>AP 1911</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TDOC</td>
<td>AP F640</td>
<td>OPEN_TRANSIENT_DATA</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>AP F641</td>
<td>CLOSE_TRANSIENT_DATA</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CLOSE_ALL_EXTRA_TD_QUEUES</td>
<td></td>
</tr>
<tr>
<td>TDM</td>
<td>AP F680</td>
<td>ADD_REPLACE_TDQDEF</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>AP F681</td>
<td>INQUIRE_TDQDEF</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>START_BROWSE_TDQDEF</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GET_NEXT_TDQDEF</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>END_BROWSE_TDQDEF</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SET_TDQDEF</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DISCARD_TDQDEF</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COMMIT_TDQDEFS</td>
<td>NO</td>
</tr>
<tr>
<td>TDXM</td>
<td>AP 05B0</td>
<td>BIND_FACILITY</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>AP 05B1</td>
<td>BIND_SECONDARY_FACILITY</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RELEASE_FACILITY</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INQUIRE_FACILITY</td>
<td>NO</td>
</tr>
<tr>
<td>SAIQ</td>
<td>AP E120</td>
<td>INQUIRE_SYSTEM</td>
<td>YES</td>
</tr>
<tr>
<td></td>
<td>AP E122</td>
<td>SET_SYSTEM</td>
<td>YES</td>
</tr>
</tbody>
</table>
Table 3. Application domain’s specific gates (continued)

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>XPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>TFAL</td>
<td>AP D600</td>
<td>ALLOCATE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>AP D601</td>
<td>CANCEL_AID</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CHECK_TRANID_IN_USE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CANCEL_AIDS_FOR_CONNECTION</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CANCEL_AIDS_FOR_TERMINAL</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DISCARD_AIDS</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FIND TRANSACTION_OWNER</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GET MESSAGE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INITIALIZE_AID_Pointers</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INQUIRE_ALLOCATE_AID</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LOCATE_AID</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LOCATE_REMDEL_AID</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LOCATE_SHIPPABLE_AID</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MATCH_TASK_TO_AID</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PURGE_ALLOCATE_AIDS</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RECOVER_START_DATA</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>REMOVE_EXPIRED_AID</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>REMOVE_EXPIRED_REMOTE_AID</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>REMOVE_MESSAGE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>REMOVE_REMOTE_DELETES</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>REROUTE_SHIPPABLE_AIDS</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RESCHEDULE_BMS</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RESET_AID_QUEUE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RESTORE_FROM_KEYPOINT</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RETRIEVE_START_DATA</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SCHEDULE_BMS</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SCHEDULE_START</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SCHEDULE_TDP</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SLOWDOWN_PURGE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TAKE_KEYPOINT</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TERM_AVAILABLE_FOR_QUEUE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TERMINAL_NOW_UNAVAILABLE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UNCHAIN_AID</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UPDATE_TRANNUM_FOR_RESTART</td>
<td>NO</td>
</tr>
<tr>
<td>TFBF</td>
<td>AP 1730</td>
<td>BIND_FACILITY</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>AP 1731</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TFIQ</td>
<td>AP 1700</td>
<td>INQUIRE_TERMINAL_FACILITY</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>AP 1701</td>
<td>INQUIRE_MONITOR_DATA</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SET_TERMINAL_FACILITY</td>
<td>NO</td>
</tr>
<tr>
<td>TFRF</td>
<td>AP 1710</td>
<td>RELEASE_FACILITY</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>AP 1711</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TFXM</td>
<td>AP 1790</td>
<td>INIT_XM_CLIENT</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>AP 1791</td>
<td>BIND_XM_CLIENT</td>
<td>NO</td>
</tr>
<tr>
<td>MRXM</td>
<td>AP 17B0</td>
<td>INIT_XM_CLIENT</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>AP 17B1</td>
<td>BIND_XM_CLIENT</td>
<td>NO</td>
</tr>
<tr>
<td>62XM</td>
<td>AP 17C0</td>
<td>INIT_XM_CLIENT</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>AP 17C1</td>
<td>BIND_XM_CLIENT</td>
<td>NO</td>
</tr>
</tbody>
</table>

**ABAB gate, CREATE_ABEND_RECORD function**

The CREATE_ABEND_RECORD function of the ABAB gate is used to create an abend record (TACB).
Input parameters

[ABEND_CODE] is the four-character transaction abend code.

[Failing_PROGRAM] is the name of the program in which the abend occurred.

[REQUEST_ID] is the request ID from the TCTTE for a terminal-oriented task.

[Failing_Resource] is the name of the system TCTTE (the connection) if the abend was raised by DFHZAND.

[REMOTE_SYSTEM] is the name of the remote system if the abend was raised in the client transaction to reflect an abend occurring in the DPL server.

[SENSE_BYTES] is the SNA sense bytes if the abend was raised by DFHZAND.

[ERROR_MESSAGE] is the error message sent from the remote system if the abend was raised by DFHZAND.

[EXECUTION_KEY] is a code indicating the execution key at the time the abend was issued, or at the time the operating system abend or program check occurred.

[STORAGE_TYPE] is a code indicating the storage hit on an OC4.

[ERROR_OFFSET] is the offset of a program check or operating system abend in the failing application program or CICS AP domain program.

[GENERAL_REGS] is the contents of the general purpose registers at the time of a program check or operating system abend.

[PSW] is the contents of the PSW at the time of a program check or operating system abend.

[INTERRUPT_DATA] is the interrupt code and instruction length code etc, at the time of a program check or operating system abend.

[ALET] is the access list entry token (ALET) at the time of a program check or operating system abend.

[STOKEN] is the subspace token (STOKEN) at the time of a program check or operating system abend.

[SPACE] indicates whether the task was in SUBSPACE or BASESPACE mode at the time of a program check or operating system abend. It can have any of these values: BASESPACE|SUBSPACE|NOSPACE

[GREG_ORDER] indicates the order of the registers passed in GENERAL_REGS.
DFHSRP saves the registers in the abend record in the order 0-15, and INQUIRE_ABEND_RECORD will always return them in this order. It can have either of these values:
R14|TOR13|R0|TOR15

**[ACCESS_REGISTERS]**
is the contents of the access registers at the time of a program check or operating system abend.

**[FLOATING_POINT_REGISTERS]**
is the contents of the floating point registers at the time of a program check or operating system abend.

**[STATUS_FLAGS]**
is the status flags at the time of the abend.

**Output parameters**

**ABEND_TOKEN**
is the token allocated by ABAB for this abend. The token must be passed on subsequent UPDATE_ABEND_RECORD and START_ABEND requests to ABAB. The token is no longer valid after a START_ABEND request.

**RESPONSE**
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**[REASON]**
is returned when RESPONSE is DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
</tbody>
</table>

**ABAB gate, UPDATE_ABEND_RECORD function**
The UPDATE_ABEND_RECORD function of the ABAB gate is used to update an abend record (TACB).

**Input parameters**

**ABEND_TOKEN**
is the token allocated by ABAB for this abend (on a preceding CREATE_ABEND_RECORD request). The token must be passed on subsequent UPDATE_ABEND_RECORD and START_ABEND requests to ABAB. The token is no longer valid after a START_ABEND request.

**[ABEND_CODE]**
is the four-character transaction abend code.

**[Failing_PROGRAM]**
is the name of the program in which the abend occurred.

**[REQUEST_ID]**
is the request ID from the TCTTE for a terminal-oriented task.

**[Failing_Resource]**
is the name of the system TCTTE (the connection) if the abend was raised by DFHZAND.

**[REMOTE_SYSTEM]**
is the name of the remote system if the abend was raised in the client transaction to reflect an abend occurring in the DPL server.
Application domain (AP)

[SENSE_BYTES]
is the SNA sense bytes if the abend was raised by DFHZAND.

[ERROR_MESSAGE]
is the error message sent from the remote system if the abend was raised by DFHZAND.

[EXECUTION_KEY]
is a code indicating the execution key at the time the abend was issued, or at the time the operating system abend or program check occurred.

[STORAGE_TYPE]
is a code indicating the storage hit on an OC4.

[ERROR_OFFSET]
is the offset of a program check or operating system abend in the failing application program or CICS AP domain program.

[GENERALRegisters]
is the contents of the general purpose registers at the time of a program check or operating system abend.

[PSW]
is the contents of the PSW at the time of a program check or operating system abend.

[INTERRUPT_DATA]
is the interrupt code and instruction length code etc, at the time of a program check or operating system abend.

[ALET]
is the access list entry token (ALET) at the time of a program check or operating system abend.

[STOKEN]
is the subspace token (STOKEN) at the time of a program check or operating system abend.

[SPACE]
indicates whether the task was in SUBSPACE or BASESPACE mode at the time of a program check or operating system abend. It can have any of these values:
BASESPACE|SUBSPACE|Nospace

[GREG_ORDER]
indicates the order of the registers passed in GENERAL_registers. DFHSRP saves the registers in the abend record in the order 0-15, and INQUIRE_ABEND_RECORD will always return them in this order. It can have either of these values:
R14TOR13|ROTORM

[ACCESS_registers]
is the contents of the access registers at the time of a program check or operating system abend.

[FLOATING_POINT_registers]
is the contents of the floating point registers at the time of a program check or operating system abend.

[STATUS_FLAGS]
is the status flags at the time of the abend.
Output parameters

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

**[REASON]**

is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>INVALID_TOKEN</td>
</tr>
</tbody>
</table>

**ABAB gate, START_ABEND function**

The START_ABEND function of the ABAB gate is used to start transaction abend processing.

**Input parameters**

**ABEND_TOKEN**

is the token allocated by ABAB for this abend (on a preceding CREATE_ABEND_RECORD request).

**[DUMP]**

indicates whether a transaction dump should be produced for this abend. It can have either of these values:

YES | NO

**[IGNORE_HANDLES]**

indicates whether this abend should be passed to any EXEC CICS HANDLE routines that are active. IGNORE_HANDLES(YES) results in EXEC CICS HANDLE being ignored at all levels of the program stack. It can have either of these values:

YES | NO

**Output parameters**

**RETRY_ADDRESS**

if an XPCTA exit requests retry, control returns to the point of invocation of start_info, passing the retry address. This address includes the AMODE indicator in the first bit; it can be used as the target address in a DFHAM TYPE=BRANCH by the caller of START_ABEND. **GENERAL_REGISTERS** is also set to point to the list of registers to be used for the retry, and **SPACE** to indicate the subspace.

**[GENERAL_REGISTERS]**

is the contents of the general purpose registers at the time of a program check or operating system abend.

**[SPACE]**

indicates whether the task was in SUBSPACE or BASESPACE mode at the time of a program check or operating system abend.

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED
Application domain (AP)

[REASON]

is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>INVALID_TOKEN</td>
</tr>
</tbody>
</table>

**ABAB gate, INQUIRE_ABEND_RECORD function**

The INQUIRE_ABEND_RECORD function of the ABAB gate is used to inquire about an abend record (TACB).

**Input parameters**

[ABEND_TYPE]

indicates which abend record the information is to be extracted from. It can have any of these values:

LATEST | FIRST | LASTASRA

**Output parameters**

[ABEND_CODE]

is the four-character transaction abend code.

[DUMP]

indicates whether a dump was requested for this abend. It can have either of these values:

YES | NO

[REQUEST_ID]

is the request ID from the TCTTE for a terminal-oriented task.

[FAILING_RESOURCE]

is the name of the system TCTTE (the connection) if the abend was raised by DFHZAND.

[FAILING_PROGRAM]

is the name of the program in which the abend occurred.

[REMOTE_SYSTEM]

is the name of the remote system if the abend was raised in the client transaction to reflect an abend occurring in the DPL server.

[SENSE_BYTES]

is the SNA sense bytes if the abend was raised by DFHZAND.

[ERROR_MESSAGE]

is the error message sent from the remote system if the abend was raised by DFHZAND.

[EXECUTION_KEY]

is a code indicating the execution key at the time the abend was issued, or at the time the operating system abend or program check occurred.

[STORAGE_TYPE]

is a code indicating the storage hit on an OC4.

[ERROR_OFFSET]

is the offset of a program check or operating system abend in the failing application program or CICS AP domain program.
[GENERAL_REGISTERS]  
is the contents of the general purpose registers at the time of a program check or operating system abend.

[PSW]  
is the contents of the PSW at the time of a program check or operating system abend.

[INTERRUPT_DATA]  
is the interrupt code and instruction length code etc, at the time of a program check or operating system abend.

[ALET]  
is the access list entry token (ALET) at the time of a program check or operating system abend.

[STOKEN]  
is the subspace token (STOKEN) at the time of a program check or operating system abend.

[SPACE]  
indicates whether the task was in SUBSPACE or BASESPACE mode at the time of a program check or operating system abend. It can have any of these values:

BASESPACE|SUBSPACE|NOSPACE

[ACCESS_REGISTERS]  
is the contents of the access registers at the time of a program check or operating system abend.

[FLOATING_POINT_REGISTERS]  
is the contents of the floating point registers at the time of a program check or operating system abend.

[STATUS_FLAGS]  
is the status flags at the time of the abend.

[IGNORE_HANDLES]  
indicates whether this abend should be passed to any EXEC CICS HANDLE routines that are active. IGNORE_HANDLES(YES) results in EXEC CICS HANDLE being ignored at all levels of the program stack. It can have either of these values:

YES|NO

RESPONSE  
is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]  
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>NO_ABEND_RECORD</td>
</tr>
</tbody>
</table>

ABAB gate, TAKE_TRANSACTION_DUMP function  
The TAKE_TRANSACTION_DUMP function of the ABAB gate is used to take a transaction dump.
Application domain (AP)

Notes:
1. The TRANSACTION resource definition must specify dump and DUMP(YES) must be specified or defaulted on the associated START_ABEND call.
2. A transaction dump is not taken if any of the following is true:
   - The application is going to handle the abend; that is, there is an active handle at this level and IGNORE_HANDLES(NO) is specified or defaulted on the associated START_ABEND call.
   - The application is Language Environment/370 enabled, in which case the language interface deals with the abend.
   - A transaction dump is currently in progress.

Input parameters
None.

Output parameters
None.

APAP gate, TRANSFER_SIT function
The TRANSFER_SIT function of the APAP gate is used to transfer the address of DFHSIT to the AP domain after a GET_PARAMETERS call from this domain to the parameter manager domain.

Input parameters
SIT specifies the address and length of the system initialization table (DFHSIT).

Output parameters
RESPONSE is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>INVALID_ADDRESS, INCONSISTENT_RELEASE</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_SIT_LENGTH, INVALID_ADDRESS, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

APEX gate, INVOKE_USER_EXIT function
The INVOKE_USER_EXIT function of the APEX gate is used to invoke the user exit at a specified exit point.

Input parameters
EXIT_POINT is the name of the exit.

TRACE indicates whether or not user exits are to be traced. It can have either of these values:
YES|NO
[EXIT_PARAMETER_n]  
is the parameter (number \( n \)) required by the exit. The nature of the parameter varies from one exit to another.

**Output parameters**

**EXIT_RETURN_CODE**  
is the return code, if any, issued by the exit.

**RESPONSE**  
is the domain’s response to the call. It can have any of these values:

\[
\text{OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED}
\]

**[REASON]**  
is returned when RESPONSE is EXCEPTION, DISASTER, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>EXIT_PROGRAM_FAILURE</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION, INVALID_EXIT_POINT</td>
</tr>
</tbody>
</table>

**APID gate, PROFILE function**

The PROFILE function of the APID gate extracts information from the AP domain profile for timeout.

**Input parameters**

**NAME**  
is the name of the profile

**Output parameters**

**RTIMEOUT**  
is the read timeout value.

**RESPONSE**  
is the domain’s response to the call. It can have any of these values:

\[
\text{OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED}
\]

**[REASON]**  
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>NOT_FOUND, TM_LOCATE_FAILED</td>
</tr>
</tbody>
</table>

**APID gate, QUERY_NETNAME function**

The PROFILE function of the APID gate extracts information from the AP domain profile for timeout.

**Input parameters**

**SYSID**  
is the name of the sysid
Application domain (AP)

Output parameters
NETNAME
is the value of the netname for the given sysid.

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>NOT_FOUND, TM_LOCATE_FAILED</td>
</tr>
</tbody>
</table>

APIQ gate, INQ_APPLICATION_DATA function

The INQ_APPLICATION_DATA function of the APIQ gate is used to inquire about application data owned by the application domain.

Input parameters
None.

Output parameters
EIB    is the address of the EXEC Interface Block.
SYSEIB is the address of the System EXEC Interface Block.
TCTUA  is the address of the Task Control Table User Area.
TCTUASIZE is the length (in bytes) of the Task Control Table User Area.
TWA    is the address of the Task Work Area.
TWASIZE is the length (in bytes) of the Task Work Area.

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, INQ_FAILED, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>DPL_PROGRAM, NO_TRANSACTION_ENVIRONMENT, TRANSACTION_DOMAIN_ERROR</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION</td>
</tr>
</tbody>
</table>
APJC gate, WRITE_JOURNAL_DATA function

The WRITE_JOURNAL_DATA function of the APJC gate is used to write a single record into a named journal.

**Input parameters**

**JOURNALNAME**
- is the journal identifier name.

**JOURNAL_RECORD_ID**
- is the system type record identifier.

**FROM**
- is the address of the record.

**[RECORD_PREFIX]**
- is the journal record user prefix.

**WAIT**
- specifies whether or not CICS is to wait until the record is written to auxiliary storage before returning control to the exit program. It can have either of these values:
  - YES
  - NO

**Output parameters**

**RESPONSE**
- is the domain’s response to the call. It can have any of these values:
  - OK
  - EXCEPTION
  - DISASTER
  - INVALID
  - KERNERROR
  - PURGED

**[REASON]**
- is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>JOURNAL_NOT_FOUND, LENGTH_ERROR,</td>
</tr>
<tr>
<td></td>
<td>JOURNAL_NOT_OPEN, STATUS_ERROR, IO_ERROR</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

APLH gate, ESTABLISH_LANGUAGE function

The ESTABLISH_LANGUAGE function of the APLH gate is used to establish the language of a compiled Java program with hot-pooling.

**Input parameters**

**LOAD_POINT**
- is the load point address of the program.

**ENTRY_POINT**
- is the entry point address of the program.

**[PROGRAM_LENGTH]**
- is the length of the program.

**[DEFINED_LANGUAGE]**
- is the language defined for the program. It can have any of these values:
  - ASSEMBLER
  - C370
  - COBOL
  - LE370
  - PLI
  - NOT_DEFINED
EXECUTION_KEY

is the key in which CICS gives control to the program, and determines whether the program can modify CICS-key storage. It can have either of these values:

CICS|USER

DATA_LOCATION

defines whether the program can handle only 24-bit addresses (data located below the 16MB line) can handle 31-bit addresses (data located above or below the 16MB line). It can have either of these values:

ANY|BELOW

LANGUAGE_BLOCK

is a token identifying the current language block for the program.

Output parameters

[NEW_BLOCK]

is a new token identifying the new language block for the program.

[LANGUAGE_ESTABLISHED]

is the language established for the program. It can have any of these values:

ASSEMBLER|C370|COBOL|COBOL2|LE370|PLI|JVM|
ASSEMBLER CICS|MVSLE370|
NOT_DEFINED|NOT_APPLIC

[CICSVAR_THREADSAFE]

is the threadsafe value established for the program. It can have any of these values:

YES|NO|NOT_DEFINED

[RUNTIME_ENVIRONMENT]

is the runtime environment established for the program. It can have any of these values:

JVM_RUNTIME|LE370_RUNTIME|NON_LE370_RUNTIME|
HOTPOLL_RUNTIME

RESPONSE

is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]

is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>TRANSACTION_ABEND</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

APLH gate, START_PROGRAM function

The START_PROGRAM function of the APLI gate is used to start a compiled Java program using hot-pooling.
Input parameters

**PROGRAM**

is the eight-character name of the program.

**LINK_LEVEL**

is the 16-bit value indicating the link-level of the program.

**[CEDF_STATUS]**

indicates whether or not the EDF diagnostic screens are displayed when the program is running under the control of the execution diagnostic facility (EDF). It can have any of these values:

CEDF | NOCEDF

**[EXECUTION_SET]**

indicates whether you want CICS to link to and run the program as if it were running in a remote CICS region (with or without the API restrictions of a DPL program). It can have either of these values:

FULLAPI | DPLSUBSET | NOT_APPLIC

**[PARM_LIST_PTR]**

is an optional token identifying the parameter list for the program.

**COMMAREA**

is an optional token identifying the communications area for the program.

**[ENVIRONMENT_TYPE]**

is the environment type of the program. It can have any of these values:

EXEC | GLUE | PLT | SYSTEM | TRUE | URM

**[SYNCONRETURN]**

defines whether or not a syncpoint is to be taken on return from the linked program. It can have either of these values:

YES | NO

Output parameters

**[NEW_BLOCK]**

is a new token identifying the new language block for the program.

ASSEMBLER | C370 | COBOL | COBOL2 | LE370 | PLI

| NOT_DEFINED | NOT_APPLIC

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

**[REASON]**

is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>TRANSACTION_ABEND</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

**APLH gate, NOTIFY_REFRESH function**

The NOTIFY_REFRESH function is called to inform AP domain when a program is refreshed, so that it can quiesce all users of the program.
Application domain (AP)

**Input parameters**

**PROGRAM**

is the eight-character name of the program.

**Output parameters**

**[ABEND_CODE]**

is the four-character abend code that is to be issued by CICS when an exception response is given and the cause of the error is a transaction abend.

**RESPONSE**

is the domain’s response to the call. It can have any of these values: OK, EXCEPTION, DISASTER, INVALID, KERNERROR, PURGED

**[REASON]**

is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>TRANSACTION_ABEND</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

**APLI gate, ESTABLISH_LANGUAGE function**

The ESTABLISH_LANGUAGE function of the APLI gate is used to establish the language of a conventional compiled program.

**Input parameters**

**LOAD_POINT**

is the load point address of the program.

**ENTRY_POINT**

is the entry point address of the program.

**[PROGRAM_LENGTH]**

is the length of the program.

**[DEFINED_LANGUAGE]**

is the language defined for the program. It can have any of these values: ASSEMBLER, C370, COBOL, LE370, PLI, NOT_DEFINED

**EXECUTION_KEY**

is the key in which CICS gives control to the program, and determines whether the program can modify CICS-key storage. It can have either of these values: CICS, USER

**DATA_LOCATION**

defines whether the program can handle only 24-bit addresses (data located below the 16MB line) can handle 31-bit addresses (data located above or below the 16MB line). It can have either of these values: ANY, BELOW

**LANGUAGE_BLOCK**

is a token identifying the current language block for the program.
Output parameters

[NEW_BLOCK]
 is a new token identifying the new language block for the program.

[LANGUAGE_ESTABLISHED]
 is the language established for the program. It can have any of these values:
 ASSEMBLER|C370|COBOL|COBOL2|LE370|PLI|JVM|
 ASSEMBLER_CICS|MVSLE370|
 NOT_DEFINED|NOT_APPLIC

[CICSVAR_THREADSAFE]
 is the threadsafe value established for the program. It can have any of these values:
 YES|NO|NOT_DEFINED

[RUNTIME_ENVIRONMENT]
 is the runtime environment established for the program. It can have any of these values:
 JVM_RUNTIME|LE370_RUNTIME|NON_LE370_RUNTIME|
 HOTPOOL_RUNTIME

RESPONSE
 is the domain’s response to the call. It can have any of these values:
 OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
 is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>TRANSACTION_ABEND</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

APLI gate, START_PROGRAM function

The START_PROGRAM function of the APLI gate is used to start a program.

Input parameters

PROGRAM
 is the eight-character name of the program.

LINK_LEVEL
 is the 16-bit value indicating the link-level of the program.

[CEDF_STATUS]
 indicates whether or not the EDF diagnostic screens are displayed when the program is running under the control of the execution diagnostic facility (EDF). It can have any of these values:
 CEDF|NOCEDF

[EXECUTION_SET]
 indicates whether you want CICS to link to and run the program as if it were running in a remote CICS region (with or without the API restrictions of a DPL program). It can have either of these values:
 FULLAPI|DPLSUBSET|NOT_APPLIC
**Application domain (AP)**

[PARMLIST_PTR] is an optional token identifying the parameter list for the program.

COMMAREA is an optional token identifying the communications area for the program.

[ENVIRONMENT_TYPE] is the environment type of the program. It can have any of these values:
EXEC|GLUE|PLT|SYSTEM|TRUE|URM

[SYNCONRETURN] defines whether or not a syncpoint is to be taken on return from the linked program. It can have either of these values:
YES|NO

**Output parameters**

[NEW_BLOCK] is a new token identifying the new language block for the program.
ASSEMBLER|C370|COBOL|COBOL2|LE370|PLI
NOT_DEFINED|NOT_APPLIC

RESPONSE is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>TRANSACTION_ABEND</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

**APLJ gate, ESTABLISH_LANGUAGE function**

The ESTABLISH_LANGUAGE function of the APLI gate is used to establish the language parameters of a JAVA bytecode program.

**Input parameters**

LOAD_POINT is the load point address of the program.

ENTRY_POINT is the entry point address of the program.

[PROGRAM_LENGTH] is the length of the program.

[DEFINED_LANGUAGE] is the language defined for the program. It can have any of these values:
ASSEMBLER|C370|COBOL|LE370|PLI|NOT_DEFINED

EXECUTION_KEY is the key in which CICS gives control to the program, and determines whether the program can modify CICS-key storage. It can have either of these values:
CICS|USER
DATA_LOCATION
defines whether the program can handle only 24-bit addresses (data
located below the 16MB line) can handle 31-bit addresses (data located
above or below the 16MB line). It can have either of these values:
ANY|BELOW

LANGUAGE_BLOCK
is a token identifying the current language block for the program.

Output parameters
[NEW_BLOCK]
is a new token identifying the new language block for the program.

[LANGUAGE_ESTABLISHED]
is the language established for the program. It can have any of these
values:
ASSEMBLER|C370|COBOL|COBOL2|LE370|PLI|JVM|
ASSEMBLER_CICS|MVSLE370|
NOT_DEFINED|NOT_APPLIC

[CICSVAR_THREADSAFE]
is the threadsafe value established for the program. It can have any of
these values:
YES|NO|NOT_DEFINED

[RUNTIME_ENVIRONMENT]
is the runtime environment established for the program. It can have any of
these values:
JVM_RUNTIME|LE370_RUNTIME|NON_LE370_RUNTIME|
HOTPOOL_RUNTIME

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID.
Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>TRANSACTION_ABEND</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

APLJ gate, START_PROGRAM function
The START_PROGRAM function of the APLI gate is used to start a Java bytecode
program.

Input parameters
PROGRAM
is the eight-character name of the program.

LINK_LEVEL
is the 16-bit value indicating the link-level of the program.

[CEDF_STATUS]
indicates whether or not the EDF diagnostic screens are displayed when
Application domain (AP)

the program is running under the control of the execution diagnostic facility (EDF). It can have any of these values:
CEDF|NOCEDF

[EXECUTION_SET]
indicates whether you want CICS to link to and run the program as if it were running in a remote CICS region (with or without the API restrictions of a DPL program). It can have either of these values:
FULLAPI|DPLSUBSET|NOT_APPLIC

[PARMLIST_PTR]
is an optional token identifying the parameter list for the program.

COMMAREA
is an optional token identifying the communications area for the program.

[ENVIRONMENT_TYPE]
is the environment type of the program. It can have any of these values:
EXEC|GLUE|PLT|SYSTEM|TRUE|URM

[SYNCONRETURN]
defines whether or not a syncpoint is to be taken on return from the linked program. It can have either of these values:
YES|NO

Output parameters

[NEW_BLOCK]
is a new token identifying the new language block for the program.
ASSEMBLER|C370|COBOL|COBOL2|LE370|PLI|
NOT_DEFINED|NOT_APPLIC

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>TRANSACTION_ABEND</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

APRM gate, TRANSACTION_INITIALIZATION function

The TRANSACTION_INITIALIZATION function of the APRM gate is called from the transaction manager domain to the AP Domain during transaction initialization. The AP domain allocates the recovery table, and initializes the RM transaction token to be the address of the recovery table.

Input parameters
None.

Output parameters
RESPONSE
is the domain’s response to the call. It can have any of these values:
[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>BAD_ENVIRONMENT</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>GETMAIN_FAILURE</td>
</tr>
</tbody>
</table>

**APRM gate, TRANSACTION_TERMINATION function**

The TRANSACTION_TERMINATION function of the APRM gate is called from the transaction manager domain during transaction termination. A DFHSP TYPE=KCP macro will be issued for the transaction. The AP domain then releases the recovery table, and sets the RM transaction token to the address of a fetch-protected area.

**Input parameters**

**TERMINATION_TYPE**
is the type of transaction termination. It can have either of these values:
- NORMAL
- ABNORMAL

**[RESTART_REQUESTED]**
indicates whether or not the transaction is to be restarted on termination. It can have either of these values:
- YES
- NO

**Output parameters**

**RESPONSE**
is the domain’s response to the call. It can have any of these values:
- OK
- EXCEPTION
- DISASTER
- INVALID
- KERNERROR
- PURGED

[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>BAD_ENVIRONMENT, LINK_FAILURE</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>BACKOUT_FAILED, TRANSACTION_ABEND</td>
</tr>
</tbody>
</table>

**APRM gate, INQUIRE function**

The INQUIRE function of the APRM gate is used to extract the resource manager attributes of a transaction.

**Input parameters**

**[UOW_TOKEN]**
is the optional extended token (ETOKEN) for the transaction.

**Output parameters**

**[PROHIBIT_RESTART]**
indicates whether or not the transaction is to be prevented from restarting. It can have either of these values:
Application domain (AP)

YES|NO

[SYNCPOINT_TAKEN]
indicates whether or not a syncpoint is to be taken on transaction termination. It can have either of these values:
YES|NO

[CICS_UOW_ID]
is the optional ETOKEN identifying the CICS unit-of-work for the transaction.

[EXTERNAL_UOW_ID]
is the optional ETOKEN identifying the non-CICS unit-of-work for the transaction.

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

APRT gate, ROUTE_TRANSACTION function

The ROUTE_TRANSACTION function of the APRT gate is used to dynamically route transactions (which are defined to be dynamic and not automatically initiated) based on decisions made by the dynamic transaction routing program. For transactions which are automatically initiated or are defined to be remote and not dynamic, DFHAPRT will statically route such transactions.

Input parameters

DYNAMIC
indicates whether or not the transaction is defined as dynamic. It can have either of these values:
YES|NO

REMOTE
indicates whether or not the transaction is defined as remote. It can have either of these values:
YES|NO

REMOTE_NAME
is the four-character transaction identifier by which this transaction is to be known on the remote CICS region.

REMOTE_SYSTEM
is the eight-character name of the remote CICS region to which the transaction is to be routed.

DTRTRAN
indicates whether or not dynamic transaction routing is available. It can have either of these values:
YES|NO

Output parameters

RAN_LOCALLY
indicates whether or not the transaction ran on the local CICS region (that is, was not routed to a remote CICS region). It can have either of these values:
YES|NO
ABEND_CODE
is the four-character transaction abend code issued if the transaction terminates abnormally.

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>PROGRAM_NOT_FOUND, TRANSACTION_ABEND, ISC_DISABLED, REMOTE_CONN_OOS, REMOTE_CONN_OOS_SYS_CHGD, ALL_SESSIONS_BUSY, ROUTE_FAILED, DTRTRAN_REJECTED</td>
</tr>
</tbody>
</table>

APTC gate, CANCEL function

The CANCEL function of the APTC gate invalidates the listening function.

Input parameters
TOKEN
is the token for the session TCTTE

Output parameters
RESPONSE
is the domain’s response to the call. It can have any of the following values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>TOKEN_UNKNOWN, TC_ERROR</td>
</tr>
</tbody>
</table>

APTC gate, CLOSE function

The CLOSE function of the APTC gate is used in cleanup.

Input parameters
TOKEN
is the token for the session TCTTE

Output parameters
RESPONSE
is the domain’s response to the call. It can have any of the following values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:
Application domain (AP)

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>TOKEN_UNKNOWN, TC_ERROR</td>
</tr>
</tbody>
</table>

APTC gate, CLOSE function

The CLOSE function of the APTC gate is used in cleanup.

Input parameters

TOKEN

is the token for the session TCTTE

Output parameters

RESPONSE

is the domain’s response to the call. It can have any of the following values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>TOKEN_UNKNOWN, TC_ERROR</td>
</tr>
</tbody>
</table>

APTC gate, EXTRACT_PROCESS function

The EXTRACT_PROCESS function of the APTC gate extracts information for the request.

Input parameters

NONE

No input parameters

Output parameters

CONVID

is the conversation id (which is the session tctte termid).

PROCESS_NAME

is the name of the process to be invoked

SYNCELEVEL

is the synlevel of the conversation

PIPDATA

Applicable only for LU6.2 conversations

PIPDATA LENGTH

Applicable only for LU6.2 conversations.

RESPONSE

is the domain’s response to the call. It can have any of the following values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON] is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:
APTC gate, ISSUE_NOTIFY function

The ISSUE_NOTIFY function of the APTC gate is used to inform the program DFHAPTC that a request has arrived.

Input parameters

TOKEN
is the token for the session TCTTE

USER_TOKEN
is the token supplied the the person who is to be notified.

ACTION
specifies the action that should be taken.

Output parameters

RESPONSE
is the domain’s response to the call. It can have any of the following values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>TOKEN_UNKNOWN, ISSUE_NOTIFY_ERROR, ACTION_UNKNOWN</td>
</tr>
</tbody>
</table>

APTC gate, LISTEN function

The LISTEN function of the APTC gate is used to update the TCTTE with the user token.

Input parameters

TOKEN
is the token for the session TCTTE

USER_TOKEN
is a token supplied by requeststreams.

Output parameters

RESPONSE
is the domain’s response to the call. It can have any of the following values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>TOKEN_UNKNOWN, TC_ERROR</td>
</tr>
</tbody>
</table>
APTC gate, OPEN function

The OPEN function of the APTC gate is used to allocate a session to the specified AOR.

**Input parameters**

**SYSID**

specifies the name of the AOR

**TRANID**

is the transaction name to be attached in the AOR.

**NETNAME**

specifies the netname or applid of the AOR.

**QUEUE**

is the queue option specified by the routing program.

**Output parameters**

**ERROR CODE**

indicates the code passed back from the allocate procedure.

**RESPONSE**

is the domain’s response to the call. It can have any of the following values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

**[REASON]**

is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>OPEN_ERROR</td>
</tr>
</tbody>
</table>

APTC gate, RECEIVE function

The RECEIVE function of the APTC gate is used to receive data.

**Input parameters**

**TOKEN**

is the token for the session TCTTE

**RECEIVE_BUFFER**

is the buffer into which the reply is to be placed.

**Output parameters**

**LAST**

is an indicator to indicate if this is the last flow.

**RESPONSE**

is the domain’s response to the call. It can have any of the following values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

**[REASON]**

is returned when RESPONSE is EXCEPTION. Possible values are:
APTC gate, SEND function

The SEND function of the APTC gate is used to send the request to the AOR.

Input parameters

TOKEN

is the token for the session TCTTE

SEND_BLOCK

is the block data with the length and send data pointer.

PREFIX_AREA

specifies the requeststreams information.

LAST

is an indicator to indicate if this is the last flow.

Output parameters

RESPONSE

is the domain’s response to the call. It can have any of the following values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON]

is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>TOKEN_UNKNOWN, TC_ERROR, NO_TCTTE</td>
</tr>
</tbody>
</table>

APTC gate, SET_SESSION function

The SET_SESSION function of the APTC gate is used to send the request to the AOR.

Input parameters

TOKEN

is the token for the session TCTTE

RECOVERY_STATUS

indicates if recovery is necessary.

Output parameters

RESPONSE

is the domain’s response to the call. It can have any of the following values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON]

is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
</tbody>
</table>
**(AP) Application domain**

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>TOKEN_UNKNOWN, TC_ERROR</td>
</tr>
</tbody>
</table>

**APTD gate, WRITE_TRANSIENT_DATA function**

The WRITE_TRANSIENT_DATA function of the APTD gate is used to write a single record (or multiple records) to a named transient data queue.

**Input parameters**

**QUEUE**

specifies the name of the queue to which the data is to be written.

**FROM_LIST**

is a list specifying the address and the length of each record that is to be written to the specified queue.

**[RSL_CHECK]**

states whether resource-level checking is to be carried out. It can take the values:

YES | NO

**Output parameters**

**[TD_RECORD]**

indicates the number of records that were successfully written to the transient data queue.

**[TD_MIN_LENGTH]**

indicates the minimum allowable length of a transient data record if a RESPONSE of EXCEPTION, and a REASON of LENGTH_ERROR is returned.

**[TD_MAX_LENGTH]**

indicates the maximum allowable length of a transient data record if a RESPONSE of EXCEPTION, and a REASON of LENGTH_ERROR is returned.

**RESPONSE**

is Transient Data’s response to the call. It can have any of the following values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

**[REASON]**

is returned when RESPONSE is EXCEPTION, DISASTER, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, CSM_ERROR, DCT_ERROR, DIRECTORY_MGR_ERROR, LOGIC_ERROR</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>QUEUE_REMOTE, QUEUE_NOTFOUND, QUEUE_NOT_AUTH, QUEUE_DISABLED, QUEUE_NOT_OPEN, QUEUE_NOT_OUTPUT, QUEUE_FULL, NO_SPACE, IO_ERROR, LENGTH_ERROR, LOCKED, NO_RECOVERY_TABLE</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FROM_LIST_P, INVALID_FROM_LIST_N, INVALID_FROM_P, INVALID_FROM_N, INVALID_RSL_CHECK</td>
</tr>
</tbody>
</table>
APTD gate, READ_TRANSIENT_DATA function

The READ_TRANSIENT_DATA function of the APTD gate is used to read a single record from a named transient data queue.

Input parameters

**QUEUE**

specifies the name of the queue to which a record is to be read.

**INTO**

specifies a piece of storage into which the record is placed.

**SUSPEND**

specifies whether the caller wishes to wait if the record to be read has not been committed to the queue yet. It can take the values:

- YES | NO

**[RSL_CHECK]**

states whether resource level checking is to be carried out. It can take the values:

- YES | NO

**[DATA_LOC]**

if this is a READ TD SET rather than an INTO, DATA_LOC specifies whether Transient Data should obtain the required SET storage from above or below the 16MB line. It can take the values:

- ANY | BELOW

**[DATA_KEY]**

if this is a READ TD SET rather than an INTO, DATA_KEY specifies whether Transient Data should obtain the required SET storage from CICS key or user key storage. It can take the values:

- CICS | USER

Output parameters

**RESPONSE**

is Transient Data’s response to the call. It can have any of the following values:

- OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

**[REASON]**

is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, CSM_ERROR, DCT_ERROR, DIRECTORY_MGR_ERROR, LOGIC_ERROR</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>QUEUE_REMOTE, QUEUE_NOT_FOUND, QUEUE_NOT_AUTH, QUEUE_DISABLED, QUEUE_NOT_OPEN, QUEUE_NOT_INPUT, QUEUE_BUSY, IO_ERROR, LENGTH_ERROR, LOCKED</td>
</tr>
</tbody>
</table>

APTD gate, DELETE_TRANSIENT_DATA function

The DELETE_TRANSIENT_DATA function of the APTD gate is used to delete the specified transient data queue.
Application domain (AP)

Input parameters

QUEUE
specifies the name of the queue to which the data is to be deleted.

[RSL_CHECK]
states whether resource level checking is to be carried out. It can take the values:
YES|NO

[DISCARDING_DEFINITION]
states whether this DELETEQ request is part of an attempt by Transient Data to discard a transient data queue definition. It can take the values:
YES|NO

Output parameters

RESPONSE
is Transient Data’s response to the call. It can have any of the following values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, CSM_ERROR, DCT_ERROR, DIRECTORY_MGR_ERROR, LOGIC_ERROR</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>QUEUE_REMOTE, QUEUE_NOT_FOUND, QUEUE_NOT_AUTH, QUEUE_DISABLED, QUEUE_EXTRA, IO_ERROR, LOCKED, NO_RECOVERY_TABLE</td>
</tr>
</tbody>
</table>

APTD gate, RESET_TRIGGER_LEVEL function

The RESET_TRIGGER_LEVEL function of the APTD gate is used to reset a transient data queue so that another trigger transaction can be attached. Sometimes it is necessary to include the RESET_TRIGGER_LEVEL function if a trigger transaction abends.

Input parameters

QUEUE
specifies the name of the queue for which the trigger transaction is to be reset.

Output parameters

RESPONSE
is Transient Data’s response to the call. It can have any of the following values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER. Possible values are: ABEND, DCT_ERROR, CSM_ERROR, DIRECTORY_MGR_ERROR, and LOGIC_ERROR.
**APTD gate, INITIALISE_TRANSIENT_DATA function**

The INITIALISE_TRANSIENT_DATA function of the APTD gate is invoked as part of the initialization process for the transient data facility.

**Input parameters**

None.

**Output parameters**

RESPONSE is Transient Data’s response to the call. It can have any of the following values:

- OK
- EXCEPTION
- DISASTER
- INVALID
- KERNERROR
- PURGED

[REASON] is returned when RESPONSE is DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, CSM_ERROR, DCT_ERROR, DIRECTORY_MGR_ERROR, LOGIC_ERROR</td>
</tr>
</tbody>
</table>

**APXM gate, TRANSACTION_INITIALIZATION function**

The TRANSACTION_INITIALIZATION function of the APXM gate is called from the transaction manager domain to the AP Domain during transaction initialization. The AP domain allocates the AP domain transaction lifetime control blocks, and anchors them in the AP domains transaction token.

**Input parameters**

None.

**Output parameters**

RESPONSE is the domain’s response to the call. It can have any of these values:

- OK
- EXCEPTION
- DISASTER
- INVALID
- KERNERROR
- PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>GETMAIN_FAILURE</td>
</tr>
</tbody>
</table>

**APXM gate, RMI_START_OF_TASK function**

The RMI_START_OF_TASK function of the APXM gate is called from transaction manager domain to the AP Domain during transaction initialization. The AP domain invokes any task-related user exits enabled for start of task.

**Input parameters**

None.

**Output parameters**

RESPONSE is the domain’s response to the call. It can have any of these values:

- OK
- EXCEPTION
- DISASTER
- INVALID
- KERNERROR
- PURGED
**APXM gate, TRANSACTION_TERMINATION function**

The TRANSACTION_TERMINATION function of the APXM gate is called from the transaction manager domain during transaction termination, and releases AP domain transaction lifetime resources.

**Input parameters**

**TERMINATION_TYPE**
- is the type of transaction termination. It can have either of these values:
  - NORMAL
  - ABNORMAL

**Output parameters**

**RESPONSE**
- is the domain’s response to the call. It can have any of these values:
  - OK
  - EXCEPTION
  - DISASTER
  - INVALID
  - KERNERROR
  - PURGED

[REASON]
- is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>FREEMAIN_FAILURE</td>
</tr>
</tbody>
</table>

**BRAT gate, ATTACH function**

The ATTACH function of the BRAT gate is called to attach a transaction with a bridge primary client.

**Input parameters**

**TRANSACTION_ID**
- The 4 byte transaction id of the user transaction to be attached.

[BREXIT]
- An optional program name to be used as the bridge exit. If this is not specified, DFHBRAT will get the default value from transaction manager. If there is no default bridge exit, an error is returned.

[USERID]
- The USERID that should be signed-on to the terminal. This is only set when no facility token is passed.

[BRDATA]
- The address and length of a block of storage containing data to be passed to bridge exit. This is used as part of the primary client data.

**Output parameters**

**RESPONSE**
- is the domain’s response to the call. It can have any of these values:
  - OK
  - EXCEPTION
  - DISASTER
  - INVALID
  - KERNERROR
  - PURGED

[REASON]
- is returned when RESPONSE is EXCEPTION, DISASTER or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NO_BREXIT, NO_STORAGE, USERID_NOT_AUTH_BREXIT, NOT_FOUND, DISABLED, NO_XM_STORAGE,</td>
</tr>
<tr>
<td></td>
<td>NOT_ENABLED_FOR_SHUTDOWN, STATE_SYSTEM_ATTACH</td>
</tr>
</tbody>
</table>
**BRIQ gate, INQUIRE_CONTEXT function**

The INQUIRE_CONTEXT of the BRIQ gate is called to inquire on bridge state data.

**Input parameters**

**[TRANSACTION_TOKEN]**

The XM transaction token for the task to be inquired upon.

**Output parameters**

**[CALL_EXIT_FOR_SYNC]**

Can have either of these two values:
- YES
- NO

**[BRIDGE_ENVIRONMENT]**

Can have either of these two values:
- YES
- NO

**[CONTEXT]**

The transaction context. It can have either of these values:
- NORMAL
- BRIDGE
- BREXIT

**[START_CODE]**

The emulated startcode of the user transaction

**[BRIDGE_TRANSACTION_ID]**

The transaction identifier of the bridge monitor (if CONTEXT is BRIDGE or BREXIT).

**[BRIDGE_EXIT_PROGRAM]**

The name of the bridge exit program (if CONTEXT is BRIDGE or BREXIT).

**[BRIDGE_FACILITY_TOKEN]**

A token identifying the bridge facility

**[IDENTIFIER]**

Data created by the bridge exit for problem determination purposes.

**[BRDATA]**

Data passed to the bridge exit during attach.

**RESPONSE**

is the domain’s response to the call. It can have any of these values:
- OK
- EXCEPTION
- DISASTER
- INVALID
- KERNERROR
- PURGED

**[REASON]**

is returned when RESPONSE is EXCEPTION, DISASTER or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>BAD_TOKEN, NO_TRANSACTION_ENVIRONMENT</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT</td>
</tr>
</tbody>
</table>

**RESPONSE Possible REASON values**

**DISASTER**

- ABEND

**INVALID**

- INVALID_FORMAT

---

Chapter 4. Application domain (AP) 53
ICXM gate, **BIND_FACILITY function**

The **BIND_FACILITY** function of the ICXM gate is used to associate a transaction with the ICE that caused the transaction to be started.

**Input parameters**
None.

**Output parameters**

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

- OK
- EXCEPTION
- DISASTER
- INVALID
- KERNERROR
- PURGED

**[REASON]**

is returned when RESPONSE is DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
</tbody>
</table>

---

ICXM gate, **RELEASE_FACILITY function**

The **RELEASE_FACILITY** function of the ICXM gate is used to disassociate a transaction from the ICE that caused the transaction to be attached.

**Input parameters**

**TERMINATION_TYPE**

is the type of termination of the transaction to ICE association. It can have either of these values:

- NORMAL
- ABNORMAL

**[RESTART_REQUESTED]**

indicates whether or not the transaction is to be restarted. It can have either of these values:

- YES
- NO

**Output parameters**

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

- OK
- EXCEPTION
- DISASTER
- INVALID
- KERNERROR
- PURGED

**[REASON]**

is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>RESTART_FAILURE</td>
</tr>
</tbody>
</table>

---

ICXM gate, **INQUIRE_FACILITY function**

The **INQUIRE_FACILITY** function of the ICXM gate is used to inquire about the interval control facilities that support facility management calls from the transaction management domain.

**Input parameters**

**[FACILITY_TOKEN]**

is the token identifying the transaction that has been trigger-level attached.
Output parameters

FACILITY_NAME
is the four-character name of the transaction that has been trigger-level
attached.

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

TFXM gate, INIT_XM_CLIENT function

The INIT_XM_CLIENT function of the TFXM gate is the initialization phase of the
transaction initialization that has been initiated from a terminal or an LU6.1
session.

Input parameters

[PRIMARY_CLIENT_BLOCK]
is the address of the TCTTE and its length.

Output parameters

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|DISASTER|KERNERROR

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
</tbody>
</table>

TFXM gate, BIND_XM_CLIENT function

The BIND_XM_CLIENT function of the TFXM gate is the bind phase of the
transaction initialization that has been initiated from a terminal or an LU6.1
session.

Input parameters

[PRIMARY_CLIENT_BLOCK]
is the address of the TCTTE and its length.

Output parameters

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|DISASTER|KERNERROR

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
</tbody>
</table>

MRXM gate, INIT_XM_CLIENT function

The INIT_XM_CLIENT function of the MRXM gate is the initialization phase of the
transaction initialization that has been initiated from a terminal or an MRO session.
Application domain (AP)

Input parameters

[PRIMARY_CLIENT_BLOCK]
   is the address of the TCTTE and its length.

Output parameters

RESPONSE
   is the domain’s response to the call. It can have any of these values:
   OK|DISASTER|KERNERROR

[REASON]
   is returned when RESPONSE is DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
</tbody>
</table>

MRXM gate, BIND_XM_CLIENT function

The BIND_XM_CLIENT function of the MRXM gate is the bind phase of the transaction initialization that has been initiated from a terminal or an MRO session.

Input parameters

[PRIMARY_CLIENT_BLOCK]
   is the address of the TCTTE and its length.

Output parameters

RESPONSE
   is the domain’s response to the call. It can have any of these values:
   OK|DISASTER|KERNERROR

[REASON]
   is returned when RESPONSE is DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
</tbody>
</table>

62XM gate, INIT_XM_CLIENT function

The INIT_XM_CLIENT function of the 62XM gate is the initialization phase of the transaction initialization that has been initiated from a terminal or an LU6.2 or APPC session.

Input parameters

[PRIMARY_CLIENT_BLOCK]
   is the address of the TCTTE and its length.

Output parameters

RESPONSE
   is the domain’s response to the call. It can have any of these values:
   OK|DISASTER|KERNERROR

[REASON]
   is returned when RESPONSE is DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
</tbody>
</table>
62XM gate, BIND_XM_CLIENT function

The BIND_XM_CLIENT function of the 62XM gate is the bind phase of the transaction initialization that has been initiated from a terminal or an LU6.2 or APPC session.

**Input parameters**

[PRIMARY_CLIENT_BLOCK]

is the address of the TCTTE and its length.

**Output parameters**

RESPONSE

is the domain’s response to the call. It can have any of these values:

OK|DISASTER|KERNERROR

[REASON]

is returned when RESPONSE is DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
</tbody>
</table>

RTSU gate, COMMIT_SURROGATE function

The COMMIT_SURROGATE function of the RTSU gate is used to update the state of a surrogate TCTTE when a Unit of Work is committed or backed out.

**Input parameters**

SURROGATE

The address of the surrogate TCTTE

[UOW_STATUS]

Indicates if the Unit of Work is being committed or backed out. It can have either of these two values:

FORWARD|BACKWARD

**Output parameters**

FREE_REQUIRED

Indicates if the surrogate should now be freed (because, for instance, the relay link has been freed). It can have either of these two values:

YES|NO

RESPONSE

is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]

is returned when RESPONSE is DISASTER or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
</tbody>
</table>
### RTSU gate, FREE_SURROGATE function

The FREE_SURROGATE function of the RTSU gate is used to free a surrogate TCTTE from the currently executing task.

#### Input parameters
- **SURROGATE**
  - The address of the surrogate TCTTE

#### Output parameters
- **RESPONSE**
  - The domain’s response to the call. It can have any of these values:
    - OK
    - EXCEPTION
    - DISASTER
    - INVALID
    - KERNERROR
    - PURGED

- **[REASON]**
  - Is returned when RESPONSE is DISASTER or INVALID. Possible values are:
    - ABEND

### RTSU gate, GET_RECOVERY_STATUS function

The GET_RECOVERY_STATUS function of the RTSU gate is used to determine what actions are required of the relay link at syncpoint.

#### Input parameters
- **SURROGATE**
  - The address of the surrogate TCTTE

#### Output parameters
- **RECOVERY_STATUS**
  - Indicates the syncpoint protocols required on the relay link. It can have any of these values:
    - NECESSARY
    - UNNECESSARY
    - SYNC_LEVEL_1

- **ABORT_ALLOWED**
  - Indicates whether, during the syncpoint protocols, an ABORT FMH7 should be sent on the relay link. It can have either of these values:
    - YES
    - NO

- **RESPONSE**
  - The domain’s response to the call. It can have any of these values:
    - OK
    - EXCEPTION
    - DISASTER
    - INVALID
    - KERNERROR
    - PURGED

- **[REASON]**
  - Is returned when RESPONSE is DISASTER or INVALID. Possible values are:
The PREPARE_SURROGATE function of the RTSU gate is used to update the state of a surrogate TCTTE at the start of syncpoint.

**Input parameters**

- **SURROGATE**: The address of the surrogate TCTTE
- **INITIATOR**: Indicates if the associated relay link is the initiator of the syncpoint request. It can have either of these two values: YES|NO

**Output parameters**

- **RESPONSE**: is the domain’s response to the call. It can have any of these values: OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED
- **[REASON]**: is returned when RESPONSE is EXCEPTION, DISASTER or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>INVALID_STATE</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION, INVALID_SURROGATE</td>
</tr>
</tbody>
</table>

The RESET_SURROGATE function of the RTSU gate is used to restore the state of a surrogate TCTTE when ISSUE_ABEND or ISSUE_ERROR was received on the relay link in reply to an ISSUE PREPARE request.

**Input parameters**

- **SURROGATE**: The address of the surrogate TCTTE
- **REPLY_TO_PREPARE**: Indicates which reply was received in response to ISSUE_PREPARE. It can have either of these two values: ISSUE_ERROR|ISSUE_ABEND

**Output parameters**

- **RESPONSE**: is the domain’s response to the call. It can have any of these values: OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>INVALID_STATE</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION, INVALID_SURROGATE</td>
</tr>
</tbody>
</table>
**SAIQ gate, INQUIRE_SYSTEM function**

The INQUIRE_SYSTEM function of the SAIQ gate is used to inquire about system data owned by the application domain.

**Input parameters**

**[GMMTEXT]**

is an optional token identifying the text of the “good-morning” message.

**Output parameters**

**[AKP]** is a fullword binary field indicating the activity keypoint frequency, in the range 200 through 65,535, of the local CICS region.

**[CICSREL]**

is a 4-character string indicating the level (version and release numbers) of CICS code present.

**[CICSSSTATUS]**

is the current status of the local CICS system. It can have any of these values:

| ACTIVE | FIRSTQUIESCE | FINALQUIESCE | INITIALIZING |

**[CICSSYS]**

is the one-character identifier of the operating system for which the running CICS system has been built. A value of “X” represents MVS system with extended addressing.

**[CWA]**

is the address of the CWA.

**[CWALENGTH]**

is the length (in bytes) of the CWA.

**[DATE]**

is a four-character packed-decimal value indicating the current date (00yydddcc, where yy=years, ddd=days, c is the sign).

**[DCE_SUFFIX]**

is the two-character suffix of the DCE initialization side file, as specified on the DCESUFFX system initialization parameter.

**[DTRPRGRM]**

is the 8-character name of the program controlling the dynamic routing of transactions.

**[GMMLENGTH]**

is a halfword binary field indicating the length of the “good-morning” message text.

**[GMMTRANID]**

is the four-character identifier of the “good-morning” transaction.
[INITSTATUS] is the initialization status of the local CICS region. It can have any of these values:
FIRSTINIT|SECONDINIT|THIRDINIT|INITCOMPLETE

[JOBNAME] is the eight-character MVS job name for the local CICS region.

[OPREL] indicates the release number of the operating system currently running. The values is ten times the formal release number. For example, “21” represents Release 2.1.

[OPSYS] is a one-character identifier indicating the type of operating system currently running. A value of “X” represents MVS.

[PLTPI] is the two-character suffix of the program list table, which contains a list of programs to be run in the final stages of system initialization.

[SECURITYMGR] indicates whether an external security manager (such as RACF) is active in the CICS region, or whether no security is being used. It can have either of these values:
EXTSECURITY|NOSECURITY

[SHUTSTATUS] is the shutdown status of the local CICS region. It can have any of these values:
CONTROLSHUT|SHUTDOWN|CANCELLED|NOTSHUTDOWN

[STARTUP] is the type of startup used for the local CICS region. It can have any of these values:
COLDSTART|WARMSTART|EMERGENCY|LOGTERM|STANDBY|AUTOSTART

[STARTUPDATE] is a four-character packed-decimal value indicating the date on which the local CICS region was started.

[TERMURM] is the eight-character name of the terminal autoinstall program.

[TIMEOFDAY] is a four-character packed-decimal value indicating the time at which the local CICS region was started (hhmmssstc, where hh=hours, mm=minutes, ss=seconds, c is the sign).

[XRFSTATUS] indicates whether the local CICS region is a PRIMARY (active) or TAKEOVER (alternate) XRF CICS region, or has no XRF support. It can have any of these values:
PRIMARY|TAKEOVER|NOXRF

RESPONSE is the domain’s response to the call. It can have any of these values:
OK|exception|disaster|invalid|kernerror|purged

[REASON] is returned when RESPONSE is DISASTER or EXCEPTION. Possible values
SAIQ gate, SET_SYSTEM function

The SET_SYSTEM function of the SAIQ gate is used to set system data values owned by the application domain.

Input parameters

[AKP] is a fullword binary field indicating the activity keypoint frequency, in the range 200 through 65535, of the local CICS region.

[DCE_SUFFIX] is the two-character suffix of the DCE initialization side file.

[DTRPRGRM] is the 8-character name of the program controlling the dynamic routing of transactions.

[GMMTEXT] is an optional token identifying the text of the “good-morning” message.

[GMMLENGTH] is a halfword binary field indicating the length of the “good-morning” message text.

Output parameters

RESPONSE is the domain’s response to the call. It can have any of these values:

OK, EXCEPTION, DISASTER, INVALID, KERNERROR, PURGED

[REASON] is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

RESPONSE Possible REASON values
DISASTER ABEND, SET_FAILED, LOOP
EXCEPTION LENGTH_ERROR, UNKNOWN_DATA

TDOC gate, OPEN_TRANSIENT_DATA function

The OPEN_TRANSIENT_DATA function of the TDOC gate is used to open an extrapartition transient data queue.

Input parameters

QUEUE specifies the name of the extrapartition transient data queue to be opened.

TD_QUEUE_TOKEN can be specified instead of QUEUE. The token uniquely identifies the extrapartition queue to be opened.
Output parameters

RESPONSE

is Transient Data’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]

is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>QUEUE_INTRA, QUEUE_REMOTE, QUEUE_OPEN, QUEUE_NOT_FOUND</td>
</tr>
<tr>
<td>DISASTER</td>
<td>DCT_ERROR, DIRECTORY_MGR_ERROR, LOGIC_ERROR</td>
</tr>
</tbody>
</table>

TDOC gate, CLOSE_TRANSIENT_DATA function

The CLOSE_TRANSIENT_DATA function of the TDOC gate is used to close an extrapartition transient data queue.

Input parameters

QUEUE

specifies the name of the extrapartition transient data queue to be closed.

TD_QUEUE_TOKEN

can be specified instead of QUEUE. The token uniquely identifies the extrapartition queue to be closed.

Output parameters

RESPONSE

is Transient Data’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]

is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, DCT_ERROR, DIRECTORY_MGR_ERROR, LOGIC_ERROR</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>QUEUE_INTRA, QUEUE_REMOTE, QUEUE_CLOSED, QUEUE_NULL, QUEUE_NOT_FOUND, QUEUE_NOT_CLOSED</td>
</tr>
</tbody>
</table>

TDOC gate, CLOSE_ALL_EXTRA_TD_QUEUES function

The CLOSE_ALL_EXTRA_TD_QUEUES function of the TDOC gate closes all extrapartition transient data queues which are currently open in the system. The CLOSE_ALL_EXTRA_TD_QUEUES function is usually invoked as part of a warm shutdown.

Input parameters

None.
**Output parameters**

**RESPONSE**

is Transient Data’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**[REASON]**

is returned when RESPONSE is DISASTER. Possible values are: ABEND, DCT_ERROR, DIRECTORY_MGR_ERROR, and LOGIC_ERROR.

**TDTM gate, ADD_REPLACE_TDQUEUE function**

The ADD_REPLACE_TDQUEUE function of the TDTM gate is used to install a transient data queue definition.

**Input parameters**

**QUEUE_NAME**

specifies the name of the queue to be installed.

**TD_QUEUE_TOKEN**

can be specified instead of QUEUE. The token uniquely identifies a DCT entry that has already been built, but needs to be installed.

**TD_TYPE**

specifies the queue type. Possible values are:

EXTRA|INTRA|INDIRECT|REMOTE

**BLOCK_LENGTH**

specifies the block length of an extrapartition queue.

**BUFFER_NUMBER**

specifies the number of buffers to be associated with an extrapartition queue.

**DDNAME**

specifies the DDNAME to be associated with an extrapartition queue.

**DISPOSITION**

specifies the disposition of the data set to be associated with an extrapartition queue. Possible values are:

SHR|OLD|MOD

**DSNAME**

specifies the DSNAME of the data set to be associated with an extrapartition queue.

**ERROR_OPTION**

specifies the action to be taken in the event of an I/O error. This input parameter applies to extrapartition queues only. Possible values are:

IGNORE|SKIP

**FACILITY**

specifies the facility associated with this intrapartition queue when a trigger transaction is attached. Possible values are:

TERMINAL|FILE|SYSTEM

**FACILITY_ID**

specified together with the FACILITY option, FACILITY_ID identifies the facility that the trigger transaction should be associated with.

**INDIRECT_DEST**

specifies the destination queue if this queue is an indirect queue.
WAIT_ACTION
specifies the action to be taken if this logically recoverable intrapartition queue suffers an indoubt failure. Possible values are:
QUEUE|REJECT
WAIT specifies whether this logically recoverable intrapartition queue can wait for the resolution of an indoubt failure. Possible values are:
YES|NO
OPEN_TIME
specifies whether this extrapartition queue should be opened as part of installation processing. Possible values are:
INITIAL|DEFERRED
RECORD_LENGTH
specifies the record length of an extrapartition queue in bytes.
RECORD_FORMAT
specifies the format of records held in an extrapartition queue. Possible values are:
FIXUNB|FIXUNBA|FIXUNBM|FIXBLK|FIXBLKA|FIXBLKM|
VARBLK|VARBLKA|VARBLKM|VARUNB|VARUNBA|
VARUNBM|UNDEFINED
RECOVERY
specifies the recovery type of an intrapartition queue. Possible values are:
NO|PH|LG
REMOTE_NAME
specifies the remote name of the queue if this is a remote queue definition.
REMOTE_SYSTEM
specifies the remote system identifier (SYSID) if this is a remote queue definition.
REWIND
specifies where the tape is positioned in relation to the end of the data set. This input parameter applies to extrapartition queues only. Possible values are:
REREAD|LEAVE
TRANSACTION_ID
specifies the ATI transaction to be invoked when the trigger level is reached.
TRIGGER_LEVEL
specifies the trigger level of the intrapartition queue.
TYPE_FILE
indicates whether this queue is:
• An input queue
• An output queue
• Whether the queue is to be read backwards.

Possible values are:
INPUT|OUTPUT|RDBACK
USERID
specifies the userid to be associated with a trigger-level attached transaction.
SYSOUTCLASS
specifies the SYSOUT class to be used for the associated output extrapartition queue.

Output parameters
RESPONSE
is Transient Data’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, CATALOG_WRITE_FAILED, DCT_ERROR, DIRECTORY_MGR_ERROR, LOGIC_ERROR</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>COLD_START_IN_PROGRESS, DDNAME_NOT_FOUND, DFHINTRA_NOT_OPENED, DISABLE_PENDING, DUPLICATE, INSUFFICIENT_STORAGE, NOT_CLOSED, NOT_DISABLED, NOT_EMPTY, NOT_SAME_TYPE, QUEUE_NOT_OPENED, SECURITY_FAILURE, USERID_NOTAUTHED</td>
</tr>
</tbody>
</table>

TDTM gate, INQUIRE_TDQUEUE function
The INQUIRE_TDQUEUE function of the TDTM gate is used to inquire on a specified queue.

Input parameters
QUEUE_NAME
specifies the name of the queue to be inquired upon.

Output parameters
[ATI_FACILITY]
specifies the facility associated with this intrapartition queue when a trigger transaction is attached. Possible values are:
TERMINAL|FILE|SYSTEM

[ATI_TERMID]
specified together with the FACILITY option, FACILITY_ID identifies the facility that the trigger transaction should be associated with.

[ATI_TRANID]
specifies the ATI transaction to be invoked when the trigger level is reached.

[BUFFER_NUMBER]
specifies the number of buffers to be associated with an extrapartition queue.

[DDNAME]
specifies the DDNAME to be associated with an extrapartition queue.

[DISPOSITION]
specifies the disposition of the data set to be associated with an extrapartition queue. Possible values are:
SHR|OLD|MOD
[DSNAME] specifies the DSNAME of the data set to be associated with the extrapartition queue.

[EMPTY_STATUS] indicates whether the queue contains any records, and whether the queue is full. This option applies to extrapartition queues only. Possible values are:
FULL|EMPTY|NOTEMPTY

[ENABLE_STATUS] indicates the status of the queue. Possible values are:
ENABLED|DISABLING|DISABLED

[ERROR_OPTION] specifies what action is to be taken in the event of an I/O error. This option applies to extrapartition queues only. Possible values are:
IGNORE|SKIP

[INDIRECT_DEST] specifies the destination queue if this queue is an indirect queue.

[WAIT] specifies whether this logically recoverable intrapartition queue can wait for the resolution of an indoubt failure. Possible values are:
YES|NO

[WAIT_ACTION] specifies the action to be taken if this logically recoverable intrapartition queue suffers an indoubt failure. Possible values are:
QUEUE|REJECT

[NUM_ITEMS] states the number of committed items in the queue.

[OPEN_STATUS] indicates whether the queue is open. Possible values are:
OPEN|CLOSED

[RECORD_FORMAT] specifies the format of the records held on the extrapartition queue. Possible values are:
FIXUNB|FIXUNBA|FIXUNBM|FIXBLK|FIXBLKA|FIXBLKM|
VARBLK|VARBLKA|VARBLKM|VARUNB|VARUNBA|
VARUNBM|UNDEFINED

[RECORD_LENGTH] specifies the record length of the extrapartition queue.

[RECOVERY] specifies the recovery type of an intrapartition queue. Possible values are:
NO|PH|LG

[REMOTE_NAME] specifies the remote name of the queue if this is a remote queue definition.

[REWIND] specifies where the tape is positioned in relation to the end of the data set. This input parameter applies to extrapartition queues only. Possible values are:
REREAD|LEAVE
Application domain (AP)

[TD_QUEUE_TOKEN]
states which token is associated with this queue.

[TD_TYPE]
specifies the queue type. Possible values are:
EXTRA|INTRA|INDIRECT|REMOTE

[TRIGGER_LEVEL]
specifies the trigger level of the intrapartition queue.

[TYPE_FILE]
specifies whether this queue is:
• An input queue
• An output queue
• Whether it is a queue that is to be read backwards.
Possible values are:
INPUT|OUTPUT|RDBACK

[USERID_TOKEN]
indicates which token is associated with the USERID that was specified for this intrapartition queue.

[SYSOUTCLASS]
specifies the SYSOUT class to be used for the associated output extrapartition queue.

[BLOCK_LENGTH]
specifies the block length of an extrapartition queue.

RESPONSE
is Transient Data’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

REASON
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, DCT_ERROR, DIRECTORY_MGR_ERROR, LOGIC_ERROR</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>QUEUE_NOT_FOUND</td>
</tr>
</tbody>
</table>

TDTM gate, START_BROWSE_TDQDEF function

The START_BROWSE_TDQDEF function of the TDTM gate initiates a browse from a specified queue, or from the start of the DCT.

Input parameters

START_AT
specifies a queue from which the browse should start.

Output parameters

BROWSE_TOKEN
is returned and uniquely identifies this browse session.

RESPONSE
is Transient Data’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
returned when RESPONSE is DISASTER. Possible values are: ABEND, DCT_ERROR, DIRECTORY_MGR_ERROR, and LOGIC_ERROR

TDTM gate, GET_NEXT_TDQDEF function
The GET_NEXT_TDQDEF function of the TDTM gate returns information about a queue as part of a browse operation.

Input parameters
BROWSE_TOKEN
identifies the browse session.

Output parameters
QUEUE_NAME
is the name of the queue.

[ATI_FACILITY]
specifies the facility associated with this intrapartition queue when a trigger transaction is attached. Possible values are: TERMINAL|FILE|SYSTEM

[ATI_TERMID]
specified together with the FACILITY option, FACILITY_ID identifies the facility that the trigger transaction should be associated with.

[ATI_TRANID]
specifies the ATI transaction to be invoked when the trigger level is reached.

[BUFFER_NUMBER]
specifies the number of buffers to be associated with an extrapartition queue.

[DDNAME]
specifies the DDNAME to be associated with an extrapartition queue.

[DISPOSITION]
specifies the disposition of the data set to be associated with an extrapartition queue. Possible values are: SHR|OLD|MOD

[DSNAME]
specifies the DSNAME of the data set to be associated with the extrapartition queue.

[EMPTY_STATUS]
indicates whether the queue contains any records, and whether the queue is full. This option applies to extrapartition queues only. Possible values are: FULL|EMPTY|NOTEMPTY

[ENABLE_STATUS]
indicates the status of the queue. Possible values are: ENABLED|DISABLING|DISABLED

[ERROR_OPTION]
specifies what action is to be taken in the event of an I/O error. This option applies to extrapartition queues only. Possible values are:
**Application domain (AP)**

**[INDIRECT_DEST]**

specifies the destination queue if this queue is an indirect queue.

**[WAIT]**

specifies whether this logically recoverable intrapartition queue can wait for the resolution of an indoubt failure. Possible values are:

YES | NO

**[WAIT_ACTION]**

specifies the action to be taken if this logically recoverable intrapartition queue suffers an indoubt failure. Possible values are:

QUEUE | REJECT

**[NUM_ITEMS]**

states the number of committed items in the queue.

**[OPEN_STATUS]**

indicates whether the queue is open. Possible values are:

OPEN | CLOSED

**[RECORD_FORMAT]**

specifies the format of the records held on the extrapartition queue. Possible values are:

FIXUNB | FIXUNBA | FIXUNBM | FIXBLK | FIXBLKA | FIXBLKM
VARBLK | VARBLKA | VARBLKM | VARUNB | VARUNBA | VARUNBM | UNDEFINED

**[RECORD_LENGTH]**

specifies the record length of the extrapartition queue.

**[RECOVERY]**

specifies the recovery type of an intrapartition queue. Possible values are:

NO | PH | LG

**[REMOTE_NAME]**

specifies the remote name of the queue if this is a remote queue definition.

**[REWIND]**

specifies where the tape is positioned in relation to the end of the data set. This input parameter applies to extrapartition queues only. Possible values are:

REREAD | LEAVE

**[TD_QUEUE_TOKEN]**

states which token is associated with this queue.

**[TD_TYPE]**

specifies the queue type. Possible values are:

EXTRA | INTRA | INDIRECT | REMOTE

**[TRIGGER_LEVEL]**

specifies the trigger level of the intrapartition queue.

**[TYPE_FILE]**

specifies whether this queue is:

- An input queue
- An output queue
- Whether it is a queue that is to be read backwards.

Possible values are:
[USERID_TOKEN]
indicates which token is associated with the USERID that was specified for this intrapartition queue.

[SYSOUTCLASS]
specifies the SYSOUT class to be used for the associated output extrapartition queue.

[BLOCK_LENGTH]
specifies the block length of an extrapartition queue.

RESPONSE
is Transient Data’s response to the call. It can have any of these values:
OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON]
returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, DCT_ERROR, DIRECTORY_MGR_ERROR, LOGIC_ERROR</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>NO_MORE_DATA_AVAILABLE</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_BROWSE_TOKEN</td>
</tr>
</tbody>
</table>

TDTM gate, END_BROWSE_TDQDEF function

The END_BROWSE_TDQDEF function of the TDTM gate terminates a browse session.

Input parameters
BROWSE_TOKEN
identifies the browse session.

Output parameters
RESPONSE
is Transient Data’s response to the call. It can have any of these values:
OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON]
returned when RESPONSE is DISASTER, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, DCT_ERROR, DIRECTORY_MGR_ERROR, LOGIC_ERROR</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_BROWSE_TOKEN</td>
</tr>
</tbody>
</table>

TDTM gate, SET_TDQUEUE function

The SET_TDQUEUE function of the TDTM gate updates attributes of an installed transient data queue.
Application domain (AP)

**Input parameters**

**QUEUE_NAME**  
identifies the queue to be updated.

**[ATI_FACILITY]**  
specifies the type of facility associated with this queue. Possible values are:  
TERMINAL FILE SYSTEM

**[ATI_TERMID]**  
indicates whether the ATI facility is to be updated.

**[ATI_TRANID]**  
indicates whether the ATI transaction is to be updated.

**[ATI_USERID]**  
indicates whether the USERID associated with the ATI transaction is to be updated.

**[USERID_TOKEN]**  
is the token that is supplied by the user domain when the userid is added to the system.

**Output parameters**

**OLD_USER_TOKEN**  
identifies the token associated with a previous USERID.

**RESPONSE**  
is Transient Data’s response to the call. It can have any of these values:  
OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

**[REASON]**  
returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, DCT_ERROR, DIRECTORY_MGR_ERROR, LOGIC_ERROR, CATALOG_WRITE_ERROR</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>IS_CXRF, NOT_CLOSED, DISABLE_PENDING, NOT_DISABLED, QUEUE_IS_INDOUBT, QUEUE_NOT_FOUND</td>
</tr>
</tbody>
</table>

**TDTM gate, DISCARD_TQDEF function**

The DISCARD_TQDEF function of the TDTM gate deletes an installed transient data queue definition and removes it from the catalog. A DELETEQ command is issued as part of the discard process.

**Input parameters**

**QUEUE_NAME**  
identifies the queue to be discarded.

**[TD_QUEUE_TOKEN]**  
can be specified instead of QUEUE_NAME. TD_QUEUE_TOKEN identifies the queue to be discarded.

**Output parameters**

**RESPONSE**  
is Transient Data’s response to the call. It can have any of these values:
TDTM gate, **COMMIT_TDQDEFS function**

The COMMIT_TDQDEFS function of the TDTM gate catalogs all installed transient data queue definitions as part of cold start processing.

**Input parameters**

**TOKEN**
specifies the catalog to which the queue definitions are to be written.

**Output parameters**

**RESPONSE**
is Transient Data’s response to the call. It can have any of these values:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>DCT_ERROR, DIRECTORY_MGR_ERROR, LOGIC_ERROR, CATALOG_DELETE_FAILED,</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>NAME_STARTS_WITH_C, NOT_CLOSED, NOT_DISABLED, DISABLE_PENDING, QUEUE_NOT_FOUND</td>
</tr>
</tbody>
</table>

**TDXM gate, **BIND_FACILITY** function**

The BIND_FACILITY function of the TDXM gate is used to associate a transaction with the definition for the transient data queue that caused the transaction to be trigger-level attached, where the principal facility is the queue itself (that is there is no terminal associated with the queue).

**Input parameters**

None.

**Output parameters**

**RESPONSE**
is Transient Data’s response to the call. It can have any of these values:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
</tbody>
</table>

**TDXM gate, **BIND_SECONDARY_FACILITY** function**

The BIND_SECONDARY_FACILITY function of the TDXM gate is used to associate a transaction with the definition for a transient data queue that has
caused the transaction to be trigger-level attached (where the principal facility is a terminal and the secondary facility is the transient data queue itself).

**Input parameters**
None.

**Output parameters**

**FACILITY_NAME**
- is the name of the transient data queue. The queue is the secondary facility and has been associated with this transaction.

**RESPONSE**
- is Transient Data’s response to the call. It can have any of the following values:
  - OK
  - EXCEPTION
  - DISASTER
  - INVALID
  - KERNERROR
  - PURGED

**REASON**
- is returned when RESPONSE is DISASTER. Possible value is ABEND.

### TDXM gate, RELEASE_FACILITY function

The RELEASE_FACILITY function of the TDXM gate is used to disassociate a transaction from the TD queue. (The principal facility type is either TERMINAL or TDQUEUE.)

**Input parameters**

**TERMINATION_TYPE**
- is the type of transaction termination. It can have either of these values:
  - NORMAL
  - ABNORMAL

[RESTART_REQUESTED]
- indicates whether or not the transaction is to be restarted. It can have either of these values:
  - YES
  - NO

**Output parameters**

**RESPONSE**
- is the domain’s response to the call. It can have any of these values:
  - OK
  - EXCEPTION
  - DISASTER
  - INVALID
  - KERNERROR
  - PURGED

**REASON**
- is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>RESTART_FAILURE</td>
</tr>
</tbody>
</table>

### TDXM gate, INQUIRE_FACILITY function

The INQUIRE_FACILITY function of the TDXM gate is used to inquire about the transient data facilities that support facility manager calls from the transaction manager domain.

**Input parameters**

[FACILITY_TOKEN]
- is the token identifying the transaction that has been trigger-level attached.
Output parameters

FACILITY_NAME
is the four-character name of the transaction that has been trigger-level
attached.

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

TFAL gate, ALLOCATE function
The ALLOCATE function of the TFAL gate is used to allocate a terminal for a
transaction.

Input parameters

REQUEST_ID
is the four-character transaction identifier initiating the attach.

[MODE_NAME]
is the eight-character mode-name of the terminal to be attached.

SYSTEM_TOKEN
is the token identifying the CICS region to which the terminal is to be
attached.

[PRIVILEGED]
indicates whether or not the terminal is to be attached as a privileged
terminal. It can have either of these values:
YES|NO

[NON_PURGEABLE]
indicates whether or not the terminal is to be purgeable. It can have either
of these values:
YES|NO

Output parameters

TERMINAL_TOKEN
is the token identifying the terminal that has been attached.

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID.
Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>GETMAIN_FAILED, LOGIC_ERROR</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>ALLOCATE_FAILURE, ALLOCATE_PURGED</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

TFAL gate, CANCEL_AID function
The CANCEL_AID function of the TFAL gate is used to cancel a
terminal-transaction AID.
Application domain (AP)

**Input parameters**

**TERMID**

is the four-character terminal identifier.

**TRANID**

is the four-character transaction identifier.

**TERM_OWNER_NETNAME**

is the APPLID of the CICS region that “owns” the terminal.

**Output parameters**

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

**[REASON]**

is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NOT_FOUND</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

**TFAL gate, CANCEL_AIDS_FOR_CONNECTION function**

The CANCEL_AIDS_FOR_CONNECTION function of the TFAL gate is used to cancel AIDs for the given CICS region.

**Input parameters**

**SYSTEM_TOKEN**

is the token identifying the CICS region.

**CALLER**

is the method used to call this function. It can have either of these values:

BUILDER | API

**FORCE**

indicates whether or not system AIDs are to be canceled. It can have either of these values:

YES | NO

**FACILITY**

indicates the facility type associated with the AIDs. It can have either of these values:

CONNECTION | TERMINAL

**Output parameters**

**[AIDS_CANCELLLED]**

indicates whether or not AIDs were canceled as a result of this request. It can have either of these values:

YES | NO

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED
TFAL gate, CANCEL_AIDS_FOR TERMINAL function

The CANCEL_AIDS_FOR TERMINAL function of the TFAL gate is used to cancel all AIDs for the given terminal.

Input parameters

**Note:** Specify either TERMID or TERMINAL_TOKEN, not both.

**TERMID**

is the four-character terminal identifier.

**TERMINAL_TOKEN**

is the token identifying the terminal.

**CALLER**

is the method used to call this function. It can have one of these values:

| BUILDER | API | BUILDER_REMDEL |

**FORCE**

indicates whether or not system AIDs are to be canceled. It can have either of these values:

| YES | NO |

**FACILITY**

indicates the facility type associated with the AIDs. It can have either of these values:

| CONNECTION | TERMINAL |

Output parameters

**[AIDS_CANCELLED]**

indicates whether or not AIDs were canceled as a result of this request. It can have either of these values:

| YES | NO |

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

| OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED |

**[REASON]**

is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NULL_TERMINAL_TOKEN,</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>
TFAL gate, CHECK_TRANID_IN_USE function

The CHECK_TRANID_IN_USE function of the TFAL gate is used to check whether any of the AID chains contain references to the given TRANID.

Input parameters

TRANID
is the four-character transaction identifier.

Output parameters

IN_USE
indicates whether or not the transaction identifier (specified by the TRANID parameter) is in use. It can have either of these values:
YES|NO

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

TFAL gate, DISCARD_AIDS function

The DISCARD_AIDS function of the TFAL gate is used to attach a task which will release start data and free the AIDs in the chain addressed by the AID_TOKEN.

Input parameters

AID_TOKEN
is the token identifying the chain of AIDs.

Output parameters

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

TFAL gate, FIND_TRANSACTION_OWNER function

The FIND_TRANSACTION_OWNER function of the TFAL gate is used to determine the CICS region that owns the given transaction (that is, at which the transaction instance originated).

Input parameters

TERMINAL_TOKEN
is the token identifying the terminal.
TRANID
is the four-character transaction identifier.

Output parameters
TRAN_OWNER_SYSID
is the four-character system identifier for the CICS region that owns the transaction instance.

RESPONSE
is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NOT_FOUND, TOR_LINK_NOT_ACTIVE,</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

TFAL gate, GET_MESSAGE function
The GET_MESSAGE function of the TFAL gate is used to get a message from a terminal.

Input parameters
TERMINAL_TOKEN
is the token identifying the terminal.

PREVIOUS_AID_TOKEN
is the AID token identifying the previous transaction that ran at this terminal.

Output parameters
AID_TOKEN
is the AID token identifying the current transaction for which the message was got.

TSQUEUE_NAME
is the eight-character name of the temporary storage queue name of the message whose BMS AID was found.

BMS_TITLE_PRESENT
indicates whether or not a BMS title is present on the terminal. It can have either of these values:
YES|NO

RESPONSE
is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NOT_FOUND</td>
</tr>
</tbody>
</table>
TFAL gate, INITIALIZE_AID_POINTERS function

The INITIALIZE_AID_POINTERS function of the TFAL gate is used to initialize the AID pointers for the given CICS region.

**Input parameters**

**SYSTEM_TOKEN**

is the token identifying the CICS region.

**Output parameters**

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON]

is returned when RESPONSE is INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

TFAL gate, INQUIRE_ALLOCATE_AID function

The INQUIRE_ALLOCATE_AID function of the TFAL gate is used to inquire about the AIDs allocated for the given CICS region.

**Input parameters**

**SYSTEM_TOKEN**

is the token identifying the CICS region.

[PRIVILEGED]

indicates whether or not to inquire only about privileged ISC type AIDs. It can have either of these values:

YES | NO

**Output parameters**

**EXISTS**

indicates whether or not the AID exists. It can have either of these values:

YES | NO

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON]

is returned when RESPONSE is INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>
TFAL gate, LOCATE_AID

The LOCATE_AID function of the TFAL gate is used for automatic transaction initiation to determine the AID for the specified terminal, and if found, to use the transaction identifier from the AID to attach the task.

Input parameters

TERMID
is the four-character terminal-identifier.

[TYPE]
denotes the type of AID to be located. It can have one of these values:
BMS | PUT | INT | TDP | ISC | REMDEL

Output parameters

[TRANID]
is the four-character transaction identifier associated with the specified terminal.

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NOT_FOUND</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

TFAL gate, LOCATE_REMDEL_AID

The LOCATE_REMDEL_AID function of the TFAL gate is used to determine the AID (for a delete remote TERMINAL definition request) for the specified system (SYSTEM_TOKEN specified) or after the given (PREVIOUS_AID_TOKEN specified).

Input parameters

SYSTEM_TOKEN
is the token identifying the CICS region.

PREVIOUS_AID_TOKEN
is the AID token identifying the previous transaction that ran at this terminal.

Output parameters

AID_TOKEN
is the AID token identifying the transaction to be deleted.

TARGET_SYSID
is the four-character system identifier for the target CICS system.

TERMID
is the four-character terminal identifier from the REMDEL AID.

TERM_OWNER_NETNAME
is the eight-character netname from the REMDEL AID.
RESPONSE
is the domain’s response to the call. It can have any of these values:
OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NOT_FOUND</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

**TFAL gate, LOCATE_SHIPPABLE_AID**
The LOCATE_SHIPPABLE_AID function of the TFAL gate is used to determine an AID (for a delete remote TERMINAL definition request or for a remote terminal request) to be shipped to the specified system.

**Input parameters**

SYSTEM_TOKEN
is the token identifying the CICS region.

**Output parameters**

AID_TOKEN
is the AID token identifying the transaction to be deleted.

LAST
Indicates that either:
- there is a single qualifying AID or all qualifying AIDs have the same AIDTRMID (YES), or
- in addition to the AID returned there are other qualifying AIDs (NO)

It can have either of these values:
YES | NO

**RESPONSE**
is the domain’s response to the call. It can have any of these values:
OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NOT_FOUND</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

**TFAL gate, MATCH_TASK_TO_AID function**
The MATCH_TASK_TO_AID function of the TFAL gate is used to inquire about AIDs for the given terminal and transaction.

**Input parameters**

TERMINAL_TOKEN
is the token identifying the terminal.
TRANID
is the four-character transaction identifier.

Output parameters

TDQUEUE_NAME
is the eight-character name of the transient data queue for the AID.

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NOT_FOUND, MATCHED_TERMID_ONLY</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

TFAL gate, PURGE_ALLOCATE_AIDS
The PURGE_ALLOCATE_AIDS function of the TFAL gate is used to delete purgeable allocate AIDs for a given connection after user exit XZIQUE in DFHZISP has issued return code 8 (delete all) or return code 12 (delete all for given modegroup).

Input parameters

SYSTEM_TOKEN
is the token identifying the CICS region.

[MODE_NAME]
The name of the modegroup. If this parameter is omitted, the default is all modegroups.

Output parameters

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

TFAL gate, RECOVER_START_DATA
The RECOVER_START_DATA function of the TFAL gate is used to retrieve a PUT-type AID stored in a DWE and rechain it onto the TCTSE in front of the first AID for the terminal.

Input parameters

AID_TOKEN
is the AID token identifying the transaction to be deleted.
Output parameters

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>NULL_SYSTEM_TOKEN, GETMAIN_FAILED</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

The REMOTE_DELETE function of the TFAL gate is used to chain a REMOTE DELETE (REMDEL) AID onto the system entry of the specified target CICS region. The REMDEL AID tells the target region to delete its shipped definition of the specified terminal.

Input parameters

TARGET_SYSID
is the four-character system identifier for the target CICS region.

TERMINAL_TOKEN
is the token identifying the terminal.

TERMID
is the four-character terminal identifier for the terminal associated with the transaction.

TERM_OWNER_NETNAME
Is the VTAM APPLID of the CICS region that “owns” the terminal.

Note: The terminal identifier can either be specified as TERMID and TERM_OWNER_NETNAME (where TERMID is the name known in the terminal owning system), or it can be specified by TERMINAL_TOKEN if the TCTTE address is known.

Output parameters

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>GETMAIN_FAILED,</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>TOR_LINK_NOT_ACTIVE</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>
TFAL gate, REMOVE_EXPIRED_AID

The REMOVE_EXPIRED_AID function of the TFAL gate is used to search all AID chains for a BMS AID that has yet to be initiated and which matches the eligibility parameters. Unchain the first such AID found, copy details from the AID into the caller’s parameter list, and freemain the AID.

Input parameters

[NORMAL_EXPIRY_TIME]

is the normal threshold time.

[ADJUSTED_EXPIRY_TIME]

is the adjusted threshold time.

[MSGID]

is the BMS message identifier

[LDC] is the logical device code

Note: If MSGID and LDC are specified, the expiry time is not checked.

Output parameters

TSQUEUE_NAME

is the eight-character name of the temporary storage queue name of the message whose BMS AID was found.

TRANID

is the four-character transaction identifier associated with the specified terminal.

TERMID

is the four-character terminal identifier for the terminal associated with the transaction.

RESPONSE

is the domain’s response to the call. It can have any of these values: OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]

is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NOT_FOUND</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

TFAL gate, REMOVE_EXPIRED_REMOTE_AID

The REMOVE_EXPIRED_REMOTE_AID function of the TFAL gate is used to search for an uninitiated remote AID which is older than the expiry time specified by the caller, unchain the AID, and cleanup any associated resources.

Input parameters

NORMAL_EXPIRY_TIME

is the normal threshold time.

ADJUSTED_EXPIRY_TIME

is the adjusted threshold time.
Application domain (AP)

Output parameters

TRANID
is the four-character transaction identifier associated with the specified terminal.

TERMID
is the four-character terminal identifier for the terminal associated with the transaction.

TERM_OWNER_SYSID
is the system identifier of the CICS region that “owns” the terminal.

SHIPPED
identifies whether the AID has been shipped. It can have either of these values:
YES|NO

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NOT_FOUND</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

TFAL gate, REMOVE_MESSAGE

The REMOVE_MESSAGE function of the TFAL gate is used to:
1. Find an uninitiated BMS AID for the specified terminal
2. Unchain and freemain the AID, provided that the AID security fields match those of the currently signed-on operator
3. Return the TS queue name from the AID.

Input parameters

TERMINAL_TOKEN
is the token identifying the terminal.

[MSGID]
is the BMS message identifier

Output parameters

TSQUEUE_NAME
is the eight-character name of the temporary storage queue name for the message whose BMS AID was found.

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:
### RESPONSE Possible REASON values

<table>
<thead>
<tr>
<th>REASON</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NOT_FOUND, SECURITY_MISMATCH</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

### TFAL gate, REMOVE_REMOTE_DELETES

The REMOVE_REMOTE_DELETES function of the TFAL gate is used to unchain and freemain all REMDEL AIDs from the AID chain of the specified system entry. Optional parameters TERMID and TERM_OWNER_NETNAME may be specified; in which case only those REMDEL AIDs which match the specified values are removed.

**Input parameters**

- **TARGET_SYSID**
  is the four-character system identifier for the target CICS region.

- **SYSTEM_TOKEN**
  is the token identifying the CICS region.

**Note:** Specify either the TARGET_SYSID parameter or the SYSTEM_TOKEN parameter, not both.

- **[TERMID]**
  is the four-character terminal identifier for the terminal associated with the transaction.

- **[TERM_OWNER_NETNAME]**
  is the netname of the region that “owns” the terminal.

**Output parameters**

- **RESPONSE**
  is the domain’s response to the call. It can have any of these values: OK, EXCEPTION, DISASTER, INVALID, KERNERROR, PURGED

- **[REASON]**
  is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NOT_FOUND</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

### TFAL gate, REROUTE_SHIPPABLE_AIDS

The REROUTE_SHIPPABLE_AIDS function of the TFAL gate is used to redirect AIDs for remote terminals from one remote system to another.

**Input parameters**

- **ORIGINAL_SYSTEM_TOKEN**
  is the token identifying the remote system which was the AIDs’ original target.

- **TARGET_SYSTEM_TOKEN**
  is the token identifying the remote system which is the AIDs’ new target.
TERMINAL_NETNAME

is the eight-character NETNAME which identifies the terminal whose AIDs are to be rerouted.

Output parameters

RESPONSE

is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON]

is returned when RESPONSE is DISASTER or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>LOGIC_ERROR</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

TFAL gate, RESCHEDULE_BMS

The RESCHEDULE_BMS function of the TFAL gate is used to build a BMS AID and chain it to the front of the AID queue.

Input parameters

TERMINAL_TOKEN

is the token identifying the terminal.

TRANID

is the four-character transaction identifier associated with the specified terminal.

TSQUEUE_NAME

is the eight-character name of the temporary storage queue name of the message whose BMS AID was found.

BMS_TIMESTAMP

Timestamp for BMS AID. Used to test if AID is older than specified EXPIRY_TIME.

[OPIDENT]

Identifies the operator

Note: You can specify either the OPIDENT parameter or the OPCLASS parameter, not both.

[OPCLASS]

Identifies the operator class.

Note: You can specify either the OPIDENT parameter or the OPCLASS parameter, not both.

[BMS_TITLE_PRESENT]

Indicates if title in message control record. You can specify either of these values:

YES | NO

Output parameters

RESPONSE

is the domain’s response to the call. It can have any of these values:
[REASON] is returned when RESPONSE is DISASTER or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>GETMAIN_FAILED</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

**TFAL gate, RESET_AID_QUEUE**

The RESET_AID_QUEUE function of the TFAL gate is used to:
1. Give ALP a chance to reset the AID queue when a transaction ends
2. Give ALP a chance to bid for the use of the terminal if ATI tasks are waiting.

**Input parameters**

TERMINAL_TOKEN is the token identifying the terminal.

**Output parameters**

RESPONSE is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON] is returned when RESPONSE is INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

**TFAL gate, RESTORE_FROM_KEYPOINT**

The RESTORE_FROM_KEYPOINT function of the TFAL gate is used to: reschedule a chain of AIDs that we restored from the catalog during CICS system initialization.

**Input parameters**

AID_TOKEN A token denoting the chain of AIDs which are to be rescheduled.

**Output parameters**

RESPONSE is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON] is returned when RESPONSE is INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>
The RETRIEVE_START_DATA function of the TFAL gate is used to return the AID address and temporary storage queue name associated with the start data for the specified transaction and terminal.

**Input parameters**

- **TERMINAL_TOKEN**
  - is the token identifying the terminal.

- **TRANID**
  - is the four-character transaction identifier associated with the specified terminal.

**Output parameters**

- **TSQUEUE_NAME**
  - is the eight-character name of the temporary storage queue name of the message whose BMS AID was found.

- **RESPONSE**
  - is the domain’s response to the call. It can have any of these values:
    - OK
    - EXCEPTION
    - DISASTER
    - INVALID
    - KERNERROR
    - PURGED

- **[REASON]**
  - is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NOT_FOUND</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

The SCHEDULE_BMS function of the TFAL gate is used to: schedule a BMS AID.

**Input parameters**

- **TERMID**
  - is the four-character terminal identifier for the terminal associated with the transaction.

- **TRANID**
  - is the four-character transaction identifier associated with the specified terminal.

- **TSQUEUE_NAME**
  - is the eight-character name of the temporary storage queue name of the message whose BMS AID was found.

- **BMS_TIMESTAMP**
  - is the timestamp for the BMS AID. This is used to test if the AID is older than its EXPIRY_TIME.

- **[OPIDENT]**
  - Identifies the operator.

**Note:** You can specify either the OPIDENT parameter or the OPCLASS parameter, not both.
[OPCLASS]
Identifies the operator class.

**Note:** You can specify either the OPIDENT parameter or the OPCLASS parameter, not both.

[BMS_TITLE_PRESENT]
Indicates if the title is in the message control record. You can specify either of these values:

- YES
- NO

[TERTIAL_NETNAME]
is the eight-character NETNAME which identifies the terminal whose AIDs are to be rerouted.

**Output parameters**

RESPONSE is the domain’s response to the call. It can have any of these values:

- OK
- EXCEPTION
- DISASTER
- INVALID
- KERNERROR
- PURGED

[REASON] is returned when RESPONSE is DISASTER or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>GETMAIN_FAILED</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

**TFAL gate, SCHEDULE_START**
The SCHEDULE_START function of the TFAL gate is used to schedule a PUT or INT type AID

**Input parameters**

TRANID is the four-character transaction identifier associated with the specified terminal.

TERMID is the four-character terminal identifier for the terminal associated with the transaction.

[TRAN_OWNER_SYSID] is the system identifier of the CICS region that “owns” the transaction.

[TERM_OWNER_SYSID] is the system identifier of the CICS region to which the request should be shipped.

**Note:** You can specify either the TERM_OWNER_SYSID parameter or TERM_OWNER_NETNAME parameter, not both.

[TERM_OWNER_NETNAME] is the system identifier of the CICS region to which the request should be shipped.

**Note:** You can specify either the TERM_OWNER_SYSID parameter or TERM_OWNER_NETNAME parameter, not both.
Application domain (AP)

[ROUTED_FROM_TERMID]
  is the four-character terminal identifier for the terminal from which a task
  was transaction-routed to issue this START request.

[SHIPPED_VIA_SESSID]
  is the identifier of the session via which this START request was function
  shipped.

[MODE_NAME]
  is the mode name to be used

[TSQUEUE_NAME]
  is the name of the temporary storage queue which contains the data
  associated with the START request.

[FEPI]  indicates that this is a FEPI START request. It can have either of these
  values:
  YES|NO

[RECOVERABLE_DATA]
  indicates that the request is associated with recoverable data. It can have
  either of these values:
  YES|NO

[IN_DOUBT]
  indicates that the Unit of Work making the request is in doubt, and the
  request should not be scheduled until the Unit of Work is committed. It
  can have either of these values:
  YES|NO

[TERMINAL_NETNAME]
  is the eight-character NETNAME of the terminal associated with the
  transaction.

[SHIPPED_VIA_SYSID]
  identifies the connection via which this request was function shipped or
  transaction routed.

[TOR_NETNAME]
  is the netname of the CICS region that owns the terminal.

Output parameters

RESPONSE
  is the domain’s response to the call. It can have any of these values:
  OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
  is returned when RESPONSE is DISASTER or INVALID. Possible values
  are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>GETMAIN_FAILED</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

TFAL gate, SCHEDULE_TDP

The SCHEDULE_TDP function of the TFAL gate is used to schedule a TDP type
AID.
Input parameters

TRANID  
is the four-character transaction identifier associated with the specified terminal.

TERMID  
is the four-character terminal identifier for the terminal associated with the transaction.

TDQUEUE_NAME  
is the destination identifier for the TD queue.

[TERMINAL_NETNAME]  
is the eight-character NETNAME of the terminal associated with the transaction.

Output parameters

AID_TOKEN  
is the AID token identifying the transaction to be deleted.

RESPONSE  
is the domain’s response to the call. It can have any of these values:  
OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON]  
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>GETMAIN_FAILED</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>UNKNOWN_TRANID</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

TFAL gate, SLOWDOWN_PURGE

The SLOWDOWN_PURGE function of the TFAL gate is used to:
1. Search the specified system entry’s AID chain for the first allocate-type AID associated with a stall-purgeable task
2. Cancel the identified transaction.

Input parameters

SYSTEM_TOKEN  
is the four-character terminal identifier for the terminal associated with the transaction.

Output parameters

RESPONSE  
is the domain’s response to the call. It can have any of these values:  
OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON]  
is returned when RESPONSE is EXCEPTION, or INVALID. Possible values are:
**Application domain (AP)**

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NOT_FOUND</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

**TFAL gate, TAKE_KEYPOINT**

The TAKE_KEYPOINT function of the TFAL gate is used to return a chain of AIDs which are to be written to the global catalog.

**Input parameters**
None.

**Output parameters**

**AID_TOKEN**

is the token identifying the chain of AIDs.

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

**[REASON]**

is returned when RESPONSE is INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

**TFAL gate, TERMAVAILABLE_FOR_QUEUE**

The TERMAVAILABLE_FOR_QUEUE function of the TFAL gate is used, when a terminal becomes available for allocation, to give DFHALP the chance to attach or resume a task which requires this terminal.

**Input parameters**

**TERMINAL_TOKEN**

is the token identifying the terminal.

**Output parameters**

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

**[REASON]**

is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>GETMAIN_FAILED ATTACH_ERROR</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>NOT_FOUND</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

**TFAL gate, TERMINAL_NOW_UNAVAILABLE**

The TERMINAL_NOW_UNAVAILABLE function of the TFAL gate is used to perform required actions when a terminal or connection becomes unavailable.
Input parameters
TERMINAL_TOKEN
is the token identifying the terminal.

Output parameters
RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

TFAL gate, UNCHAIN_AID
The UNCHAIN_AID function of the TFAL gate is used to unchain and optionally freemain the specified AID.

Input parameters
AID_TOKEN
is the AID token identifying the transaction to be deleted.

FREEMAIN
indicates whether freemain is wanted. It can have either of these values:
YES|NO

Output parameters
RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

TFAL gate, UPDATE_TRANNUM_FOR_RESTART
The UPDATE_TRANNUM_FOR_RESTART function of the TFAL gate is used to update the AID’s TRANNUM to that of the restarted task.

Input parameters
TERMINAL_TOKEN
is the token identifying the terminal.

ORIGINAL_TRANNUM
is the TRANNUM set in the AID when original task was attached.

NEW_TRANNUM
is the new TRANNUM to be set in the AID.
Output parameters

RESPONSE is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NULL_TERMINAL_TOKEN</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

TFBF gate, BIND_FACILITY function

The BIND_FACILITY function of the TFBF gate is used to associate a transaction with the terminal.

Input parameters

[PROFILE] is the eight-character name of the profile to be used to associate the transaction and terminal.

[PARTITIONSET_NAME] is the eight-character name of a partition set. This parameter is used only of the value of PARTITIONSET is NAME.

[PARTITIONSET] indicates if a partition set is to be used for the terminal facility. It can have any of these values:
NONE|NAME|OWN|KEEP

Output parameters

RESPONSE is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, REMOTE_SCHEDULE_FAILURE, SECURITY_FAILURE, TABLE_MANAGER_FAILURE</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>NO_TERMINAL, TRANSACTION_ABEND</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

TFIQ gate, INQUIRE_TERMINAL_FACILITY function

The INQUIRE_TERMINAL_FACILITY function of the TFIQ gate is used to inquire about attributes of a named terminal facility.
Input parameters

Note: Specify a value for either the TRANSACTION_TOKEN or TERMINAL_TOKEN parameter, not both.

[TRANSACTION_TOKEN]
  is a token identifying a transaction for which you want to inquire about the associated terminal.

[TTERMINAL_TOKEN]
  is a token identifying a terminal.

Output parameters

[FACILITY_NAME]
  is the four-character name of the terminal facility.

[NETNAME]
  is the eight-character netname of the terminal facility.

[PSEUDO_CONV_COMMAREA]
  is a block into which the communications area for a pseudo-conversational transaction is copied.

[TERMINAL_TRAFFIC_READ]
  indicates whether or not reading is supported. It can have either of these values:
  YES|NO

[TERMINAL_TRAFFIC_WRITE]
  indicates whether or not writing is supported. It can have either of these values:
  YES|NO

[TERMINAL_USER_AREA]
  is a block into which the terminal user area is copied.

[NATIONAL_LANGUAGE_IN_USE]
  is the three-character code indicating the national language in use for the terminal facility. (See Table 93 on page 102.)

[INSPECT_DATA]
  is a token indicating the LE/370 runtime options for the terminal facility.

[STORAGE_FREEZE]
  indicates whether or not storage normally freed during the processing of a transaction for the terminal facility is to be frozen. (The frozen storage is not freed until the end of the transaction.) It can have either of these values:
  YES|NO

RESPONSE
  is the domain’s response to the call. It can have any of these values:
  OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
  is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NO_TERMINAL</td>
</tr>
</tbody>
</table>
RESPONSE | Possible REASON values
--- | ---
INVALID | INVALID_TERMINAL_TYPE

**TFIQ gate, SET_TERMINAL_FACILITY function**

The SET_TERMINAL_FACILITY function of the TFIQ gate is used to set attributes of a named terminal facility.

**Input parameters**

**Note:** Specify a value for either the TRANSACTION_TOKEN or TERMINAL_TOKEN parameter, not both.

**[TRANSACTION_TOKEN]**

is a token identifying a transaction for which you want to inquire about the associated terminal.

**[TERMINAL_TOKEN]**

is a token identifying a terminal.

**[COUNT_STORAGE_VIOLATION]**

indicates whether or not storage violations are to be counted for this terminal facility. It can have either of these values:

- YES
- NO

**[INPUTMSG]**

is a block into which the input message for a pseudo-conversational transaction is copied.

**[PSEUDO_CONV_NEXT_TRANSID]**

is the four-character identifier of the transaction to which control is passed on a normal return from a pseudo-conversational transaction (to which the pseudo_conversational data is passed).

**[PSEUDO_CONV_COMMAREA]**

is a block into which the communications area for a pseudo-conversational transaction is copied.

**[PSEUDO_CONV_IMMEDIATE]**

is the four-character identifier of the transaction to which control is passed on an immediate return from a pseudo-conversational transaction (to which the pseudo_conversational data is passed).

**[NATIONAL_LANGUAGE_IN_USE]**

is the three-character code indicating the national language in use for the terminal facility. (See Table 93 on page 1023.)

**[INSPECT_DATA]**

is a token indicating the LE/370 runtime options for the terminal facility.

**[STORAGE_FREEZE]**

indicates whether or not storage normally freed during the processing of a transaction for the terminal facility is to be frozen. (The frozen storage is not freed until the end of the transaction.) It can have either of these values:

- YES
- NO
Output parameters

RESPONSE is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NO_TERMINAL, PERMANENT_TRANSID</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_TERMINAL_TYPE</td>
</tr>
</tbody>
</table>

TFIQ gate, INQUIRE_MONITOR_DATA function

The INQUIRE_MONITOR_DATA function of the TFIQ gate is used to inquire about monitoring data of the terminal facility.

Input parameters

None.

Output parameters

[FACILITY_TYPE] indicates the type of terminal facility. It can have any of these values:
LU61|LU62|IRC|IRC_XCF|OTHER

[FACILITY_NAME] is the four-character name of the terminal facility.

[NETNAME] is the eight-character netname of the terminal facility.

[INPUT_MSG_LENGTH] is the length (in bytes) of the input message for the terminal facility.

[SERVICE_REPORTING_CLASS] is a token indicating the service reporting class for the terminal facility (for MVS workload manager purposes).

RESPONSE is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NO_TERMINAL</td>
</tr>
</tbody>
</table>
**Application domain (AP)**

### Application domain’s generic gates

Table 4 summarizes the application domain’s generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

**Table 4. Application domain’s generic gates**

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>APDM</td>
<td>AP 0900</td>
<td>INITIALISE_DOMAIN</td>
<td>DMDM</td>
</tr>
<tr>
<td></td>
<td>AP 0901</td>
<td>QUIESCE_DOMAIN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TERMINATE_DOMAIN</td>
<td></td>
</tr>
<tr>
<td>APDS</td>
<td>AP 0500</td>
<td>TASK_REPLY</td>
<td>DSAT</td>
</tr>
<tr>
<td></td>
<td>AP 0501</td>
<td>PURGE_INHIBIT_QUERY</td>
<td></td>
</tr>
<tr>
<td>APST</td>
<td>AP D400</td>
<td>COLLECT_STATISTICS</td>
<td>STST</td>
</tr>
<tr>
<td></td>
<td>AP D401</td>
<td>COLLECTRESOURCE_STATS</td>
<td></td>
</tr>
<tr>
<td>APTI</td>
<td>AP F110</td>
<td>STORAGE_NOTIFY</td>
<td>SMNT</td>
</tr>
<tr>
<td></td>
<td>AP F111</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AP F300</td>
<td>NOTIFY</td>
<td>TISR</td>
</tr>
<tr>
<td></td>
<td>AP F301</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For descriptions of these functions and their input and output parameters, refer to the sections dealing with the corresponding generic formats, as follows:

**Functions and parameters**

- Format DMDM—“Domain manager domain’s generic formats” on page 377
- Format DSAT—“Dispatcher domain’s generic formats” on page 324
- Format STST—“Statistics domain’s generic format” on page 1087
- Format SMNT—“Storage manager domain’s generic format” on page 1116
- Format TISR—“Timer domain’s generic format” on page 1228

### Application domain’s generic formats

Table 5 describes the generic formats owned by the application domain and shows the functions performed on the calls.

**Table 5. Generic formats owned by application domain**

<table>
<thead>
<tr>
<th>Format</th>
<th>Calling module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>APUE</td>
<td>DFHUEM</td>
<td>SET_EXIT_STATUS</td>
</tr>
</tbody>
</table>

In the descriptions of the formats that follow, the “input” parameters are input not to the application domain, but to the domain being called by the application domain. Similarly, the “output” parameters are output by the domain that was called by the application domain, in response to the call.

**APUE format, SET_EXIT_STATUS function**

The SET_EXIT_STATUS function of the APUE format is used to set the exit status at a specified exit point.
Input parameters

EXIT_POINT
is the name of the exit to be enabled or disabled.

EXIT_STATUS (ACTIVE|INACTIVE)
indicates whether the exit is to be made active or inactive.

Output parameters

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION, INVALID_EXIT_POINT</td>
</tr>
</tbody>
</table>

Control blocks

The main CICS control block in the AP domain is the common system area (CSA), which exists from CICS system initialization time until CICS is closed down. The CSA contains:
- Register save area
- Pointers to the CICS control modules
- Control information
- System constants
- Time-control storage
- Work area for statistics
- Task abnormal termination interface
- Pointers to CICS system tables.

Figure 3 on page 102 shows the main fields in the CSA.
The CSA has an extension area known as the CSA optional features list. The address of the optional features list is held in CSAOPFLA in the CSA, and also in TCACSOAD in the TCA.

Figure 4 shows the main fields in the optional features list.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X'4C'</td>
<td>CSACDTA</td>
</tr>
<tr>
<td></td>
<td>Address of currently dispatched task</td>
</tr>
<tr>
<td>X'54'</td>
<td>CSAICEBA</td>
</tr>
<tr>
<td></td>
<td>Address of interval control element chain</td>
</tr>
<tr>
<td>X'8C'</td>
<td>CSASITBA</td>
</tr>
<tr>
<td></td>
<td>Address of system initialization table (SIT)</td>
</tr>
<tr>
<td>X'CB'</td>
<td>CSAOPFLA</td>
</tr>
<tr>
<td></td>
<td>Address of CSA optional features list</td>
</tr>
<tr>
<td>X'12B'</td>
<td>CSATCTBA</td>
</tr>
<tr>
<td></td>
<td>Address of terminal control table</td>
</tr>
<tr>
<td>X'130'</td>
<td>CSADCTBA</td>
</tr>
<tr>
<td></td>
<td>Address of destination control table</td>
</tr>
<tr>
<td>X'13C'</td>
<td>CSAQCAA</td>
</tr>
<tr>
<td></td>
<td>Address of queue control area</td>
</tr>
</tbody>
</table>

Figure 3. Main fields of the Common system area (CSA)

The CSA has an extension area known as the CSA optional features list. The address of the optional features list is held in CSAOPFLA in the CSA, and also in TCACSOAD in the TCA.

Figure 4 shows the main fields in the optional features list.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X'24'</td>
<td>CSASRTBA</td>
</tr>
<tr>
<td></td>
<td>Address of system recovery table</td>
</tr>
<tr>
<td>X'3C'</td>
<td>CSATSTBA</td>
</tr>
<tr>
<td></td>
<td>Address of temporary storage table</td>
</tr>
<tr>
<td>X'5C'</td>
<td>CSASRAA</td>
</tr>
<tr>
<td></td>
<td>Address of SRB control area</td>
</tr>
<tr>
<td>X'7C'</td>
<td>CSACSAAD</td>
</tr>
<tr>
<td></td>
<td>Address of CSA</td>
</tr>
</tbody>
</table>

Figure 4. Main fields of the CSA optional features list (CSAOPFL)

See the CICS Data Areas manual for a detailed description of these control blocks.
There is also a user-defined work area, called the common work area (CWA). The user can govern the length and storage key of the CWA by using the WRKAREA and CWAKEY system initialization parameters.

The CWA is available to any task while it has control of the system (that is, for operations performed between requests to CICS).

### Modules

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHAPDM</td>
<td>AP domain/domain manager gate service module. Handles the following calls</td>
</tr>
<tr>
<td></td>
<td>made by the domain manager to the AP domain:</td>
</tr>
<tr>
<td></td>
<td>INITIALISE_DOMAIN</td>
</tr>
<tr>
<td></td>
<td>QUIESCE_DOMAIN</td>
</tr>
<tr>
<td></td>
<td>TERMINATE_DOMAIN</td>
</tr>
<tr>
<td>DFHAPEX</td>
<td>AP domain user exit service module. This module handles INVOKE_USER_EXIT</td>
</tr>
<tr>
<td></td>
<td>made by several domains to the AP domain.</td>
</tr>
<tr>
<td>DFHAPID</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>PROFILE</td>
</tr>
<tr>
<td></td>
<td>QUERY_NETNAME</td>
</tr>
<tr>
<td>DFHAPIQ</td>
<td>AP domain task data inquire and set gate service module. Handles the</td>
</tr>
<tr>
<td></td>
<td>following call to the AP domain:</td>
</tr>
<tr>
<td></td>
<td>INQ_APPLICATION_DATA</td>
</tr>
<tr>
<td>DFHAPJC</td>
<td>AP domain/journal gate service module. This module handles WRITE_JOURNAL</td>
</tr>
<tr>
<td></td>
<td>_DATA calls made by the user exits’ XPI.</td>
</tr>
<tr>
<td>DFHAPSM</td>
<td>AP domain storage notify gate service module.</td>
</tr>
<tr>
<td>DFHAPST</td>
<td>AP domain functional gate for statistics. This module accepts a request</td>
</tr>
<tr>
<td></td>
<td>for and then supervises the copying and resetting of statistics counters</td>
</tr>
<tr>
<td></td>
<td>in the AP domain by calling the appropriate DFHSTxx modules to</td>
</tr>
<tr>
<td></td>
<td>access the counters.</td>
</tr>
<tr>
<td>DFHAPTC</td>
<td>AP domain module DFHAPTC provides a common mechanism for issuing TC</td>
</tr>
<tr>
<td></td>
<td>macro calls. It issues the following calls:</td>
</tr>
<tr>
<td></td>
<td>CANCEL</td>
</tr>
<tr>
<td></td>
<td>CLOSE</td>
</tr>
<tr>
<td></td>
<td>EXTRACT_PROCESS</td>
</tr>
<tr>
<td></td>
<td>ISSUE_NOTIFY</td>
</tr>
<tr>
<td></td>
<td>LISTEN</td>
</tr>
<tr>
<td></td>
<td>OPEN</td>
</tr>
<tr>
<td></td>
<td>RECEIVE</td>
</tr>
<tr>
<td></td>
<td>SEND</td>
</tr>
<tr>
<td></td>
<td>SET_SESSION</td>
</tr>
<tr>
<td>DFHAPTI</td>
<td>AP domain timer domain gate service module. This module handles NOTIFY</td>
</tr>
<tr>
<td></td>
<td>calls made by the timer domain to the AP domain.</td>
</tr>
<tr>
<td>DFHAPTIM</td>
<td>CICS interval control midnight task. This module deals with NOTIFY</td>
</tr>
<tr>
<td></td>
<td>requests from the timer domain.</td>
</tr>
<tr>
<td>DFHAPTIX</td>
<td>CICS expiry analysis task. This module deals with NOTIFY requests from</td>
</tr>
<tr>
<td></td>
<td>the timer domain.</td>
</tr>
<tr>
<td>DFHAPXM</td>
<td>AP domain/transaction manager gate service module. Handles the following</td>
</tr>
<tr>
<td></td>
<td>calls made by the transaction manager to the AP domain:</td>
</tr>
<tr>
<td></td>
<td>TRANSACTION_INITIALIZATION</td>
</tr>
<tr>
<td></td>
<td>RMI_START_OF_TASK</td>
</tr>
<tr>
<td></td>
<td>TRANSACTION_TERMINATION</td>
</tr>
</tbody>
</table>
Application domain (AP)

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHICXM</td>
<td>AP domain/interval control principal facility management gate service module. Handles the following calls made by the transaction manager to the AP domain: BIND_FACILITY, RELEASE_FACILITY, INQUIRE_FACILITY</td>
</tr>
<tr>
<td>DFHSAIQ</td>
<td>AP domain system data inquire and set gate service module. Handles the following calls to the AP domain: INQUIRE_SYSTEM, SET_SYSTEM</td>
</tr>
<tr>
<td>DFHSRP</td>
<td>Default system recovery program for the AP domain. It includes the ABAB functions. For more information about DFHSRP, see &quot;Chapter 86: System recovery program&quot; on page 1139.</td>
</tr>
<tr>
<td>DFHTDXM</td>
<td>AP domain/transient data principal facility management gate service module. Handles the following calls made by the transaction manager to the AP domain: BIND_FACILITY, BIND_SECONDARY_FACILITY, RELEASE_FACILITY, INQUIRE_FACILITY</td>
</tr>
<tr>
<td>DFHTFBF</td>
<td>AP domain/terminal facility manager bind facility gate service module. Handles the following call made by the terminal facility manager to the AP domain: BIND_FACILITY</td>
</tr>
<tr>
<td>DFHTFIQ</td>
<td>AP domain/terminal facility manager inquire and set gate service module. Handles the following calls made by the terminal facility manager to the AP domain: INQUIRE_TERMINAL_FACILITY, INQUIRE_MONITOR_DATA, SET_TERMINAL_FACILITY</td>
</tr>
<tr>
<td>DFHTFRF</td>
<td>AP domain/terminal facility manager release facility gate service module. Handles the following calls made by the terminal facility manager to the AP domain: RELEASE_FACILITY</td>
</tr>
</tbody>
</table>

Exits

Various global user exit points are provided for this domain, and these are described under the appropriate functions in the rest of this book.

Trace

Various trace point IDs are provided for this domain, and these are described under the appropriate functions in the rest of this book.

For more information about the trace points, see the CICS Trace Entries. For more information about using traces in problem determination, see the CICS Problem Determination Guide.
Chapter 5. AP domain initialization program

The AP domain initialization program is resident only long enough to start up the AP domain.

Modules

The main initialization program is DFHAPSIP. DFHAPSIP calls a series of modules DFHSIA1, DFHSIB1, ..., DFHSIJ1, which complete initialization. DFHAPSIP receives control from DFHAPDM. For further information about DFHAPDM, see page Modules on page 103.

Exits

No global user exit points are provided for this function.

Trace

The following point ID is provided for this function:

• AP 0700 (DFHSII1 add gate), for which the trace level is Exc.

For more information about the trace points, see the CICS Trace Entries. For more information about using traces in problem determination, see the CICS Problem Determination Guide.
Chapter 6. AP domain KC subcomponent

The AP domain KC subcomponent does the following:

- Provides an enqueue facility
- Manages profile definitions (making use of table manager program (see "FEPI Resource Manager work queues" on page 661)).
- Converts some DFHKC macro calls into dispatcher domain calls and transaction manager domain calls.

Design overview

This section describes the macro calls supported by the AP domain KC subcomponent.

DFHKC macro calls

ATTACH.
This call is converted into a transaction manager domain XMAT ATTACH call to create an instance of the requested transaction. This request is only used to create CICS system transactions and may not be used to attach a user transaction.

DEQ.
DEQ is used to reduce the use count of a resource previously enqueued on by this transaction. If the use count reaches zero, the resource is freed for use by another transaction. The NQED DEQUEUE service of the NQ domain is used for this function.

ENQ.
The caller passes a resource name or address. The AP domain KC subcomponent issues an NQED ENQUEUE request to the NQ domain.

INITIALIZE.
INITIALIZE is used during CICS initialization to tell the AP domain KC subcomponent to build profile table entries in storage.

PROFBROWSE.
This is used to browse profile table (PFT) entries.

PROFLOC.
This finds the profile table (PFT) entry for the profile ID passed.

REPLACE.
This replaces an existing profile table entry by a new version.

RESUME.
This call is converted into a dispatcher domain DSSR RESUME call to resume the suspended task.

WAIT.
Wait calls are converted into the appropriate dispatcher domain call.

WAITINIT.
This is used once during initialization to wait for the completion of an earlier INITIALIZE call.
AP domain KC subcomponent

Control blocks

Static storage area (SSA).
The AP domain KC subcomponent uses an SSA as a permanent work area. Field SSAKCP in the static storage area address list (as defined by the DSECT DFHSSADS) points to the AP domain KC subcomponents static storage area. The address of the static storage area address list is held in field CSASSA in the CSA optional features list.

See the CICS Data Areas manual for a detailed description of these control blocks.

Modules

The following are link-edited together to form the DFHKCP module:

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHKCP</td>
<td>This is a startup routine that passes control to either DFHXCP or DFHXCP.</td>
</tr>
<tr>
<td>DFHXCP</td>
<td>Processes DFHKC ATTACH, RESUME, and WAIT macro calls to the transaction manager and dispatcher and handles the DFHKC PROFLOC AND PROFBROWSE (profile locate and profile browse) services.</td>
</tr>
<tr>
<td>DFHXCPC</td>
<td>Processes DFHKC DEQ and ENQ macro calls to the AP domain KC subcomponent.</td>
</tr>
<tr>
<td></td>
<td>Receives DFHKC INITIALIZE, REPLACE, WAITINIT, and DISCARD macro calls to the transaction manager and passes them on to DFHKCQ.</td>
</tr>
<tr>
<td>DFHKCQ</td>
<td>Processes DFHKC INITIALIZE, REPLACE, WAITINIT, and DISCARD macro calls to the AP domain KC subcomponent.</td>
</tr>
<tr>
<td>DFHKCSC</td>
<td>Provides chain scanning facilities for the DISCARD TRANSACTION command.</td>
</tr>
</tbody>
</table>

Exits

There are two global user exit points in DFHEKC: XNQEREQ and XNQEREQC. See the CICS Customization Guide for further information.

Trace

The following point ID is provided for the AP domain KC subcomponent

- AP F0xx, for which the trace levels are AP 1, AP 2, and Exc.

For more information about the trace points, see the CICS Trace Entries. For more information about using traces in problem determination, see the CICS Problem Determination Guide.

Dumps

F007 DFHXCP was called to process a AP domain KC subcomponent request but did not recognize the function code in the TCA.
External interfaces

The AP domain KC subcomponent calls the following domains: DS, GC, KE, ME, MN, NQ, SM, TR and XM.

The AP domain KC subcomponent calls the following CICS AP domain function:
• Table manager

Statistics

No statistics are created by the AP domain KC subcomponent
Chapter 7. AP domain termination program

The AP domain (system) termination program (DFHSTP) provides for an orderly shutdown of CICS. When an PERFORM SHUTDOWN or PERFORM TAKEOVER command is used, either on the CEMT transaction or by an EXEC CICS command, the DFHEIPSH program invokes DFHSTP to handle it.

Design overview

Figure 5 shows the relationships between the components of AP domain termination.

Notes:

1. When a PERFORM SHUTDOWN or PERFORM TAKEOVER command is used, either on the CEMT transaction or by an EXEC CICS command, the DFHEIPSH program:
   - Loads the transaction list table (XLT) and program list table (PLT) from the DFHRPL DD concatenation
   - Transfers control to DFHSTP by means of a DFHPGXE PREPARE_XCTL_EXEC domain call.

For an immediate shutdown, statistics are collected at the step described by 1. Following this, the resource managers and the subsystem interface are terminated; no load of tables, terminal quiescing, or execution of programs specified in the PLT occurs, that is to say the steps described in notes 111, 12 on page 112, 13 on page 112, and 14 on page 112 are not performed on an immediate shutdown. Also, CICS files are not closed during step 15 on page 113 on an immediate shutdown.
2. Terminal activity is quiesced via an indicator in the CSA. This tells terminal control not to attach any transactions other than those specified in the XLT and those specifying SHUTDOWN(ENABLED) in their associated TRANSACTION resource definitions. The termination task logically disconnects itself from the physical terminal to allow other activity on that terminal.

3. The termination task allows all other tasks (except any journal tasks) to complete before linking to the first program specified in the first portion of the PLT.

4. When all programs in the first portion of the PLT have executed, terminal activity is quiesced completely, using bit CSATQIM in CSASSI2 in the CSA. If monitoring is running, it is stopped. The ICE and AID chains are broken (addresses saved in the TWA), the IRC session is quiesced, and the programs specified in the second portion of the PLT are executed.

5. All open files managed by CICS file control are closed by the file control shutdown program, DFHFCSD; temporary-storage control, DFHTSP is requested to output its buffer; and a keypoint is taken by the warm keypoint program, DFHWKP.

6. Control is returned to the operating system, with or without a dump (depending upon the parameters specified in the shutdown request causing termination).
   For the high-performance option (HPO), the service request block (SRB) in the system queue area (SQA) is freed by using a CICS SVC (DFHCSVC).

---

**Modules**

DFHSTP

---

**Exits**

There is one global user exit point in DFHSTP: XSTERM. See the CICS Customization Guide for further information.

---

**Trace**

No trace points are provided for this function.
Chapter 8. Autoinstall for terminals, consoles and APPC connections

Autoinstall for terminals provides the ability to log on to CICS from a logical unit (LU), known to VTAM but not previously defined to CICS, and to make a connection to a running CICS system.

A new connection is created and installed automatically if autoinstall for connections is enabled, and either of the following occurs:

- An APPC BIND request or CINIT request is received for an APPC service manager (SNASVCMG) session that does not have a matching CICS CONNECTION definition
- A BIND is received for a single session that does not have a matching CICS CONNECTION definition.

A new console is created and installed automatically if autoinstall for consoles is enabled and a CIB (Command Input Buffer sent from MVS) is received by CICS (DFHZCNA) and the console TCTTE does not already exist.

For an introduction to autoinstall, and information about how to implement it, see the CICS Resource Definition Guide.

The CICS Customization Guide gives information about implementing the autoinstall user program. The CICS-supplied programs are:

- DFHZATDX, which provides autoinstall for terminals only
- DFHZATDY, which provides autoinstall for terminals and APPC connections.

These programs are user-replaceable, because you may need to tailor the basic function to suit your CICS environment.

Design overview

Before a VTAM device can communicate with CICS, a VTAM session must be established between the device and CICS. The sequence of operations is LOGON, Open Destination (OPNDST), and Start Data Traffic (SDT). CICS can also initiate the LOGON by using a SIMLOGON.

The session can be requested by:

- Specifying AUTOCONNECT when the terminal is defined to CICS
- A VTAM master terminal command requesting a LOGON to CICS for a given terminal; for example, V NET;LOGON=CICSA, ID=L3277C1
- An individual terminal operator issuing a LOGON request (LOGON APPLID(CICSA))
- A CICS master terminal command requesting LOGON for a given terminal (CEMT SET TERMINAL(xxxx) INSERVICE ACQUIRED)
- CICS internally requesting a LOGON; for example, to process an ATI request
- LOGAPPL=CICS in the LU statement.

Consoles are not VTAM resource but they use a similar mechanism to autoinstall the TCTTE.
Autoinstall of a terminal logon flow

This section describes the flow of control for a terminal that is to be logged on by autoinstall.

1. When a terminal or single session APPC device attempts to log on, VTAM drives the logon exit. The CICS logon exit is DFHZLGX (load module DFHZCY).

In the following circumstances, an LU is a candidate for autoinstall:

- If it is not already defined to CICS (using RDO)
- If neither CICS nor VTAM is quiescing
- If the autoinstall user program (specified by the AIEXIT system initialization parameter) exists
- If the VTAM RPL is present
- If it is not an LU6.1 session or an LU6.2 parallel session
- If it is an LU6.2 single session terminal and the ISC=YES system initialization parameter is specified
- If the maximum number of concurrent logon requests (specified by the AIQMAX system initialization parameter) has not been exceeded.

DFHZLGX searches for the terminal in the terminal control table (TCT) by comparing the NETNAME passed by VTAM with the NETNAME found in the NIB descriptor for each installed terminal.

If a match is not found and AUTOINSTALL is enabled (TCTVADEN is set), CICS verifies that the terminal is eligible for autoinstall. Processing then consists of:

- Building an autoinstall work element (AWE) by issuing an MVS GETMAIN for subpool 1
- Copying the CINIT RU into the AWE
- Adding the AWE to the end of the AWE chain, which is chained from the TCT prefix.

If a match is found showing that this autoinstall terminal already exists, a postponed work element (PWE) is created and the terminal is reinstalled after deletion of the TCTTE (TCTEDZIP is ON) or if AILDELAY=0. If, however, AILDELAY<>0 but TCTEDZIP is not ON (that is, the TCTTE deletion is pending), the TCTTE is reused after cleanup.

2. Later, the work element (AWE) is actioned by DFHZACT attaching transaction CATA. For every AWE on the AWE chain, the DFHZATA autoinstall program is dispatched, passing to DFHZATA the AWE’s address.

3. The DFHZATA program:

   a. Validates the BIND image in the CINIT RU. If the image is not valid, issue message DFHZC6901.

   b. If VTAM Model Terminal Support (MTS) is being used (ACF/VTAM 3.3 or later), and the name of a CICS model has been supplied in a X'2F' MTS control vector, DFHZATA checks that the model exists by using the AIIQ subroutine interface of the AITM manager (see “Chapter 9, Autoinstall [terminal model manager] on page 125”). If the model does not exist, issue message DFHZC6936.

DFHZATA compares the BIND image contained in the MTS model with the BIND image passed in the CINIT RU. If there is a mismatch, issue message DFHZC6937.
Autoinstall for terminals and APPC connections

This validated MTS model is the only model passed to the autoinstall control program.

c. In the absence of an MTS model name, DFHZATA browses the autoinstall terminal model (AITM) table using the AIQ subroutine interface of the AITM manager. These models must have been installed, with appropriate TYPETERM definitions, either at system initialization or by a CEDA INSTALL command.

Compare the BIND image contained in each model with the BIND image passed in the CINIT RU, and build a list of suitable models to be passed to the autoinstall control program.

For autoinstall of an LU to be successful, the following must match:
- CINIT BIND image, taken from the VTAM LOGMODE entry specified for the LU in the VTAMLST
- Autoinstall terminal model BIND image, built according to the specifications in the TYPETERM and TERMINAL definitions.

(Both versions of the BIND image should accurately define the characteristics of the device.) If the model BIND matches the CINIT BIND, the model is added to the list of candidate entries.

If the list is empty (no matching models are found), the request is rejected and message DFHZC6987 is written to the CADL log.

d. On completion of the model search, if any, DFHZATA links to the autoinstall control program (the CICS-supplied default is DFHZATDX).

e. Issue DFHZCP_INSTALL to create the TCTTE. DFHZATA uses information from the model selected by the exit program and the associated TYPETERM entry to build the TCTTE.

f. If the install was successful, commit the TCTTE and queue it for LOGON processing. The new TCTTE is queued for OPNDST processing, then later the “good morning” message is written.

g. Free the AWE.

Autoinstall of APPC device logon flow

This section describes the flow of control for an APPC parallel session device (or single session via a BIND) that is to be logged on by autoinstall.

1. When an APPC device attempts to logon, VTAM drives the logon exit DFHZLGX if a CINIT is received, or the SCIP exit DFHZBLX if a BIND is received.

Note that DFHZBLX is a new VTAM exit module that is called by DFHZSCX if an LU62 BIND has been received.

In the following circumstances, an APPC LU is a candidate for autoinstall.
- If the connection is not already defined to CICS.
- If the connection is not already installed.
- If the autoinstall user program (specified by the AIEXIT system initialization parameter) exists and caters for functions 2-4 as well as functions 0-1.
- If the VTAM ACB is open.
- If it is an APPC parallel session connection.
- If it is an APPC single session connection with an incoming BIND (as opposed to CINIT - which uses terminal autoinstall).
- If ISC=YES is specified in the SIT.
Autoinstall for terminals and APPC connections

- If the maximum number of concurrent logon requests (specified by AIQMAX) has not been exceeded.
- If the customer has installed the correct ‘template’ connection that is to be ‘cloned’ (or copied) to create the new connection.

DFHZLGX or DFHZBLX searches for the connection in the terminal control table (TCT) by comparing the NETNAME passed by VTAM with the NETNAME found in the NIB descriptor for each installed session.

If a match is found and AUTOINSTALL is enabled (TCTVADEN is set), CICS verifies that the terminal is eligible for autoinstall. Processing then consists of:
- Building an autoinstall work element (AWE) by issuing an MVS GETMAIN for subpool 1.
- Copying the CINIT RU (DFHZLGX) or BIND (DFHZBLX) into the AWE.
- Adding the AWE to the end of the AWE chain, which is chained from the TCT prefix.

If a match is found showing that this connection already exists then the logon proceeds as for a defined connection.

2. Later, the AWE is actioned by DFHZACT attaching transaction CATA. For every AWE on the AWE chain, the DFHZATA autoinstall program is dispatched, passing to DFHZATA the AWE’s address.

3. The DFHZATA program:
   a. Validates the BIND image passed in the AWE. If the image is not valid, issue message DFHZC6901.
   b. Calls DFHZGAI Function(CREATE_CLONE_BPS) to create a Builder Parameter Set from which to create the new connection (‘clone’). This is done by calling the customer supplied autoinstall user exit program (which can be based on DFHZATDY) in which the customer chooses which ‘template’ connection the new connection should be copied from.
      If at any point DFHZGAI finds a problem it issues message DFHZC6920 or DFHZC6921 or DFHZC6922 with an exception trace entry which will explain the reason for failure.
   c. Issue DFHZCP function(INSTALL) to create the CONNECTION, MODEGROUP and SESSIONs, based on the attributes of the template connection.
   d. For parallel sessions with an incoming BIND, chose the SNASVCMG secondary session and call DFHZGAI (SET_TCTTE_FOR_OPNDST). This mimics code in DFHZBLX to check the session against the incoming BIND.
      If at any point DFHZGAI finds a problem it issues message DFHZC6923 with an exception trace entry which explains the reason for failure.
   e. For parallel session with an incoming CINIT, chose the SNASVCMG primary session.
   f. If the install was successful, commit the CONNECTION and queue it for logon processing. The new CONNECTION is queued for OPNDST processing.
   g. Free the AWE.

Autoinstall of an APPC Generic Resource connection

If this system is registered as a generic resource and a bind is received from another generic resource then VTAM exit DFHZBLX will initiate an autoinstall if there is no generic or member name connection available for use.
Autoinstall for terminals and APPC connections

An AWE is created with extra parameters such as the generic resource name and member name of the partner and possibly a suggested template.

Autoinstall then continues as for normal APPC and the extra parameters are reflected into the TCSE and TCTTE via the BPS.

Autoinstall of consoles install flow

1. The modify command comes into DFHZCNA via a CIB (Command Input Buffer) from MVS when a user types a console command for CICS.
2. DFHZCNA scans the Console Control Elements for a matching console name. If no CCE is found and autoinstall for consoles is enabled then an Autoinstall Work Element is created and added to the AWE queue.
3. DFHZACT scans the AWE queue and attaches the CATA transaction.
4. The CATA transaction calls DFHZATA which sees the AWE is for a console (sometimes called a Console Work Element) and calls DFHZATA2.
5. DFHZATA2 does the following:
   a. Finds the console models (AICONS is supplied in group DFHTERMC).
   b. If SIT AICONS(YES) is specified the models are passed to the autoinstall URM which returns the termid. The default AI URM returns the last 4-characters of the consolename.
   c. If SIT AICONS(AUTO) is specified DFHZGBM is called to get a name in the console bitmap in the form `AAA. The AI URM is not called.
   d. Calls DFHZCP FUNCTION(INSTALL).
   e. Issues EXEC CICS SYNCPOINT.
   f. Signs on if using preset security of USERID=*EVERY |*FIRST specified in the AI model TYPETERM.
   g. Gets a TIOA to hold the data specified in the command, e.g. if /f jobname,CEMT I TE was typed at the console then CEMT I TE is put into the TIOA.
   h. Call DFHZATT to attach the transaction specified in the MODIFY command (e.g. CEMT).

Sign-on to consoles flow

If a CIB is received with the same console name but with a different USERID then the autoinstall program DFHZATA2 is called to sign off the original USERID and sign on to the new USERID as follows:

1. DFHZCNA receives the modify and
   a. Finds the CCE
   b. Finds that the USERID is different and is already signed on
   c. Creates an AWE for signoff/on
   d. Chains the AWE for DFHZACT.
2. DFHZACT attaches CATA
3. CATA calls DFHZATA which calls DFHZATA2 for signoff/on
4. DFHZATA2 issues preset security sign off for the original USERID followed by sign on for the new USERID
5. DFHZATA2 then gets a TIOA for the modify command data and calls DFHZATT to attach the transaction as for normal autoinstall for consoles.
Disconnection flow for terminals (LU-initiated)

This section describes the flow of control when a request is made to disconnect an autoinstalled terminal (for example, by entering a CESF LOGOFF command), ultimately causing an EXEC CICS ISSUE LOGOFF command to be issued.

1. First the following functions are performed:
   - Set on the CLSDST flag in the TCTTE.
   - Put the TCTTE on the activate chain for DFHZACT to dispatch.

2. Control is then passed to the Close destination program, DFHZCLS, which performs the following functions:
   - Set on the SHUTDOWN_IN_PROGRESS flag in the TCTTE.
   - Set on the REQUEST_SHUTDOWN flag in the TCTTE.

3. The Send asynchronous commands program, DFHZDSA, is then called to send a VTAM SHUTD command to the LU (autoinstalled terminal) to be disconnected. The DFHZDSA program removes the TCTTE from the activate chain, pending completion of the SHUTD command.

4. When the VTAM SHUTD command has completed, VTAM calls the asynchronous send exit, DFHZSAX, which performs the following functions:
   - Set off the REQUEST_SHUTDOWN flag in the TCTTE.
   - Set on the SHUTDOWN_SEND flag in the TCTTE.
   - Put the TCTTE back on the activate chain for DFHZACT to dispatch.

5. VTAM then drives the asynchronous receive exit, DFHZASX, with the SHUTC (“shutdown complete”) command sent by the LU to be disconnected. DFHZASX performs the following functions:
   - Ensures that the NODE_QUIESCED_BY_CICS, SHUTDOWN_IN_PROGRESS, and CLSDST flags are still on.
   - Puts the TCTTE back on the activate chain for DFHZACT to dispatch.

6. Control is then passed to the Close_Destination program, DFHZCLS. The DFHZCLS program performs the following functions:
   - Set on the PENDING_DELETE flag in the TCTTE to prevent VTAM exits scheduling requests for the device.
   - Issue UNBIND (CLSDST POST=RESP) for the device.

7. The Close destination exit, DFHZCLX, is driven. If the CLSDST request is successful (that is, there is a positive response from UNBIND), the following functions are performed:
   - Set on the SESSION_CLOSED flag in the TCTTE.
   - Flag the TCTTE for deletion.
   - Enqueue the TCTTE to DFHZNAC.

8. Control is passed to the DFHZZNAC program, which performs the following functions:
   - Set on the DELETE_REQUIRED flag in the TCTTE.
   - Put the TCTTE on the activate chain for DFHZACT to dispatch.
   - Issue message DFHZC3462 (session terminated).

9. On the delete request, the DFHZZNCA copybook of DFHZNAC checks the value of the system initialization parameter AILDELAY.
   - If AILDELAY is zero, the TCTTE is queued via DFHZACT with the address of the TCTTE as input. Its function is to perform cleanup operations, the principal operation being to ask DFHZCQ to delete the TCTTE.
Autoinstall for terminals and APPC connections

- If AILDELAY is not zero, DFHZZNCA initiates CATD using the delay specified and passes the address of the TCTTE.

Up to three attempts are made to delete the TCTTE. This is because the reason for the failure may be the existence of a transient condition, such as the TCTTE being on the DFHZZNAC queue to output a message to CSMT. If the initial delete attempt fails, it is attempted again after one second; if this fails, another attempt is made after a further 5 seconds. If the third attempt fails, it is assumed that the failure is a hard failure, which will not disappear until the device is reconnected; in this case, message DFHZC6943 is issued, a syncpoint is taken, and the TCTTE delete status is reset to make the TCTTE reusable.

If the deletion is successful, the delete is committed, the autoinstall control program is invoked to permit any specific cleanup operations to take place, and message DFHZC6966 is issued.

If a PWE exists for this TCTTE, the PWE is requeued onto the AWE chain.

Disconnection of an autoinstalled terminal can also be requested by CICS shutdown, terminal time-out, and terminal errors. In these cases the flow is slightly different.

Deletion of autoinstalled APPC devices.

This section describes the flow of control when an APPC sync level 1 device has its last session released. This can occur as a result of unbind flows from the partner or a RELEASE command being issued against the connection in this system.

Only synclevel 1 autoinstalled connections are deleted in this way. They will have had TCSE_IMPLICIT_DELETE set by the builders from zx_delete_x in the BPS (set by DFHZGAI).

TCSE_CATLG_NO indicates that the connection is not to be written to the catalogue (SIT Parameter AIRDELAY=0).

1. After DFHZCLS, the CLSDST program, issues DFHTCPLR TIDYUP TCSEDDP and TCSE_DELETE_SCHEDULE are set and CATD is initiated with a delay of AILDELAY.

2. CATD runs DFHZATD which sets TCSE_DELETE_STARTED and calls DFHZCP FUNCTION=DELETE to delete the sessions, modegroup and connection.

If a SIMLOGON or BIND occur before the delete actually starts (TCSE_DELETE_SCHEDULED) then the connection delete is aborted and the connection reused.

If a SIMLOGON occurs during the actual delete (TCSE_DELETE_STARTED) then the delete is vetoed and the connection is reacquired.

If a BIND occurs during the actual delete (TCSE_DELETE_STARTED) then the delete goes ahead and the PWE that was created is turned into an AWE and the logon will create a new connection.

If TCSE_DELETE_AT_RESTART is set then DFHZATR will delete the connection if it has not been used after restart with a delay specified in the SIT AIRDELAY parameter.
**Autoinstall for terminals and APPC connections**

**Disconnection flow (APPC devices)**
These connections are not deleted at LOGOFF time, so the disconnection flow is the same as for a defined connection.

**Deletion of autoinstalled consoles**
Consoles are deleted after a certain period of inactivity. The default is 60 minutes but this can be overridden in the autoinstall URM.

1. The delete time is saved in the CCE during install in TCTCE_TIMEOUT_TIME.
2. DFHCESC runs at certain intervals
3. DFHCESC checks the CCEs for any console whose delete time has expired
4. For each expired CCE DFHCESC does the following
   - Attaches CATD to do the delete
   - CATD calls DFHZATD as for a terminal

**Shipping a TCTTE for transaction routing**
For transaction routing, a terminal can be defined by an entry in the terminal-owning region (TOR) with the SHIPPABLE=YES attribute. In this case, the terminal definition is shipped to any application-owning region (AOR) when the terminal user invokes a transaction owned by (and defined to) that region. Definitions for advanced program-to-program communication (APPC) devices always have the SHIPPABLE=YES attribute set.

(The entry in the TOR could have been installed using CEDA INSTALL, the GRPLIST at system initialization, or autoinstall.)

**The first time a transaction is invoked**
For non-APPC devices (see Figure 6 on page 121), the following processing is performed:

- In the AOR, look for an existing skeleton TCTTE (TCTSK) whose REMOTENAME is the same as the local name in the TOR. If found, skip the following steps; otherwise:
- Issue ZC_INQUIRE to the TOR.
- In the TOR:
  - Send a builder parameter set (BPS) representing the TCTTE to the AOR.
  - Set on the SHIPPED flag (TCTEMROP) in the TCTTE.
  - Set on the SHIPPED flag (TCSEMROP) in the TCTSE for the AOR system.
  - Rewrite each entry to the catalog.
- In the AOR:
  - Use the existing name from the TOR.
  - INSTALL the terminal (DFHZATS does the remote install).
  - Set on the SHIPPED flag (TCTSKSHI) in the TCTSK.
  - Set on the SHIPPED flag (TCSEMROG) in the TCTSE for the TOR system.
  - Rewrite each entry to the catalog.
For APPC devices:

- In the AOR, look for an existing skeleton TCTTE (TCTSK) whose REMOTENAME is the same as the local name in the TOR. If found, skip the following steps; otherwise:
  - INSTALL the terminal (DFHZATS does the remote install).
  - Set on the SHIPPED flag (TCTSKSHI) in the TCTSK.
  - Set on the SHIPPED flag (TCSEMROG) in the TCTSE for the TOR system.
  - Rewrite each entry to the catalog.

When an autoinstalled TCTTE in a TOR is deleted

If this CICS is linked to a Pre CICS/ESA 4.1 system then the following occurs.

- If the deleted entry is flagged (TCTEMROP or TCSEERDLR for APPC devices) as having been shipped, notify all remote systems that have received shipped definitions (TCSEMROP) that this terminal is being deleted.
- Determine from the TCTSK in the AOR whether a definition for this terminal has been shipped (TCTSKSHI). If so, call ZC_DELETE in the AOR.

If this CICS is linked to CICS/ESA 4.1 or above then relevant shipped terminals are deleted using a separate timing mechanism.

**Modules**

ZC (terminal control) together with the following:

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHZATA</td>
<td>Autoinstall program</td>
</tr>
<tr>
<td>DFHZATA2</td>
<td>Console autoinstall program linkedits with DFHZATA</td>
</tr>
<tr>
<td>DFHZATD</td>
<td>Autoinstall delete program</td>
</tr>
<tr>
<td>DFHZATDX</td>
<td>Autoinstall control program</td>
</tr>
</tbody>
</table>
Autoinstall for terminals and APPC connections

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHZATDY</td>
<td>Sample autoinstall user exit</td>
</tr>
<tr>
<td>DFHZATR</td>
<td>Autoinstall restart program</td>
</tr>
<tr>
<td>DFHZATS</td>
<td>Remote autoinstall/delete program</td>
</tr>
<tr>
<td>DFHZCTRI</td>
<td>Trace interpretation for DFHZGAI</td>
</tr>
<tr>
<td>DFHZGAI</td>
<td>APPC-specific autoinstall functions</td>
</tr>
</tbody>
</table>

DFHZATDX

The DFHZATDX module provides user input to autoinstall processing. This module is a component of ZCP, and is the default autoinstall user program (that is, it is used if you choose not to provide your own). For further information about the DFHZATDX sample program, see the [CICS Customization Guide](#).

DFHZATDX is also called when creating and deleting shipped terminals (skeletons).

DFHZATDY

DFHZATDY is a sample autoinstall user-replaceable module, which you must modify before you can use it. Its main function is to choose a template connection which is to be used in creating the new autoinstall connection clone. It also has to chose a name for the new connection. For further information about the DFHZATDY sample program see the [CICS Customization Guide](#).

DFHZATDY is also called when creating and deleting shipped terminals (skeletons).

Diagnosing autoinstall problems

When diagnosing problems with autoinstall, consult the following list. If you have a problem with autoinstall of APPC devices, and the following list does not resolve the problem, see "Diagnosing APPC autoinstall problems" on page 123.

- The autoinstall model table (AMT) in an SDUMP
- CEMT INQUIRE AUTINSTMODEL—showing which models are installed
- TC level-1 trace, point ID AP FC8A—showing the CINIT RU contained in the AWE on entry to DFHZATA
- CADL, CSMT, and CSNE logs:
  - Autoinstall messages (DFHZC69xx)
  - Builder messages (DFHZC59xx, DFHZC62xx, and DFHZC63xx)
  - Terminal error messages
  - Information produced by DFHZNAC
- Dump taken in the user install program (the CICS-supplied default is DFHZATDX).

Most autoinstall problems can be grouped into three categories:
1. CICS rejects the LOGON request (message DFHZC2411 on the CSNE log).
2. The device rejects the actual BIND parameters (message DFHZC2403 on the CSNE log).
3. DFHZATA diagnoses a problem (message DFHZC69xx on the CADL log).
The first category of problem is caused by CICS being in the wrong state to accept an autoinstall, for example, CICS is shutting down or AUTOINSTALL is disabled (message DFHZC2433).

The second category of problem arises when the two BIND images match, but the BIND is rejected by the actual device (message DFHZC2403). For information about valid BIND parameters, consult the 3274 Control Unit Description and Programmer's Guide, GA23-0061.

The BIND image is contained in the CINIT RU passed to the LOGON exit. This is shown in trace point ID AP FC8A.

The reason for the third category of problem should be shown in the contents of the associated DFHZC69xx message on the CADL log. For example, message DFHZC6987 shows a BIND image mismatch between the incoming CINIT and the best available model (unlikely).

The length of each BIND image is found in the halfword preceding the image. A comparison is made for the smaller of the two length values, but not exceeding X'19' (decimal 25) bytes. The comparison is accomplished by an XC (exclusive OR) of the two BIND images into a work area. The result is ANDeD with a mask that defines the required settings.

Additional bits are reset if the LU type, found in byte 14 of the BIND image, is 1, 2, 3, or 4. The final result in the work area must be 256 bytes of X'00'; any other value causes DFHZATA to reject the LOGON and write message DFHZC6987 to the CADL log.

For autoinstall to function correctly, three items must match:
1. The CINIT BIND image taken from the LOGMODE entry specified for the LU in the VTAMLST
2. The CICS MODEL BIND image built according to the specifications in the TYPETERM and TERMINAL entries
3. Device characteristics.

Diagnosing APPC autoinstall problems
When diagnosing APPC autoinstall problems, first refer to "Diagnosing autoinstall problems" on page 122. Most of points in that section apply to APPC autoinstall problems except for points that refer to autoinstall models.

Any APPC autoinstall problem should be accompanied by message DFHZC6920 to 23. These messages each have exception trace entries which should trace enough information to allow you to diagnose the problem.

There are three autoinstall instances of DFHZC2411:
• 4 System termination - CSASTIM tested.
• 5 VTAM termination - TCTVVTQS tested.
• 6 ISC=NO specified in the SIT.

There are two additional instances of DFHZC2433:
• 3 Autoinstall disabled - TCTVADEN tested in DFHZBLX.
• 4 Autoinstall temporarily disabled - TCTVADIN tested in DFHZBLX.
Autoinstall for terminals and APPC connections

There are two additional instances of DFHZC3482:

- 3 No MVS storage for DFHZBLX to obtain MVS AWE storage.
- 4 No MVS storage for reporting a failure in a dummy work element.

Diagnosing console autoinstall problems

Much of the autoinstall for terminal advice is relevant. However, the following points should also be helpful.

1. Information about autoinstalled consoles is contained in:
   - The AWE (CWE)
   - The TCT prefix in the console BITMAP
   - The CCE
   - The SNEX
   - The AI URM interface
2. When DFHZCNA is called with a modify command trace point AP FCF0 is issued and traces the CIB and CIB extension.
3. Trace point AP FCA7 shows the AWE/CWE created by DFHZCNA and passed to DFHZATA2.
4. DISCARD (used via CEMT or EXEC CICS) is useful whilst testing autoinstall for consoles.
5. CEMT INQUIRE TERMINAL is useful for seeing what consoles are installed and what their console names are.
6. The console names can vary depending on how the modify command was issued:
   - /f jobname,CEMT I TE from a TSO SDSF panel gives a console name of the USERID or the console name if changed using option 8 of SDSF.
   - f jobname,CEMT I TE from a TSO console gives a console name of the TSO USERID.
   - M/F jobname, CEMT I TE from the TSO SDSF panel gives a console name of MASTnn where nn is the names of the system. If SEC=YES is specified in the SIT then the user must first sign on with m/f jobname,CESN.
   - // MODIFY jobname,CEMT I TE from a job stream gives a console names of INTERNAL. If SEC=YES is specified in the SIT then the user must first sign on with m/f jobname,CESN.
7. The console name BITMAP is dumped in the TCP section of system dumps.
8. The extended control blocks are dumped if present when a system dump is taken.

VTAM exits

A VTAM exit is a special-purpose user-written routine that is scheduled by VTAM when the requested operation is complete. VTAM creates a trace record when the exit is given control.

RE entries represent RPL exits except SEND, RECEIVE, OPNDST, and CLSDST. UE entries represent non-RPL and asynchronous exits SCIP, LOGON, and LOSTERM.

See the [OS/390 eNetwork Communications Server: SNA Programming manual, SC31-8573](http://example.com) for general VTAM exit information.
The following point IDs are provided for the autoinstall programs (DFHZATA, DFHZATD, DFHZATR, and DFHZATS), as part of terminal control:
• AP FC80 through AP FC8C, for which the trace levels are TC 1 and TC 2.

The following point IDs are provided for APPC autoinstall:
• AP FA00 to FA21, for which the trace levels are TC1 and TC2.

The following point IDs are provided for console autoinstall:
• AP FCF0
• AP FCA3 to FCA7

RE and UE trace points are recorded when the VTAM trace API option is requested by:
F NET,TRACE(TYPE=VTAM,OPTION=API,MODE=EXT

GTF must have been started with the USR option.

Each VTAM exit routine in CICS sets an ID byte in the TCTTE exit trace field (TCTEEIDA).

For more information about the trace points, see the CICS Trace Entries. For more information about using traces in problem determination, see the CICS Problem Determination Guide.
Chapter 9. Autoinstall terminal model manager

The autoinstall terminal model manager (an OCO component of the AP domain) is responsible for managing all operations involving the autoinstall terminal model table. Autoinstall terminal models are used during the autoinstall logon process (see step 3 on page 114). They are installed either at system initialization or using CEDA INSTALL (see "Chapter 72. Resource definition online (RDO)" on page 997), and can be discarded using either the CEMT transaction or EXEC CICS commands.

The acronym AITM is often used for “autoinstall terminal model” in the contexts of both the manager and the associated table; it is also the name of one of the subroutine call formats.

The AITM manager is implemented as a set of subroutine interfaces.

Functions provided by the autoinstall terminal model manager

Table 6 summarizes the external subroutine interfaces provided by the autoinstall terminal model manager. It shows the subroutine call formats, the level-1 trace point IDs of the modules providing the functions for these formats, and the functions provided.

<table>
<thead>
<tr>
<th>Format</th>
<th>Trace</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIIN</td>
<td>AP 0F10</td>
<td>START_INIT</td>
</tr>
<tr>
<td></td>
<td>AP 0F11</td>
<td>COMPLETE_INIT</td>
</tr>
<tr>
<td>AIIQ</td>
<td>AP 0F18</td>
<td>LOCATE_TERM_MODEL</td>
</tr>
<tr>
<td></td>
<td>AP 0F19</td>
<td>UNLOCK_TERM_MODEL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INQUIRE_TERM_MODEL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>START_BROWSE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GET_NEXT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>END_BROWSE</td>
</tr>
<tr>
<td>AITM</td>
<td>AP 0F08</td>
<td>ADD_REPL_TERM_MODEL</td>
</tr>
<tr>
<td></td>
<td>AP 0F09</td>
<td>DELETE_TERM_MODEL</td>
</tr>
</tbody>
</table>

AIIN format, START_INIT function

The START_INIT function of the AIIN format is used to attach a CICS task to perform initialization of the AITM manager.

Input parameters
None.

Output parameters
RESPONSE
is the subroutine’s response to the call. It can have any of these values:
OK|DISASTER|KERNERROR

AIIN format, COMPLETE_INIT function

The COMPLETE_INIT function of the AIIN format is used to wait for the initialization task attached by the START_INIT function to complete processing.
Autoinstall terminal model manager

**Input parameters**
None.

**Output parameters**

- **RESPONSE**
  is the subroutine’s response to the call. It can have any of these values:
  - OK
  - DISASTER
  - KERNERROR

**AIIQ format, LOCATE_TERM_MODEL function**

The LOCATE_TERM_MODEL function of the AIIQ format is used to obtain the attributes of a named autoinstall terminal model, and obtain a read lock on that entry in the AITM table in virtual storage.

**Input parameters**

- **TERM_MODEL_NAME**
  specifies the name of the autoinstall terminal model to be located.
- **BPS**
  identifies a buffer into which the attributes of the autoinstall terminal model are to be placed.

**Output parameters**

- **RESPONSE**
  is the subroutine’s response to the call. It can have any of these values:
  - OK
  - EXCEPTION
  - DISASTER
  - KERNERROR

**[REASON]**

  is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>TM_LOCATE_FAILED</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>TERM_MODEL_NOT_FOUND</td>
</tr>
</tbody>
</table>

**AIIQ format, UNLOCK_TERM_MODEL function**

The UNLOCK_TERM_MODEL function of the AIIQ format is used to release a read lock on a previously located entry from the AITM table in virtual storage.

**Input parameters**

- **TERM_MODEL_NAME**
  specifies the name of the autoinstall terminal model to be unlocked.

**Output parameters**

- **RESPONSE**
  is the subroutine’s response to the call. It can have any of these values:
  - OK
  - EXCEPTION
  - DISASTER
  - KERNERROR

**[REASON]**

  is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>TM_UNLOCK_FAILED</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>TERM_MODEL_NOT_FOUND</td>
</tr>
</tbody>
</table>
**AIIQ format, INQUIRE_TERM_MODEL function**

The INQUIRE_TERM_MODEL function of the AIIQ format is used to obtain the attributes of a named autoinstall terminal model. (No read lock is retained.)

**Input parameters**

- **TERM_MODEL_NAME**
  - specifies the name of the autoinstall terminal model to be located.

- **BPS**
  - identifies a buffer into which the attributes of the autoinstall terminal model are to be placed.

**Output parameters**

- **RESPONSE**
  - is the subroutine’s response to the call. It can have any of these values: OK|EXCEPTION|DISASTER|KERNERROR
  - [REASON] is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>TM_LOCATE_FAILED</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>TERM_MODEL_NOT_FOUND</td>
</tr>
</tbody>
</table>

**AIIQ format, START_BROWSE function**

The START_BROWSE function of the AIIQ format is used to initiate a browse of the AITM table. The browse starts at the beginning of the table.

**Input parameters**

None.

**Output parameters**

- **BROWSE_TOKEN**
  - is a token used to refer to this browse session on subsequent browse requests.

- **RESPONSE**
  - is the subroutine’s response to the call. It can have any of these values: OK|DISASTER|KERNERROR|PURGED
  - [REASON] is returned when RESPONSE is DISASTER. It has this value: START_BROWSE_FAILED

**AIIQ format, GET_NEXT function**

The GET_NEXT function of the AIIQ format is used to obtain the name and attributes of the next autoinstall terminal model in the AITM table for the specified browse session.

**Input parameters**

- **BROWSE_TOKEN**
  - is the token identifying this browse session.
Autoinstall terminal model manager

**BPS** identifies a buffer to receive the attributes of the next entry in the AITM table.

**Output parameters**

**TERM_MODEL_NAME**

is the name of the next entry in the AITM table.

**RESPONSE**

is the subroutine’s response to the call. It can have any of these values:

- OK
- EXCEPTION
- DISASTER
- KERNERROR

**[REASON]**

is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>TM_GET_NEXT_FAILED, TM_UNLOCK_FAILED</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>END_OF_MODELS</td>
</tr>
</tbody>
</table>

**AIIQ format, END_BROWSE function**

The END_BROWSE function of the AIIQ format is used to terminate a browse of the AITM table.

**Input parameters**

**BROWSE_TOKEN**

is the token identifying this browse session.

**Output parameters**

**RESPONSE**

is the subroutine’s response to the call. It can have either of these values:

- OK
- KERNERROR

**AITM format, ADD_REPL_TERM_MODEL function**

The ADD_REPL_TERM_MODEL function of the AITM format is used to add or update an entry in the AITM table in virtual storage, and record the entry on the CICS catalog.

**Input parameters**

**TERM_MODEL_NAME**

specifies the name of the autoinstall terminal model to be added or updated.

**BPS**

specifies the attributes of the named autoinstall terminal model.

**SYSTEM_STATUS**

specifies the status of the CICS system at the time of the call. It can have any one of these values:

- COLD_START
- WARM_START
- ONLINE

where ONLINE means during execution.

**Output parameters**

**RESPONSE**

is the subroutine’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|KERNERROR

[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>NOT_INITIALISED, ADD_REPL_FAILED</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>TERM_MODEL_IN_USE</td>
</tr>
</tbody>
</table>

AITM format, DELETE_TERM_MODEL function

The DELETE_TERM_MODEL function of the AITM format is used to remove an entry from the AITM table in virtual storage and the CICS catalog.

Input parameters

TERM_MODEL_NAME
specifies the name of the autoinstall terminal model to be added or updated.

SYSTEM_STATUS
specifies the status of the CICS system at the time of the call. It can have any one of these values:
COLD_START|WARM_START|ONLINE

where ONLINE means during execution.

Output parameters

RESPONSE
is the subroutine’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|KERNERROR

[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>NOT_INITIALISED, DELETE_FAILED</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>TERM_MODEL_IN_USE, TERM_MODEL_NOT_FOUND</td>
</tr>
</tbody>
</table>

Modules

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHAIDUF</td>
<td>Formats the AITM manager control blocks in a CICS system dump</td>
</tr>
<tr>
<td>DFHAIIN1</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>START_INIT</td>
</tr>
<tr>
<td></td>
<td>COMPLETE_INIT</td>
</tr>
<tr>
<td>DFHAIIN2</td>
<td>Runs as a CICS task to perform initialization of the AITM manager</td>
</tr>
</tbody>
</table>
Autoinstall terminal model manager

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHAIQ</td>
<td>Handles the following requests: LOCATE_TERM_MODEL UNLOCK_TERM_MODEL INQUIRE_TERM_MODEL START_BROWSE GET_NEXT END_BROWSE</td>
</tr>
<tr>
<td>DFHAIRP</td>
<td>Initializes the AITM table at CICS startup</td>
</tr>
<tr>
<td>DFHAITM</td>
<td>Handles the following requests: ADD_REPL_TERM_MODEL DELETE_TERM_MODEL</td>
</tr>
<tr>
<td>DFHAPTRN</td>
<td>Interprets AITM manager trace entries</td>
</tr>
</tbody>
</table>

Exits

No global user exit points are provided for this component.

Trace

The following point IDs are provided for the AITM manager:
• AP 0F00 through AP 0F1F, for which the trace levels are AP 1 and Exc.

For more information about the trace points, see the CICS Trace Entries. For more information about using traces in problem determination, see the CICS Problem Determination Guide.
Chapter 10. Basic mapping support

Basic mapping support (BMS) allows the CICS application programmer to have access to input and output data streams without including device-dependent code in the CICS application program.

BMS provides the following services:

**Message routing**
This allows application programs to send output messages to one or more terminals not in direct control of the transaction.

**Terminal paging**
This allows the user to prepare a multipage output message without regard to the physical size of the output terminal; the output can then be retrieved by page number in any order.

**Device independence**
This allows the user to prepare output without regard to the control characters required for a terminal; CICS automatically inserts the control characters and eliminates trailing blanks from each line.

Most of the BMS programs are resident in the CICS nucleus.

Design overview

BMS is an interface between CICS and its application programs. BMS formats input and output display data in response to BMS commands in programs. To do this, it uses device information from CICS system tables, and formatting information from maps that you have prepared for the program.

BMS enables an application program to read in device-dependent data and convert it to a device-independent standard form, or to generate device-dependent output data from this device-independent standard form. In both cases, the structure of the device-independent standard form, and the layout of the data on the display terminal, are determined by a user-defined map. Related maps—for example, maps used in the same application program—are grouped together into a map set. See the CICS Application Programming Guide for further information about the definition and use of maps and map sets.

On some terminals (such as the IBM 8775 display terminal and the IBM 3290 information panel), the available display area may be divided into a set of related “logical” screens called partitions. The layout and properties of the set of partitions that can be simultaneously displayed on a terminal are defined by the BMS user in a partition set. See the CICS Application Programming Guide for further details about the definition and use of partition sets.

Maps, map sets, and partition sets are assembled offline using CICS macros. The user defines and names fields and groups of fields that can be written to and read from the devices supported by BMS. The assembled maps contain all the device-dependent control characters necessary for the proper manipulation of the data stream.
Basic mapping support

Associated with each map is a table of field names which is copied into each application program that uses the map. Data is passed to and from the application program under these field names. The application program is written to manipulate the data under the various field names so that alteration of a map format does not necessarily lead to changes in program logic. New fields can be added to a map format without making it necessary to reprogram existing applications.

Output data can be supplied from the application program by placing the data in the table under the appropriate field name. As an alternative, output maps can contain field default data that is sent when data is not supplied by an application program. This facility permits the specification of titles, headers, and so on, for output maps.

Optionally, the display of all the default data can be suppressed by the application program for any output map. Each time a map is used, the application program can temporarily modify the attributes of any named field in the output map. The extended attributes can also be modified if maps are defined with the DSATTS operand.

Output map fields with no field names can contain default data, but the application program cannot replace the default data or modify the attributes of unnamed fields.

For input, the user assembles a map defining the fields that can be written to and received from a particular device. Any data received for a particular field is moved across using the field name in the symbolic storage definition for the map. Light-pen-detectable fields defined in an input map are flagged as detected if present in an IBM 3270 Information Display System input stream. An input map for a particular case can specify a subset of the fields potentially receivable; any fields received and not represented in that map are discarded. This permits the number of fields from a map that can be typed or selected to be changed, without making it necessary to reprogram applications that currently receive data from the map.

Maps are stored in the CICS program load library. When a map is required by BMS, a copy is automatically retrieved by CICS from the program load library without application program action. Multiple users of a map contained in the program load library share a single copy in main storage.

BMS permits any valid combination of field attributes to be specified by the user when generating maps. Inclusion of BMS in CICS is a system generation option and does not prevent the application program from accessing a particular device in native mode (without using BMS). Intermixing BMS and native mode support for a terminal from the same application program may yield unpredictable results. When using mixed mode support, it is the user’s responsibility to ensure the correct construction and interpretation of native mode data streams.

BMS permits the application program to pass a native mode data stream that has already been read in, and (if, for a terminal of the IBM 3270 Information Display System, the screen has been formatted) to interpret this data stream according to a given input map. This facility allows data entered with the initial reading of a transaction to be successfully mapped using BMS.

BMS provides the following services:

- Message routing
Message routing

Message routing permits the application program to send an output message to one or more terminals not in direct control of the transaction. The message is automatically sent to a terminal if the terminal status allows reception of the message. If a terminal is not immediately eligible to receive the message, the message is preserved for that terminal until a change in terminal status allows it to be sent. The message routing function is used by the CICS message-switching transaction.

A BMS map that specifies extended attributes can be used for terminals that do not support extended attributes. When sending data to a variety of terminals, some of the terminals may support extended attributes and others may not. When a BMS ROUTE request is processed, BMS looks at the TCTTEs for all specified terminals and constructs a set of all the supported attributes.

A data stream is produced by BMS using this set of attributes, and the data stream and set of attributes for each page are written to a temporary-storage record. When the page is later read from temporary storage, the data stream for each terminal is modified, if necessary, to delete attributes not supported by that terminal.

Terminal paging

Terminal paging allows the user to prepare more output than can be conveniently or physically displayed at the receiving terminal. The output can then be retrieved by pages in any order; that is, in the order in which they were prepared or by skipping forward or backward in the output pages.

Terminal paging also provides the ability to combine several small areas into one area, which is then sent to the terminal. This enables the user to prepare output without regard for the record size imposed by the output terminal.

CICS provides the terminal operator with a generalized page retrieval facility that can be used to retrieve and dispose of pages.

Device independence

Device independence allows the user to prepare output without regard for the control characters required for message heading, line separation, and so on. Input to device independence consists of a data string with optional new-line characters. Device independence divides the data string into lines no longer than those defined for the particular terminal. If new-line characters appear occasionally in the data string to further define line lengths, they are not ignored. CICS inserts the appropriate leading characters, carriage returns, and idle characters, and eliminates trailing blanks from each line. If the device does not support extended attributes, the extended attributes are ignored.

CICS allows the user to set horizontal and vertical tabs on those devices that support the facility (for example, the IBM 3767 Communication Terminal, and the IBM 3770 Data Communication System). For such devices, CICS supports data compression inbound and data compression outbound, based on the tab characteristics in the data stream under the control of the appropriate maps.
BMS makes use of the following control blocks (see Figure 7 on page 138):

<table>
<thead>
<tr>
<th>DSECT</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHMAPDS</td>
<td>Defines a physical map. It contains overlays for map set data, map data, and field data. The physical map set is stored in the CICS program library and requires a resource definition when loaded into main storage by BMS.</td>
</tr>
<tr>
<td>DFHMCAD</td>
<td>Defines a mapping control area (MCA). MCAs are used in DFHM32 and DFHML1 to merge (both) and sort (DFHML1 only) fields in different maps in the chain of map copies. The MCA contains field position, flags, and pointers to map and application data structure relating to this field.</td>
</tr>
<tr>
<td>DFHMCBDS</td>
<td>Defines the message control block (MCB). MCBs are built and referenced by DFHTPR. There is one MCB per level of page chaining. The MCBs are chained together, with the head of the chain anchored off the TCTTE BMS extension. The MCB contains a copy of the MCR, with additional working data.</td>
</tr>
<tr>
<td>DFHMCRDS</td>
<td>Defines the message control record (MCR). MCRs are held in CICS temporary storage. There is one MCR per BMS message in temporary storage. The MCR contains data such as the number of pages in this message, the list of target terminals for this message, data on which pages are for which LDCs or partitions, and so on. The MCR is written to temporary storage by DFHMCP. It is read and purged by DFHTPR, DFHTPS, and DFHTPQ.</td>
</tr>
<tr>
<td>DFHOSPWA</td>
<td>Defines the output services processor work area (OSPWA). This is the main BMS control block. For standard and full-function BMS, there is an OSPWA that is chained off the TCA and is built by DFHMCP on the first BMS command in a transaction. It contains a copy of the BMS TCA request bytes, together with the BMS status and working area. DFHTPR has its own private OSPWA. This overlays the TWA for DFHTPR unless SEND PAGE RETAIN is used. If SEND PAGE RETAIN is used, DFHTPR obtains an additional OSPWA, and chains the base OSPWA off the new OSPWA. This avoids DFHTPR damaging the base OSPWA. The OSPWA is deleted during task termination. A shorter version of the OSPWA is used by DFHMCPE (part of both the minimum-function BMS mapping control program DFHMCPE$ and also the BMS fast-path module DFHMCX). It is built in DFHMCPE’s LIFO storage, and includes space for the request information from the TCA. The DFHMCPE OSPWA is defined within DFHMCPE.</td>
</tr>
<tr>
<td>DFHPGADS</td>
<td>Defines a page control area (PGA). DFHTPP builds a PGA at the end of the device data stream in the terminal input/output area (TIOA) (addressed as ADDR(TIOADBA) + TIOATDL) for the SET and PAGING disposition. The PGA contains the 3270 write control character (WCC), flags about the type of TC write required, and the extended features used in this page of data stream.</td>
</tr>
<tr>
<td>DFHPSDDS</td>
<td>Defines a physical partition set. The partition set is stored in the CICS program library and requires a resource definition when loaded into main storage by BMS.</td>
</tr>
</tbody>
</table>

Basic mapping support

Control blocks
<table>
<thead>
<tr>
<th>DSECT</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHTTPDS</td>
<td>Defines the terminal type parameter (TTP). This contains information for a terminal type. Note that BMS builds pages on a TTP basis. For standard and full-function BMS, DFHRLR builds TTPs as follows: 1. A “direct TTP” is built for the transaction terminal. If this supports partitions or LDCs, a further direct TTP is built for each referenced LDC or partition. This contains data for that LDC or partition. These direct TTPs are chained together, and the head of the chain is contained in the OSPWA. Direct TTPs are deleted by DFHMCP on a SEND PAGE, PURGE MESSAGE, or SEND PARTNSET command. 2. If routing is in effect, there is a chain of routed TTPs, with one TTP per terminal type in the route list. Routed TTPs are deleted by DFHMCP on a SEND PAGE or PURGE MESSAGE command. Most of BMS uses the TTP rather than the TCTTE to determine terminal-related information.</td>
</tr>
</tbody>
</table>

| TCTTETTE | The TCTTETTE DSECT in the DFHTCTZE macro defines the TCTTE BMS extension. It is chained off the TCTTE (TCTTETEA field). |

| DFHTPE  | Defines the BMS partition extension. This is chained off the TCTTE BMS extension if the terminal supports partitions. |

See the [CICS Data Areas](#) manual for a detailed description of these control blocks.
Basic mapping support

TCA
- X'08' TCAFCAAA
  Address of facility
- X'158' TCAOSPWA
  Address of BMS work area

TCTTE
- X'78' TCTTETEA
  Address of TCTTE extension

OSPWA
- X'AB' OSPCTTP
  Address of current TTP
- X'AC' OSPDTTP
  Address of direct TTP
- X'B0' OSPTTP
  Address of first routing TTP
- X'C0' OSPTIOA
  Address of original TIOA
- X'D0' OSPDWE
  Address of DWE

Direct TTP
- X'24' TTPPGBUF
  Address of page buffer
- X'2C' TPMLA
  Address of loaded map set
- X'30' TTPMAPA
  Address of map (within map set)
- X'34' TTPMMFCP
  Address of modified map

Map set
- MAP

Page buffer
- MAP (copy)

User TIOA
- BMSMDA
  Address of data (TIOA)

Notes:
1. The route list area (RLA) is not used in the direct TTP.
2. Each routing TTP has the

RLA extension
- X'08' TPRRLCHA
  Address of next RLA or zero

Notes:
1. The route list area (RLA) is not used in the direct TTP.
2. Each routing TTP has the
## Modules

BMS makes use of the following modules (see Figure 8 on page 141):

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHDSB</td>
<td>Addresses the page buffer, which was composed by the page and text build program (DFHPBP).</td>
</tr>
<tr>
<td>DFHEMS</td>
<td>The EXEC interface processor for BMS commands.</td>
</tr>
<tr>
<td>DFHIIP</td>
<td>Called in response to requests for BMS services involving terminals other than IBM 3270 Information Display Systems.</td>
</tr>
<tr>
<td>DFHMCP</td>
<td>The interface between application programs and the modules that perform mapping, message switching, page and text building, device-dependent output preparation, and message disposition to terminals, temporary-storage areas, or the application program.</td>
</tr>
</tbody>
</table>
| DFHM CX  | The BMS fast path module for standard and full-function BMS, and the program for minimum BMS support. It is called by DFHMCP if the request satisfies one of the following conditions:  
  - It is a non-cumulative direct terminal send map or receive map issued by a command-level program.  
  - It is for a 3270 display or an LU3 printer which does not support outboard formatting. If the terminal supports partitions, it is in the base state.  
  - The CSPQ transaction has been started.  
  - The message disposition has not changed. |
| DFHM32   | Called in response to requests for BMS services involving terminals of the 3270 Information Display System. |
| DFHPBP   | Processes all BMS output requests (SEND MAP, SEND PAGE, and SEND TEXT). It performs the following functions:  
  - Positions the data in the page, either by actually placing it in a buffer, or by copying it and adjusting the map for an IBM 3270 Information Display System (SEND MAP ACCUM)  
  - Places the data into the page buffer (SEND TEXT ACCUM)  
  - Inserts device-dependent control characters for other than 3270 Information Display System devices, removing extended attributes. |
| DFHPHP   | Processes terminal operations that involve partitions.                    |
| DFHRLR   | Builds terminal type parameters (TTPs), which are the main blocks for building and writing out data in BMS. |
| DFHTTPP  | Directs completed pages to a destination specified in the BMS output request: SEND TEXT sends to the originating terminal; SEND MAP PAGING or SEND TEXT PAGING directs to temporary storage; and SEND MAP SET or SEND TEXT SET directs to a list of completed pages that are returned to the application program. |
| DFHTTPQ  | Checks the chain of automatic initiate descriptors (AIDs) to detect and delete AIDs that have been on the chain for an interval exceeding the purge delay time interval specified by the PRGDLAY system initialization parameter, if this has a nonzero value. |
| DFHTPR   | Processes messages built by BMS and placed in temporary storage.          |
Basic mapping support

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHTPS</td>
<td>Invoked for each terminal type to which a BMS logical message built with SEND MAP PAGING or SEND TEXT PAGING is to be sent. For each terminal designated by the originating application program, DFHTPR is scheduled to display the first page of the logical message if the terminal is in paging status, or the complete message if it is in autopage status.</td>
</tr>
</tbody>
</table>

Basic mapping support (BMS) is provided by means of a number of modules, each of which interfaces with other BMS modules, CICS control components, and application programs. The maps that are handled by BMS may be new maps, created to utilize BMS mapping capabilities. The interrelationships of CICS programs requesting mapping services are summarized in Figure 8 on page 141. Further details for specific programs within BMS are given on pages that follow.

One of three versions (MINIMUM, STANDARD, or FULL) of basic mapping support can be selected by the system initialization parameter BMS (see the CICS System Definition Guide). Where the generated versions of a BMS module differ according to the level of function provided, a suffix identifies the version as follows:
- E$ for minimum function
- A$ for standard function
- 1$ for full function.

In the module lists that follow, an asterisk (*) after a module name shows that the module is suffixed in this way. Elsewhere in this book, however, the BMS modules are usually referenced by their unsuffixed names with no distinction made between the minimum, standard, and full-function versions.

The module used by all three versions of BMS (minimum, standard, and full-function) is:
- DFHMCP* (mapping control program).

Additional modules used by both standard and full-function versions of BMS are:
- DFHDSB* (data stream build)
- DFHIIIP* (non-3270 input mapping)
- DFHMCX (fast path module)
- DFHML1 (LU1 printer mapping)
- DFHM32* (3270 mapping)
- DFHPBP* (page build program)
- DFHPHP (partition handling program)
- DFHRLR* (route list resolution)
- DFHTPP* (terminal page processor).

Additional modules used only by full-function BMS are:
- DFHTPQ (terminal page cleanup)
- DFHTPR (terminal page retrieval)
- DFHTPS (terminal page scheduling).

A detailed description of each of these modules follows in alphabetic order of module name.
DFHDSB (data stream build)

The data stream build program addresses the page buffer, composed by the page and text build program (DFHPBP). The page buffer contains lines of output data that are to be written to a terminal other than an IBM 3270 Information Display.
Basic mapping support

System. The number of lines is contained in the TTPLINES field. The data stream build program performs the following functions on the data in the page buffer:

- Truncates trailing blanks within data lines
- Substitutes strings of physical device control characters for logical new-line characters that terminate each line of data
- Provides a format management header (FMH) for some VTAM-supported devices
- Allows horizontal and vertical tab processing.

Figure 9 shows the relationships between the components of data stream build.

Notes:

1. DFHDSB is entered from the page build program to process the page buffer.
2. For SEND TEXT commands with the NOEDIT option specified, page buffer compression is skipped and control returns to DFHPBP, which calls the terminal page processor (DFHTPP).
3. For SEND TEXT commands without the NOEDIT option, the appropriate device control characters for the target device are selected for substitution.
4. The page buffer containing the data to be compressed is located through the address stored at TTPPGBUF.
5. After compression of the page buffer data, control returns to DFHPBP, which calls DFHTPP to provide disposition of the page.
DFHIIP (non-3270 input mapping)

The non-3270 input mapping program (DFHIIP) is called in response to requests for BMS services involving terminals other than IBM 3270 Information Display Systems.

Figure 10 shows the relationships between the components of non-3270 input mapping.

Notes:
1. A RECEIVE MAP request by an application program, communicating with other than an IBM 3270 Information Display System, passes information through the TCA through the mapping control program (DFHMCP) to DFHIIP.
2. The map required for an operation is either passed by the application program or loaded by DFHMCP.
3. DFHIIP communicates with storage control to obtain and release buffers for mapping operations.

DFHMCP (mapping control program)

The mapping control program (DFHMCP) is the interface between application programs and the modules that perform mapping, message switching, page and
Basic mapping support

text building, device-dependent output preparation, and message disposition to terminals, temporary-storage areas, or the application program.

Figure 11 on page 145 shows the relationships between the components of mapping control.
Basic mapping support

Notes:
1. This program is entered when an application program issues a request for basic mapping support services.
2. It may also be called by task control to process a deferred work element (DWE) if an application program terminates and there are partial pages in storage, or the message control record (MCR) created during execution of the task has not been placed in temporary storage.
3. The following information is returned to the requester: error codes, page overflow information, and (for a SEND MAP SET or SEND TEXT SET command) a list of completed pages.
4. DFHMCP communicates with temporary storage control to put the MCR for routed or stored messages, if a ROUTE command, or SEND MAP PAGING or SEND TEXT PAGING command is issued. A DELETEQ TS command is issued to request that a message be purged from temporary storage if a PURGE MESSAGE command is issued.
5. DFHMCP communicates with storage control to:
   - Acquire and free storage in which the MCR is built (a SEND MAP command after a SEND MAP PAGING, SEND TEXT PAGING, or ROUTE command)
   - Acquire and free storage in which to copy the message title (a ROUTE command with the TITLE option specified)
   - Acquire storage to build automatic initiate descriptors (AIDs) for non-routed messages, or routed messages to be delivered immediately (a SEND PAGE command)
   - Acquire a BMS work area (OSPWA) at the time of the initial BMS request
   - Acquire and free an area used for user request data if a SEND PAGE command must be simulated before processing the user’s request
   - Free the returned page list (a DELETEQ TS command)
   - Free map copies if SEND PAGE command was issued and pages were being built in response to SEND PAGE commands
   - Free terminal type parameters (TTPs) (SEND PAGE command).
6. DFHMCP communicates with program manager to:
   - Load and delete map sets
   - Link to the terminal page retrieval program (DFHTPR) to process one or more pages of a message if a SEND PAGE command is issued with the RETAIN or RELEASE option specified
   - Abnormally terminate tasks that incur errors that cannot be corrected.
7. DFHMCP communicates with interval control to:
   - Initiate transaction CSPQ
   - Obtain the current time of day, which is then used to time stamp AIDs for routed messages
   - Initiate transaction CSPS for messages to be delivered later.
8. DFHMCP communicates with task control to schedule transaction CSPQ for every terminal that is to receive a routed message to be delivered immediately.
9. Transient data control is used to send error and information messages to the master terminal.
10. Route list resolution (DFHRLR) is used to collect terminals from a user-supplied route list or from the entire TCT by terminal type, and build a
terminal type parameter (TTP), which controls message building, for each
terminal type. It is also used to build a single-element TTP for the originating
terminal.

11. Non-3270 input mapping (DFHIIP) is used to process RECEIVE MAP requests
for a terminal other than an IBM 3270 Information Display System.

12. The mapping control program calls DFHMCX if the request is eligible for the
BMS fast-path module.

13. 3270 mapping (DFHM32) is used to process RECEIVE MAP requests for an
IBM 3270 Information Display System.

14. Page and text build (DFHPBP) processes the following output requests:

15. Page and text build program (DFHPBP) processes all BMS output requests
   SEND MAP
   SEND MAP PAGING
   SEND MAP SET
   SEND PAGE
   SEND TEXT
   SEND TEXT PAGING
   SEND TEXT SET.

   For 3270 output, DFHM32 is called; for other output, DFHML1 is called.

16. The partition handling program (DFHPHP) is called when the data is in an
    inbound structured field. DFHPHP extracts the partition ID, device AID, and
cursor address.

**DFHML1 (LU1 printer with extended attributes mapping)**

The LU1 printer with extended attributes mapping program, DFHML1, is called in
response to requests for BMS services involving terminals of the 3270 Information
Display System. Figure 12 on page 148 shows how the DFHML1 program responds
to these requests.
Basic mapping support

![Diagram of LU1 printer with extended attributes mapping program interfaces](image)

**Notes:**
1. The following types of requests, by application programs communicating with LU1 printer mapping, pass information through the mapping control program (DFHMCP), and the page and text build program (DFHPBP), to DFHML1:
Basic mapping support

SEND MAP ACCUM
SEND MAP SET
SEND TEXT
SEND TEXT ACCUM
SEND TEXT SET

For one page of output, DFHML1 acquires an area and formats it into a chain of control blocks known as map control areas (MCAs). Each MCA corresponds to one map on the page and contains information about chaining down the maps and processing the fields in each map. DFHML1 then builds the data stream directly from the maps and the TIOAs.

2. Maps are either passed by the application program or loaded by DFHMCP.
3. The address of a terminal input/output area (TIOA) is supplied by the application program for all requests.
4. DFHML1 communicates with storage control to obtain and release storage for MCAs and for the mapped data.
5. All requests (see note 1 on page 148) are sent to a designated destination by the terminal page processor (DFHTPP), after the return of control to DFHPBP.

DFHM32 (3270 mapping)

The 3270 mapping program (DFHM32) is called in response to requests for BMS services involving terminals of the 3270 Information Display System. Figure 13 on page 150 shows how the 3270 mapping program responds to these requests.
Basic mapping support

Figure 13. 3270 mapping program interfaces

Notes:

1. The following types of requests by an application program communicating with an IBM 3270 Information Display System passes information through the TCA by way of the mapping control program (DFHMCP) and the page and text build program (DFHPBP) to DFHM32:

   SEND MAP ACCUM
   SEND MAP PAGING
   SEND MAP SET
   SEND TEXT
   SEND TEXT ACCUM
   SEND TEXT PAGING
   SEND TEXT SET

For one page of output, DFHM32 acquires an area and formats it into a chain of control blocks known as map control areas (MCAs). Each MCA corresponds
Basic mapping support

to one map on the page and contains information for chaining down the maps and processing the fields in each map. DFHM32 then builds the data stream directly from the maps and the TIOAs.

2. A RECEIVE MAP or RECEIVE MAP FROM request by an application program communicating with an IBM 3270 Information Display System passes information through the TCA through the message control program (DFHMCP) to DFHM32.

3. Maps are either passed by the application program or loaded by DFHMCP.

4. DFHM32 communicates with storage control to obtain and release storage for MCAs and for the mapped data.

5. All output requests (see note [1 on page 150]) are sent to a designated destination by the terminal page processor (DFHTPP) after control is returned to DFHPBP.

DFHPBP (page and text build)
The page and text build program (DFHPBP) processes all BMS output requests
SEND MAP
SEND MAP PAGING
SEND MAP SET
SEND PAGE
SEND TEXT
SEND TEXT PAGING
SEND TEXT SET.

It performs the following functions:
• Positions the data in the page, either by actually placing it in a buffer, or by copying it and adjusting the map for an IBM 3270 Information Display System (SEND MAP ACCUM)
• Places the data into the page buffer (SEND TEXT ACCUM)
• Inserts device-dependent control characters for other than 3270 Information Display System devices, removing extended attributes.

Figure 14 on page 153 shows the relationships between the components of page and text build.
Basic mapping support
Notes:

1. DFHPBP is entered from the mapping control program, DFHMCP, to process all BMS output requests. It is called once for each terminal type parameter (TTP) on the TTP chain pointed to by OSPTTP. The current TTP in the chain is pointed to by OSPCTTP.

2. DFHPBP returns control to DFHMCP when request processing is complete, or when the page must be written out before a SEND MAP ACCUM request can be processed and an OFLOW=symbolic address operand was specified.

3. OSPTR2, OSPTR3, ..., OSPTR7 contain request data from the DFHBMS macro expansion. OSPRC1 and OSPRC3 contain return codes to be examined by DFHMCP.

4. For a SEND MAP ACCUM request for an IBM 3270 Information Display System, the map is copied and chained to the TTP. For a SEND TEXT ACCUM request for an IBM 3270 Information Display System, a dummy map is created and chained to the TTP. When a page is complete, control is given to 3270 mapping (DFHM32), which combines the map copies chained to the TTP and maps the data.

   For a SEND MAP ACCUM request for an LU1 printer with extended attributes, the map is copied and chained to the TTP. For a SEND TEXT ACCUM request, a dummy map is created and chained to the TTP. When a page is complete, control is given to the LU1 printer mapping program (DFHML1), which combines the map copies chained to the TTP and maps the data.

5. DFHPBP communicates with storage control to:
   - Acquire and free buffers in which pages are built
   - Acquire storage for copies of maps for SEND MAP ACCUM or SEND TEXT ACCUM when a page is to be processed
   - Acquire storage for a copy of the user’s data for SEND MAP ACCUM or SEND TEXT ACCUM when a page is to be processed

6. DFHPBP requests program manager to terminate a transaction abnormally (ABEND) if certain errors occur that cannot be corrected.

7. A SEND TEXT ACCUM request for an IBM 3270 Information Display System causes a map set consisting of one dummy map to be passed to 3270 mapping (DFHM32). The map has one field with attributes FREEKB and FRSET.

   SEND TEXT ACCUM requests for an LU1 printer cause a map set consisting of one dummy map to be passed to the LU1 printer mapping program (DFHML1). The map has one field with attributes FREEKB and FRSET.

8. If the page is being constructed for an IBM 3270 Information Display System, control is given to DFHM32 to map the data and then to DFHTPP to output the page.

   If the page is being constructed for an LU1 printer, control is given to DFHML1 to map the data, and then to DFHTPP to output the page. Otherwise, control is given to DFHDSB to add device dependencies to the page, and then to the terminal page processor (DFHTPP) to output the page.

DFHPHP (partition handling program)

The partition handling program (DFHPHP) processes terminal operations that involve partitions. DFHPHP has one entry point, and starts with a branch table that passes control to the required routine according to the request. It consists of routines that perform the following functions:

- PHPPSI tests whether there is a partition set in storage. If there is and it is not the required partition set, that partition set is deleted. When no partition set is in storage, an attempt is made to load the appropriate partition set.
Basic mapping support

- PHPPSC builds a data stream to destroy any partitions that may already be loaded on the terminal, creates the partition set designated by the application partition set, and sets the name of the partition set in the TCTTE to be the name of the application partition set.
- PHPPIN extracts the AID, cursor address, and partition ID. The AID and cursor address are put in the TCTTE, and the partition ID is converted to a partition name and returned to the caller. A check is made that the partition ID is a member of the application partition set.
- PHPPXE sends a data stream to a terminal to activate the appropriate partition and sends an error message to any error message partition if input arrived from an unexpected partition.

Figure 15 on page 155 shows the relationships between the components of partition handling.
Notes:
1. DFHPHP is called by the mapping control program (DFHMCP) and by the terminal output macro (DFHTOM).
2. PHPPSI refers to OSPWA to check whether a partition set is loaded.
3. PHPPSI communicates with program manager to load the partition set.
Basic mapping support

4. PHPSI puts the name of the partition set in TPE (terminal partition extension) as the application partition set.
5. PHPPSC calls storage control to acquire a TIOA in which to build and free the original TIOA.
6. PHPPSC sets a slot in the TCTTE to be the partition set data stream concatenated with the terminal partition set name if the terminal is not in the base state.
7. PHPPIN places the AID and the cursor address in the TCTTE.
8. PHPPXE calls storage control to get a TIOA, retrieves the error message text by calling the message domain, fills the TIOA with data, transmits the data, and frees the TIOA.
9. PHPPSC references the partition set object to build the partition creation data stream.

DFHRLR (route list resolution program)

The route list resolution program (DFHRLR) builds terminal type parameters (TTPs), which are the main blocks for building and writing out data in BMS.

Figure 16 on page 157 shows the route list resolution program interfaces.
Basic mapping support

Figure 16. Route list resolution program interfaces
Basic mapping support

Notes:

1. DFHRLR is called by the mapping control program (DFHMCP) to determine the grouping of terminal destinations.

2. If data is to be routed, DFHRLR groups the terminals in the user’s route list by terminal type and builds a routing TTP for each type. For each TTP, the supported attributes of the corresponding terminals are accumulated. The address of the first routing TTP in the chain of TTPs is placed in OSPTTP.

3. If data is not to be routed, a direct TTP is built for the originating terminal and its address is placed in OSPDTTP.

4. DFHRLR communicates with storage control to acquire storage for the TTP.

5. Program manager services are requested by means of an ABEND command if errors occur that cannot be corrected.

DFHTPP (terminal page processor)

The terminal page processor (DFHTPP) directs completed pages to a destination specified in the BMS output request:

- SEND MAP or SEND TEXT sends to the originating terminal
- SEND MAP PAGING or SEND TEXT PAGING directs to temporary storage
- SEND MAP SET or SEND TEXT SET directs to a list of completed pages that are returned to the application program.

Figure 17 on page 159 shows the relationships between the terminal page processor and other components in response to BMS output requests.
Notes:
1. DFHTPP is entered from DFHPBP after processing by 3270 mapping (DFHM32) for 3270s, by LU1 printer with extended attributes mapping (DFHML1) for those LU1 printers, and by data stream build (DFHDSB) for other devices.
Basic mapping support

2. DFHTPP communicates with storage control to obtain:
   - The return list (to store the address of completed pages to be returned to the program)
   - Deferred work elements (DWEs), which ensure that message control information is written to disk, even if the program neglects to issue a SEND PAGE request
   - Storage for a list that correlates pages on temporary storage with the logical device codes for which they are destined.

3. Temporary-storage control is used to store pages and the message control record (MCR) for messages stored on temporary storage.

4. The terminal type parameter (TTP) controls the formatting of a message for a particular terminal type (for example, an IBM 2741 Communication Terminal). TTPPGBUF contains the address of a completed page.

5. The terminal output macro (DFHTOM) is issued to provide an open subroutine assembled within DFHTPP that puts a completed page out to the terminal. If the data stream contains extended attributes, and the terminal does not support extended attributes, the extended attributes are deleted.

DFHTPQ (undelivered messages cleanup program)

The undelivered messages cleanup program (DFHTPQ) checks the chain of automatic initiate descriptors (AIDs) to detect and delete AIDs that have been on the chain for an interval exceeding the purge delay time interval specified by the PRGDLAY system initialization parameter, if this has a nonzero value.

Figure 18 on page 161 shows the undelivered messages cleanup program interfaces.
Notes:

1. DFHTPQ is initiated the first time by the mapping control program (DFHMCP), by interval control, or by the transaction CSPQ. Thereafter, it reinitiates itself (see note 5).

2. DFHTPQ communicates with the allocation program (DFHALP) to locate and unchain AIDs.

3. DFHTPQ communicates with storage control to free AIDs that have been purged and to acquire storage for notification messages.

4. Transient data control is used to send notification messages.

5. Interval control is used to obtain the current time and to reinitiate this task (DFHTPQ).

6. DFHTPQ communicates with temporary-storage control to retrieve and replace message control records (MCRs) and to purge messages.
Basic mapping support

**DFHTPR (terminal page retrieval program)**

The terminal page retrieval program (DFHTPR) processes messages built by BMS and placed in temporary storage.

Figure 19 on page 163 shows the relationships between the components of page retrieval.
Basic mapping support

Figure 19. Page retrieval program interfaces
Basic mapping support

Notes:

1. DFHTPR can be initiated as a stand-alone transaction (CSPG), or by a user-defined paging command (for example, P/, or 3270 PA/PF keys), or linked to from a BMS conversational operation (SEND PAGE request with CTRL=RETAIN or RELEASE).

DFHTPR performs the following functions:

- Displays the first page of a routed message
- Displays subsequent pages of a message at a terminal for which a SEND PAGE request with CTRL=AUTOPAGE was specified
- Processes paging commands from a terminal
- Processes the CSPG transaction when it is entered at the terminal
- Purges a message displayed at the terminal if the terminal is in display status and other than a paging command is entered at the terminal.

2. DFHTPR is entered from the BMS mapping control program (DFHMCP) to display the first page of a message originated at the terminal if CTRL=RETAIN was specified in the BMS request. DFHTPR reads from the terminal and processes paging commands until other than a paging command is entered.

3. DFHTPR uses storage control to:

- Acquire and free message control blocks (MCBs)
- Free message control record (MCR) storage
- Acquire storage for information and error messages to be sent to the destination terminal and the master terminal
- Free an automatic initiate descriptor (AID) taken off the AID chain
- Acquire and free storage for a route list constructed in response to a COPY command entered at a terminal
- Acquire a TIOA into which to place a device-independent page when performing the COPY function.

4. Temporary-storage control is used to retrieve and replace MCRs and to retrieve and purge pages.

5. Basic mapping support is used to display error and information messages at a requesting terminal, and to send a page to the destination terminal in the COPY function.

6. Task control is used to retain exclusive control of an MCR while it is being updated.

7. DFHTPR communicates with interval control during error processing when a temporary-storage identification error is returned while attempting to retrieve an MCR. Up to four retries (each consisting of a one-second wait followed by another attempt to read the MCR) are performed. (The error may be due to the fact that an MCR has been temporarily released because another task is updating it. If so, the situation may correct itself, and a retry is successful.)

8. Terminal control is used to read in the next portion of terminal input after a page or information message is sent to the terminal when a SEND PAGE request with CTRL=RETAIN was specified.

9. Transient data control is used to send error or information messages to the master terminal.

10. The terminal output macro (DFHTOM) is issued to provide an open subroutine that puts a completed page out to the terminal.
DFHTPS (terminal page scheduling program)

The terminal page scheduling program (DFHTPS) is invoked for each terminal type to which a BMS logical message built with SEND MAP PAGING or SEND TEXT PAGING is to be sent. For each terminal designated by the originating application program, DFHTPR is scheduled to display the first page of the logical message if the terminal is in paging status, or the complete message if it is in autopage status.

Copy books

<table>
<thead>
<tr>
<th>Copy book</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHBMSCA</td>
<td>Defines constants for field attribute values, flags returned by BMS, and character attribute types and values for SEND TEXT. It is usually copied into BMS application programs.</td>
</tr>
<tr>
<td>DFHMCPE</td>
<td>Included in the minimum-function BMS mapping control program DFHMCPES$, and also forms the BMS fast-path module DFHMCX used by both standard and full-function BMS. It is a small, fast, self-contained, limited-function BMS for 3270 displays and printers.</td>
</tr>
<tr>
<td>DFHMCPIN</td>
<td>Included in the standard and full-function versions of the BMS mapping control program, DFHMCPAS and DFHMCP1$ respectively. It contains the code for input mapping.</td>
</tr>
<tr>
<td>DFHMIN</td>
<td>Included in the DFHM32 and DFHMCPE programs. It contains input mapping code for 3270 terminals.</td>
</tr>
<tr>
<td>DFHMSRCA</td>
<td>Defines constants for MSR control. This is usually copied into BMS application programs.</td>
</tr>
</tbody>
</table>

Exits

No global user exit points are provided for this function.

Trace

The following point IDs are provided for basic mapping support, all with a trace level of BM 1:
- AP 00CD, for temporary-storage errors
- AP 00CF, for exit trace
- AP 00FA, for entry trace.

For more information about the trace points, see the CICS Trace Entries. For more information about using traces in problem determination, see the CICS Problem Determination Guide.
Chapter 11. Builders

The builder modules:
• Make the autoinstall process possible (that is, build a terminal control table terminal entry (TCTTE) dynamically).
• Allows new TCT entries to be added on a running CICS system.
• Allow the TCT to be dynamically updated on a running CICS system.
• Allow TCT entries to be deleted on a running CICS system.
• Reduce emergency restart times for those systems that use the autoinstall function. These systems have to take the time to restore and recover only those terminals that were autoinstalled at the time of termination.
• Reduce warm start times for those systems that use auto-install. No auto-installed terminals (except LU6.2 parallel systems are recovered at warm start).
• Reduce shutdown times for those systems using auto-install. Auto-install catalog entries are deleted but the entry in storage is not destroyed during shutdown.

In this section, the term TCTTE is used in a general way to refer to the terminal control table entries for connections (TCT system entries, TCTSEs), mode groups (TCT modegroup entries, TCTMEs), sessions (session TCT terminal entries, TCTTEs), skeletons (TCTSKs), and models.

To build or delete a control block for a particular device, a set of builders is called. The set of builders is specified by a tree structure of patterns, each pattern specifying one builder.

The builder modules (DFHBS*) are link-edited together into the DFHZCQ load module.

On microfiche, the individual DFHBS* modules are listed separately.

Design overview

What is a builder (DFHBS*)?
A builder is responsible for all the actions that can occur on a particular subcomponent of the TCTTE. The term subcomponent means a separately obtained area of storage which is referenced from the TCTTE or a collection of fields in the TCTTE that are logically associated with one another. General terms sometimes used instead of subcomponent are object or node. For example, the NIB descriptor, LUC extension, and BMS extension are all considered to be subcomponents.

Builder parameter set (BPS)
Each time a calling module invokes DFHZCQ for INSTALL, it supplies a builder parameter set (BPS). The BPS describes the device to be defined. The device-type is determined by matching attributes in the BPS with a table of definitions, DFHTRZYT, in module DFHTRZYP.
A BPS consists of a fixed-length prefix, a bit map preceded by its own length, an area for fixed-length parameters preceded by its own length, and three variable-length parameters, BIND, USERID, and PASSWORD. Each variable-length parameter has a 1-byte length field.

TCTTE creation and deletion

This section starts by describing the structure of the main components involved in the process of creating and deleting TCTTEs. Figure 20 is in two halves: the top half shows those components that can initiate the process of collecting all the necessary data or parameters that go toward fully defining a TCTTE, and the bottom half is concerned with how to go about creating the TCTTE after it has the full set of parameters. Thus, all the processes are aiming for the same common interface. This section deals first with the top-level processes that are activated to create or delete TCTTEs; for the time being, assume that after returning from the DFHZCQ interface a TCTTE has been created. (For a more detailed description, see "DFHZCQ and TCTTE generation" on page 171.)

Figure 20. Top-level view of the components participating in TCTTE creation
Component overview

**DFHTCRP**
The DFHTCRP program is responsible for reestablishing the TCTTEs that were in existence in the previous run. There are conceptually three stages of processing in this module:
1. Initialize DFHZCQ. Initialize DFHAPRD. If START=COLD, terminate.
2. Reestablish TCTTEs that were saved on the CICS catalog. If START=WARM, terminate.
3. Call DFHAPRDR to forward-recover in-flight TCTTEs from the system log, if an emergency restart is being performed.

**DFHAMTP**
The DFHAMTP program is used as part of INSTALL processing. It calls DFHTOR, then DFHZCQ.

**DFHZATA and the CATA transaction**
CATA is a transaction that is initiated by the logon exit and causes DFHZATA to run. It is passed the CINIT which is used to deduce the parameters which must be passed to DFHZCQ in order to create a TCTTE.

**DFHZTSP**
The terminal sharing program, DFHZTSP, is used by transaction routing for devices of all types, exclusively so for non-APPC devices.

**DFHZCQ**
The DFHZCQ program supports the INSTALL and DELETE interface that results in the TCTTE being created or deleted. It relies on its callers to supply the complete set of parameters that are to be used to create the TCTTE; that is, it is not responsible for determining parameters for the TCTTE.

**DFHBS* builder programs**
The builders are responsible for creating the TCTTE. The parameters given to DFHZCQ are passed on to the builders. They extract the parameters and set the relevant fields in the TCTTE.

**DFHTBS**
The DFHTBS program is an interpreter that uses a pattern given to it by DFHZCQ to drive the whole TCTTE creation or deletion process according to certain rules.

**DFHAPRDR**
The DFHAPRDR program is the orchestrator of the commitment of TCTTE creation or deletion. It is responsible for driving DFHTBSS and DFHTONR for syncpoints, during cold start and also for recovering in-flight creates or deletes from the system log during emergency restart. It is called by the Recovery Manager, DFHTCRP and DFHAMTP during start-up and directly from DFHTBS (to roll-back an atom).

**DFHTBSS**
The DFHTBSS program is responsible for logging forward recovery records and for updating the catalog as a result of the request initiated by DFHZCQ and actioned by DFHTBS. It is driven by DFHAPRDR.

**DFHTONR**
The DFHTONR program is responsible for logging forward recovery records and for updating the catalog for install or delete requests for TYPETERMS. It is driven by DFHAPRDR.
**Builders**

**DFHZGTA**
DFHZGTA is the module called by DFHBS* and DFHZTSP (for remote system entry sessions) to add or delete index entries for TCTTE entries. It maintains locks on terminal namespaces, and handles calls to TMP to add, quiesce, delete, unlock and unquiesce entries. It is driven at syncpoint or rollback for an atom by DFHAPRDR.

**DFHZCQ and TCTTE generation**

This section describes how a TCTTE gets built and deleted. You need to understand at least one method by which a builder parameter set (BPS) is created; for example, CEDA INSTALL or AUTOINSTALL. A BPS contains all the values necessary for the creation of a TCTTE.

Figure 21 gives a more detailed view of the main components involved in the INSTALL process.

![Figure 21. Major active components in the INSTALL process](image-url)

 Builders

**DFHZGTA**
DFHZGTA is the module called by DFHBS* and DFHZTSP (for remote system entry sessions) to add or delete index entries for TCTTE entries. It maintains locks on terminal namespaces, and handles calls to TMP to add, quiesce, delete, unlock and unquiesce entries. It is driven at syncpoint or rollback for an atom by DFHAPRDR.

**DFHZCQ and TCTTE generation**

This section describes how a TCTTE gets built and deleted. You need to understand at least one method by which a builder parameter set (BPS) is created; for example, CEDA INSTALL or AUTOINSTALL. A BPS contains all the values necessary for the creation of a TCTTE.

Figure 21 gives a more detailed view of the main components involved in the INSTALL process.

![Figure 21. Major active components in the INSTALL process](image-url)
The four-stage process

In summary, the process consists of four stages:

1. **Collecting the parameters** together.
2. **Creating the storage** for the TCTTE and copying the parameters. Note however, that at the end of this stage, a TCTTE has effectively been built. It is still unknown to the rest of the CICS system, that is, the TCTTE name has not been exposed. The modules involved here are DFHTBSB and DFHBS*.
3. **Producing a recovery record**. This is done at syncpoint processing time in the DFHTBSS module. This stage is usually called Phase 1 syncpoint.
4. **Writing or updating the catalog**. Again, this is done in DFHTBSS and is called Phase 2 syncpoint. It is at about this stage that the TCTTE name becomes exposed and known to the rest of CICS.

---

What is DFHZCQRT?

DFHZCQRT is an array of “patterns” where each pattern defines a list of builders that need to be called in order to create this particular type of TCTTE, that is, a pattern is equivalent to a type of terminal. The array entry consists of two parts: information that is private to DFHZCQ, and the pattern that is interpreted by DFHTBS.

What does DFHTBSBP do?

The pattern entry is passed to DFHTBSBP (via DFHTBSB) after it has been found by DFHZCQIS. DFHTBSBP calls each builder identified by the pattern in sequence to create the object for which the builder is responsible. Note that DFHTBSBP knows nothing about the TCTTE; DFHTBSBP merely follows a set of simple rules. It keeps an audit trail of each builder that is called.

What is the RRAB used for?

The audit trail kept by DFHTBSBP is implemented by obtaining a Resource definition Recovery Anchor Block (RRAB) that has some user storage attached to it. As DFHTBSBP calls each builder to perform an action, it adds an “action element” to the RRAB. (See "What is syncpointing?" on page 172.) The address of the RRAB for a UOW is held in the ‘APRD’ recovery manager slot, which ensures that DFHAPRDR will be called at syncpoint. The RRAB stores the action blocks in two types of chains, one for actions that are not part of a named resource definition ‘atom’ and one for actions that are part of a named atom. This later type are chained off a Resource definition Action Name block (RABN). Also held in the RRAB is an indicator set by DFHTOR if DFHTONR should be called at syncpoint (if a typeterm has been installed), and a chain of Resource Definition Update Blocks (RDUB).

What is a resource definition ‘atom’?

Certain resource definitions must be installed or deleted as a single set. These definitions are called a resource definition ‘atom’. CICS installs the members of a RDO group as individual resource definitions, which can fail without causing the other resources to fail except for these atoms, which bear the name of the logical set of definitions. For example:

A connection and its associated sessions
  is named for the connection

A pool of terminals
  is named for the pool of terminals
What is a Resource definition Atom Name block (RABN)?
The RABN is only created for those atoms of resource recovery that are named. It holds the name of the atom, a chain of action elements for the atom, and the recovery outcome of the atom (whether it failed and was backed out, or succeeded and should be committed). DFHTBSB uses the RABN to decide if a session definition should not be installed because the install of the parent connection has already failed, for example. In our auto-install example, if the definition being installed is a parallel connection, there will be a RABN for it from which the action elements are chained.

What is a Resource Definition Update Block (RDUB)?
The RDUB is a record of locks held by a UOW against names in three namespaces:
1. Termids and Sysids
2. Netnames
3. Unique ids (Composed of the Netname of a Terminal Owning Region followed by a period '.' followed by the Termid or Sysid in that TOR)

During the installation, deletion, or replacement of a TCTTE definition the builders DFHBS* obtain locks by calling DFHZGTA. These locks guarantee exclusive or shared access to names in these namespaces. Exclusive access is used to prevent another task from installing another definition with the same name, netname or unique-id while this UOW is trying to install or delete (an action which may have to be reversed). Shared access is used to block another task from deleting an entry that a definition that this task is updating (for example, a system definition name may be locked by a remote terminal definition that refers to it).

RDUBs also exist on a global chain so that other UOWs can easily find out if a particular lock is held.

What is syncpointing?
When DFHTBSBP has exhausted the list of builders, it returns to its caller. Similarly, DFHZCQIS returns to its caller, which could have been autoinstall. However, there is still an audit trail that is attached to the RRAB. It is only when the calling task terminates or issues DFHSP USER or EXEC CICS SYNCPOINT that the next two stages occur.

Syncpoint processing consists of two phases. The first phase (prepare phase) requires the resource manager to write a forward-recovery record to the log. Thus, if the second phase (commit phase) fails to write to the catalog, this recovery record can be used to forward-recover the action on an emergency restart.

DFHTBS
The DFHTBS program is an interpreter that uses a pattern given to it by DFHZCQ to drive the whole TCTTE installation or deletion process according to certain rules.

DFHAPRDR
DFHAPRDR is invoked by recovery manager if the ‘APRD’ RM slot is non-zero. This slot contains the address of the RRAB for this UOW if any resource definition has taken place. It is also called by DFHTBS directly if an atom needs to be rolled-back or to commit an atom during Cold Start. DFHAPRDR examines the RRAB and chooses whether to call DFHTBSS, DFHTONR and DFHZGTA for each phase of syncpoint or individual atom commitment.

If either DFHTBSS or DFHTONR have records to log/catalog, DFHAPRDR calls the recovery manager to request that a record is written to the catalog noting that a forget record will be written once syncpoint completes. The purpose of this call is
that if CICS should fail between the start of syncpoint phase 2 and the end, on an emergency restart recovery manager will call DFHAPRDR with the log records for this UOW so that they can be re-applied to the catalog, and the TCTTE entry or entries can be re-built.

**DFHTBSS**
The DFHTBSS program is responsible for performing the correct recovery actions for each atom and UOW at syncpoint (or during the rollback of an individual atom). It writes forward recovery records to the system log and updates the catalog during phase 1 and phase 2 of syncpoint respectively. It is directly driven by DFHAPRDR.

The purpose of the builder (DFHBS*) modules is to build a TCTTE, TCTSE, and TCTME and its associated control blocks. A TCTTE is built for terminals only; a TCTSE and TCTME are built for both LU6.1 with MRO and LU6.2 single sessions; all three are built for LU6.2 parallel sessions. DFHTBSS is invoked by DFHAPRDR with a parameter list that indicates whether this call is for an individual atom or for syncpoint and which phase is in force. For phase 1, it uses the action blocks audit-trail to recall each builder. It asks each builder to supply the address and length of the subcomponent so that it can create a single record containing a copy of each component as a list; that is, the first part of the record contains a copy of the object created by the first builder in the sequence, the second part contains a copy of the object created by the second builder, and so on until the audit trail list is finished. This record is then written to the system log as a forward recovery record.

When DFHTBSS is reentered for the second phase (again a parameter on the call by DFHAPRDR), it uses the record created in the first phase as the record that is written to the catalog. During this stage, each builder is called to tidy up after the object for which it is responsible; for example, for the TCTTE itself, it puts the TCTTE in service.

Again note, DFHTBSS only implements a set of rules.

**DFHTONR**
DFHTONR is responsible for writing catalog records for TYPETERMs. It is called by DFHAPRDR.

**DFHZGTA**
DFHZGTA is the module that is called by DFHBS* modules to add index entries for TCTTE entries so that they can be located quickly either by DFHZLOC, DFHZGTI or in VTAM exit code. It calls DFHTMP services. It obtains and releases locks using the RDUB blocks, and at syncpoint is responsible for releasing all TMP locks and unquiescing any TMP entries that were quiesced by DFHBS* modules.

**Summary**
- In overview, the process consists of four stages: parameter collection, obtaining and initializing, phase 1 recovery record and logging, and phase 2 catalog record.
- A builder contains TCTTE specific code.
- DFHTBSS* modules implement the abstract rules for creating generic “objects”.
- DFHZCQRT contains patterns that define what builders are to be used to build the TCTTE.
- Syncpoint processing consists of two stages (prepare and commit).
Builders

- DFHAPRDR is responsible for orchestrating the syncpoint process for all of resource definition recovery.
- DFHTBSS is driven by DFHAPRDR using the audit trail produced by DFHTBSB.
- DFHTONR is driven by DFHAPRDR if any TYPETERMs were installed.
- DFHZGTA is driven by DFHAPRDR if any locks need to be released.

Example of an autoinstall
Consider the following: a terminal operator has logged on to the system and is being autoinstalled. The CATA transaction is responsible for collecting together the parameters required for the DFHZCQ INSTALL.

The process continues from the point where the DFHZCQ INSTALL is issued from CATA.
<table>
<thead>
<tr>
<th>Step</th>
<th>DFHZCQ</th>
<th>DFHZCQIS</th>
<th>DFHTBS</th>
<th>DFHTBSB</th>
<th>DFHBS*</th>
<th>DFHTBSS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>A call has been made to cause an install to occur. DFHZCQ ensures that other related modules are already loaded.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td>Calls the install-specific module (given in the parameter block passed to DFHZCQ).</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td>Performs various checks on the parameters passed by the caller of DFHZCQ.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td>Finds a pattern in DFHZCQRT that matches with information given in the parameters.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td>Calls DFHTBS with the pattern and parameters.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td>DFHTBS only routes the request to DFHTBSB, so is omitted from further discussions.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td>Checks that a valid pattern has been passed.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8.</td>
<td>Creates the RRAB which gets attached to the 'APRD' Recovery Manager slot.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9.</td>
<td>Calls the next builder as defined by the pattern.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.</td>
<td>Each builder creates its section of the TCTTE.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11.</td>
<td>Adds an action element to the RRAB giving information about this particular builder.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12.</td>
<td>Repeats steps 9, 10, and 11 until the pattern is finished.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13.</td>
<td>Tidies up the RRAB and returns.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15.</td>
<td>If the return code was 'OK', returns the address of the hidden TCTTE.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16.</td>
<td>Returns.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 22. Flow of control for a build (Part 1 of 2)
Patterns, hierarchies, nodes, and builders

Patterns were introduced in the previous section. This section examines in detail what they look like. To achieve this, several terms have to be explained.

What is a hierarchy?
In this context, “hierarchy” is another word for tree. The structure of the TCTTE can be thought of as a tree: at the top node is the TCTTE itself, containing pointers to lower-level nodes.

Figure 23 on page 177 shows the master node as the TCTTE, with subnodes connected to it (BMS extension, special features extension, and so on).
As a result of this structure, it can be seen that the creation process must follow several rules. For example, the storage for the master node has to be obtained before pointers to subnodes are saved in it.

What is a pattern?
The objective of a pattern is to reflect or represent the hierarchy as described above. Figure 24 outlines the shape of a pattern. For each of the nodes in Figure 23, there is a pattern. Starting with the TCTTE (the master node), there is a master pattern. B1offset references the subpattern for the BIND image node; B2offset references the subpattern for the BMS extension node; B3offset and B4offset reference the subpatterns for user area and SNTTE subnodes respectively. In total, there are five patterns: the master pattern and four subpatterns—so what is meant by pattern above was really a collection of patterns.

Note that each pattern contains the address of a builder, so we could represent the TCTTE structure as:
The purpose of the builders

The purpose of the builders is to centralize the major functional code for creation and deletion of the nodes associated with the TCTTE. Figure 24 on page 177 and Figure 25 show how the patterns refer to the builders; the pattern is exploited by the DFHTBS* code to activate the relevant builder function. For example, DFHTBSBP, when given a pattern, extracts the address of the builder and invokes the BUILD function belonging to the builder.

How does DFHTBSBP do its work?

First, you must examine more closely the structure of a builder in Figure 26 on page 178.
Remember that the pattern references a builder. In fact, it references a stub, the first word of which points to a table (BSH_EP_TABLE), and is followed by code that is responsible for enacting the entry as required by the caller. For example, if the caller wanted to call BUILD, a call would be made to the stub with value 1. The stub would extract the offset to the build code from the BSH_EP_TABLE, and perform the call.

Thus, making a call from DFHTBS* to DFHBS* is relatively simple: all that is needed is the function number (1 for BUILD, 2 for DESTROY, ...), a call to the stub, and the pattern.

Figure 26. The builder stub
 Builders

Summary

- The TCTTE is structured as a hierarchy with a master node (the TCTTE itself) and subnodes (BIND image, BMS extension, and so on).
- Patterns mimic this hierarchy and consist of a master pattern which refers to subpatterns.
- In turn, each pattern points to a builder: the master pattern refers to the master builder and the subpatterns refer to the sub-builders.
- Builders centralize the major creation and deletion functions associated with the node for which they are responsible.
- The invocation (or activation) of the builder functions is performed under the strict control of the DFHTBS* modules.
- The order of invocation is totally determined by the structuring of the patterns.

The DELETE process

By examining the hierarchy (see Figure 23 on page 177), you can see that there are certain rules that have to be established. Firstly, you should check that the TCTTE and its subcomponents are quiesced, that is, there is no activity in progress. And secondly, and perhaps more obviously, the top node must not be the first object to be freed. From this, you can derive two basic rules, or “functions”, that must be supplied by any DFHBS*:

UNREADY
For all nodes associated with the master node. Ensures that no activity is occurring; for example, that a CLSDST is not in progress. It must also achieve exclusive ownership of the object; for example, ZGTA QUIESCE ensures no locates on the given TCTTE succeed and that no other UOWs can install another similarly named object until syncpoint. Further, it initiates the ZGTA DELETE which does a TMP DELETE to remove the entry.

DESTROY
Lower objects first. (See “What about the “lower objects first” rule?”) Frees the storage belonging to the node.

What about the “lower objects first” rule?
Figure 27 on page 181 tries to add meaning to the descriptions of the UNREADY and DESTROY functions. As each builder is called (as determined by the master pattern), DFHTBSD records an audit trail of called builders. However, the audit trail is managed slightly differently for the delete process, to guarantee order of processing by DFHTBSS at phase 2 time. For further information, see Completing the process description on page 183.
Figure 27. Major active components in the DELETE process
## Example of a reinstall

<table>
<thead>
<tr>
<th>Step</th>
<th>CEDA</th>
<th>ZCQIS</th>
<th>TBSB</th>
<th>BS*</th>
<th>ZCQDL</th>
<th>TBSD</th>
<th>BS*</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. Reads the CSD and converts the definition into a builder parameter set (BPS).
2. Issues a DFHZCP INSTALL passing the BPS.
3. Using the resource type code in the BPS, searches the DFHZCQRT table for the associated pattern.
4. Calls DFHTBSB passing the BPS and the pattern.
5. Checks the pattern and creates a resource definition.
6. Using the pattern, calls the CHECKSET entry point of the associated builder.
7. The master builder does a ZGTI LOCATE to check whether the TCTTE already exists.
8. A TCTTE is found to exist, so the builder issues DFHZCP DELETE passing the address of the old TCTTE.
9. When a TCTTE is created, its position within the DFHZCQRT table is saved in the TCTTE. This value is now used to find the pattern associated with this TCTTE.
10. Calls DFHTBSD passing the object to be deleted and the pattern.
11. Extends the audit trail so that information about this delete can be recorded.
12. Calls the UNREADY entry of each builder.
13. Each builder checks whether its part of the TCTTE is being used. (A builder vetoes if it is.) Calls ZGTA QUIESCE and ZGTA DELETE to lock and remove the index entries.
14. Updates the audit trail for each called builder.
15. Returns.
16. Returns.
17. Checks return code (that is, no builder vetoed the UNREADY).
 Builders

<table>
<thead>
<tr>
<th>Step</th>
<th>CEDA</th>
<th>ZCQIS</th>
<th>TBSB</th>
<th>BS*</th>
<th>ZCQDL</th>
<th>TBSD</th>
<th>BS*</th>
</tr>
</thead>
<tbody>
<tr>
<td>18.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>33.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>34.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>35.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>36.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>37.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>38.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>39.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>41.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returns.

Checks return code and recalls the builder at the BUILD entry point passing the BPS.

Obtains some storage and copies the parameters from the BPS.
Uses ZGTA ADD calls to lock and add index entries

Repeats for each builder in the set of patterns.

Tidies up the RRAB and returns.

Records the position within DFHZCQRT that enabled you to get the pattern.

Returns.

Checks the return code and issues DFHSP USER.

Note: At this stage there are two TCTTEs: the old one that was UNREADY and the new one.

DFHTBSS is entered for the first time (phase 1). The audit trail consists of two parts (A and B). Part A contains the list of builders involved with the UNREADY; part B contains the list of builders that created the new TCTTE.

Writes a recovery record for Part A indicating that a delete is about to take place in phase 2 to the system log.

Creates a recovery record from Part B which represents the new TCTTE to be built.

Calls each builder asking for its subcomponent (FLATTEN).

Returns an address and length.

Concatenates each subcomponent into the recovery record.

Writes the recovery record to the system log.

Returns (end of phase 1).

Reenter for phase-2 processing.

Processes Part A, calling the DESTROY entry for each builder.

Each builder frees its part of the old TCTTE.

Processes Part B of the audit trail.

Writes the recovery record to the catalog.

Calls the READY entry point for each builder on the audit trail.

Each builder does any tidying up that needs to be done.

Returns.

Figure 28. Flow of control for a reinstall (Part 2 of 2)
Completing the process description

To complete the description of the creation and deletion process, two further functions must be described: CONNECT and READY.

CONNECT

Figure 23 on page 177 shows the TCTTE hierarchy. All that has happened at build time is that the separate parts of the TCTTE have been obtained. Access to these subcomponents is achieved by referencing pointers that are held in the TCTTE. So the CONNECT builder entry point is used to join the subcomponent to the TCTTE.

READY

The READY builder entry point is provided to enable any final tidying up that may be required at the end of the build process. For example, if the TCTTE has the AUTOCONNECT option, a SIMLOGON is initiated from this entry point. In general, this entry point is rarely used.

The creation/deletion state machine

Figure 29 shows the symmetry between the various builder functions.

![State Diagram]

The starting point can be either state 5 (installing a TCTTE) or state 1 (deleting a TCTTE). Thus, if several TCTTEs had been successfully built, but the last one resulted in an error, we would end up in state 4. If it were not for the last one, we would have ended up in state 3. So the caller is returned an error response, and issues a DFHSP ROLLBACK. This causes DFHTBSS to call the DESTROY function of the builders for all elements on the audit trail—even for those that were “successfully” built in this atom, or UOW. Thus, an install of a atom can be perceived as one complete unit. During the DESTROY process, if the atom is being rolled-back, the builders call ZGTA QUIESCE and ZGTA DELETE to remove index entries for the new TCTTE. Likewise during the READY process, if a delete is being rolled back, the builders call ZGTA ADD to re-instate index entries for the TCTTE.
The hierarchy and its effect upon the creation process

Summary so far
- Object creation is a four-stage process.
- It is controlled by a pattern.
- Each pattern refers to a builder.
- Each builder is responsible for a subcomponent of the TCTTE.
- Builders have a number of procedural entry points:
  - BUILD
  - CONNECT
  - DESTROY
  - READY
  - UNREADY.
- These entry points are called under the control of the DFHTBS components.

This section now looks in greater detail at how the control of the builder calling process is implemented. To do that, you need to understand in greater detail the structure of the hierarchy, and the way the DFHTBS components interpret that structure.

Figure 30 shows a more general hierarchy. Node 1 can be considered as a master node: it is at the top of the tree and has two subnodes (2 and 3). However, you could say that node 2 and its subnodes are also a tree: node 2 is the master node, and nodes 4, 5, and 6 are the subnodes. Similarly, with node 3: it has subnodes 7, 8, 9, and 10.

The DFHTBS components exploit the idea that a tree consists of a node with trees below it. In fact, DFHTBSBP uses recursion to access the tree of patterns.

Recursion
This section demonstrates how recursion is used to process a much simpler structure than that given in Figure 30. The example shown in Figure 31 on page 188 is for the DFHTBSBP program, which has the following parameters:

Input: PATTERN, HIGHERNODE, and BUILDER
Inout: AUDITTRAIL
Output:

NODE and RESPONSE.

The following list outlines the flow in DFHTBSBP. The step references refer to steps in this list.

1. Add and initialize an action to the AUDITTRAIL (this is used later in steps 5 and 11).
2. Using parameter PATTERN, find the address of the associated builder.
3. Call the builder stub with function number 1 (for BUILD) with the following parameters:
   Input: HIGHERNODE and BUILDER
   Output: NODE.

The builder uses the BUILDER parameters to create its specific object. Storage is obtained and the parameters are copied into it.

4. Check that the response from the build is ‘OK’.
5. Copy the address of the output parameter NODE into the AUDITTRAIL action.
6. Process all the subpatterns that may be attached to your pattern
7. Get the next subpattern Pn.
8. Call DFHTBSBP with the following parameters:
   Input: Pn, NODE, and BUILDER
   Inout: AUDITTRAIL
   Output: SUBNODE and SUBRESPONSE

   Note: In this step, you call yourself again, passing NODE. At the next level of recursion, this appears as HIGHERNODE.
9. Stop when the last pattern is processed.
10. Call the builder stub with function number 5 (for CONNECT) with the following parameters:
    Input parameters:
    NODE
    Inout parameters:
    HIGHERNODE

The builder’s CONNECT entry point now places the address as given by NODE into an offset of HIGHERNODE.
11. Finally, place the address of the pattern into the AUDITTRAIL action.
Consider the following simplified version of the hierarchy as given in Figure 31. The step references refer to steps in the list in the section “Recursion” on page 186.

1. Start with pattern P1. Call its associated builder (step 3 on page 187). This creates node N1.

2. All the patterns below P1 are processed, the first of which is P2.

3. Call DFHTBSBP passing P2, N1, BUILDER parameters, and others:
   a. Using the passed pattern (now P2), call the builder. This creates node N2.
   b. Process all patterns below P2; there are no subpatterns, so steps 6 through 9 on page 187 are not performed.
   c. Call the CONNECT entry of the builder, passing higher node N1 and the node just created, N2. This makes N1 point to N2.
   d. Return to caller.

4. Get the next pattern, P3.

5. Call DFHTBSBP passing P3, N1, BUILDER parameters, and others:
   a. Using the passed pattern (now P3), call the builder. This creates node N3.
b. Process all patterns below P3; there are no subpatterns, so steps 6 through 9 on page 187 are not performed.

c. Call the CONNECT entry of the builder passing in higher node N1 and the node just created N3. This makes N1 point to N3.

d. Return to caller.


7. Call the builder associated with P1 to connect node N1 to HIGHERNODE. (This is zero because there is no higher node. Usually, a master builder’s CONNECT function either does nothing or adds the TCTTE name and address into the table management tables.)

ROLLBACK

What happens when an error occurs during the install process? An example of this would be when one TCTTE within a group is still in service when a CEDA COPY command is being processed for the group with the REPLACE option specified. “Example of a reinstall” on page 183 shows such a replace operation. The builders for the existing TCTTE are called (UNREADY) in order to check that the DELETE (FREEMAIN) can proceed. Thus, the audit trail refers to all called builders.

If the “total vote” from all the UNREADY builder calls indicates OK, the build proceeds for the new TCTTE that is to replace the existing one. Thus, at the end of the process, the audit trail consists of a list of references to builders associated with the old TCTTE, and a list of references to builders for the new TCTTE (lists A and B).

Consider the case when the group contains definitions for three TCTTEs, and a VETO occurs for the last one. This means that there is an audit trail for A1, B1, A2, B2 for which there was success, and list A3 for the unsuccessful UNREADY for the third TCTTE.

The failure condition is returned to the caller (CEDA), which then issues a DFHSP ROLLBACK.

Recovery Manager invokes DFHAPRDR which in turn invokes the DFHTBSS module, with a parameter that indicates a rollback is required. Thus, the “A” lists are processed, and all the READY entry points of the builders are called. Then the “B” lists are processed, and the DESTROY builder entry is called to free the storage obtained for the supposedly new TCTTEs.

To summarize, the rollback operation for UNREADY is READY, and the one for BUILD is DESTROY.

Catalog records and the CICS global catalog data set

Overview

The fourth stage of the process is to produce a catalog record that is written to the CICS global catalog data set. This record is used on a subsequent restart to recreate the TCTTE, but in a different way from the “Build” process described above. A CEDA INSTALL means that the TCTTE lives across CICS restarts, avoiding the necessity of rerunning the install.

A RESTORE from the CICS catalog is a faster operation than a CEDA INSTALL because there is no conversion of the CSD definition to a builder parameter set, and less I/O involved.
In summary, a catalog record is produced by recalling each of the builders asking for the address of the data that they want to be recorded on the catalog. Each subcomponent of the TCTTE is then copied and concatenated into one record, which is then written to the catalog. This process is known as FLATTEN.

A CATALOG call is made when significant events change the state of a TCT entry which would be needed on a subsequent emergency restart. An example is the recording of the membername of a generic VTAM resource connection when a bind has occurred for the first time.

On the restart, the record is read from the catalog, and presented back to each of the original builders. Each builder performs a GETMAIN, and copies its section of the recovery record into the acquired storage. This process is known as UNFLATTEN.

At shutdown, auto-installed entries are removed from the catalog with an UNCATALOG call (if they were cataloged because AIRDELAY=0). This drives DFHTBS and the builders to produce similar records to those for a DELETE call, but only to take action to delete the catalog record. This is significantly more efficient than calling the builders to DELETE each entry, as the copy in storage is left untouched.

The key and the recovery record
When the build process in DFHTBSBP has finally finished, this module makes a call to the master builder at the MAKEKEY entry point. The builder produces a key that is used to index the associated recovery record. (See Figure 32.)

This information is placed on the audit trail so that it can be picked up by DFHTBS. It consists of two parts:
1. Information that allows access to the catalog
2. The recovery record header.

![Figure 32. The recovery record](image-url)
More about the audit trail

Figure 33 shows the layout of an audit trail. Internally it is known as an action block, which consists of action elements. As each builder is invoked by DFHTBSBP or DFHTBSDP, an action element is appended to the action block. Each element has a reference to a pattern (PATT). This is to allow DFHTBSS to enter the associated builder at the READY or DESTROY entry points.

CCRECP contains the address of the recovery record header. Only one of these is produced as a direct result of the MAKEKEY call to the master builder. All other action elements have their CCRECP set to zero.

DFHTBSS and the FLATTEN process

During phase-1 syncpoint processing, DFHTBSS searches the action elements for a nonzero CCRECP. On detection, it calls DFHTBSLP, passing the reference to the pattern as given in the action element.

The storage “segments” are returned to DFHTBSSP which extracts the address and length from each segment and copies them into the recovery record.

The RESTORE process

The recovery record header contains the pattern name which is used to find the master pattern in DFHZCQRT. This is then passed to DFHTBSR to drive the recovery process by calling each builder’s UNFLATTEN entry.

Each segment is extracted from the recovery record and is passed to the associated builder’s UNFLATTEN entry point. These routines usually obtain some storage and copy the segment into it.
Control blocks

Builder modules all use both LIFO and a builder parameter set (BPS), which are passed between the CSECTs (DFHBS* modules). See [Builder parameter set (BPS)] on page 167 for further information about the BPS.

Terminal storage acquired by the builders

The following terminal storage is acquired by the builders:

<table>
<thead>
<tr>
<th>Control block field</th>
<th>Description</th>
<th>Storage manager subpool</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCTSE</td>
<td>Terminal control table</td>
<td>ZCTCSE</td>
</tr>
<tr>
<td>TCTME</td>
<td>Terminal control table mode</td>
<td>ZCTCME</td>
</tr>
<tr>
<td>TCTTE</td>
<td>Terminal control table</td>
<td>ZCTCTTEL (large TCTTEs)</td>
</tr>
<tr>
<td></td>
<td>terminal entry</td>
<td>ZCTCTTE (medium TCTTEs)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ZCTTE (small TCTTEs)</td>
</tr>
<tr>
<td>TCTENIBA</td>
<td>NIB descriptor</td>
<td>ZCNIBD</td>
</tr>
<tr>
<td>TCTEBIMG</td>
<td>BIND image</td>
<td>ZCBIMG</td>
</tr>
<tr>
<td>TCTTECIA</td>
<td>User area</td>
<td>ZCTCTUA</td>
</tr>
<tr>
<td>TCTTESNT</td>
<td>Signon extension</td>
<td>ZCSNEX</td>
</tr>
<tr>
<td>TCTELUCX</td>
<td>LUC extension</td>
<td>ZCLUCEXT</td>
</tr>
<tr>
<td>TCTETEBA</td>
<td>BMS extension</td>
<td>ZCBMSEXT</td>
</tr>
<tr>
<td>TCTETEPA</td>
<td>Partition extension</td>
<td>ZCTPEXT</td>
</tr>
<tr>
<td>TCTTECCE</td>
<td>Console control element</td>
<td>ZCCCE</td>
</tr>
</tbody>
</table>

TCTTE layout

![TCTTE layout diagram](image)

Figure 34. TCTTE layout

Formatted dumps give the TCTTE first, followed by its supporting control blocks.

Terminal definition

CEDA DEFINE puts a definition on the CSD. The definition is in the form of a CEDA command.

CEDA INSTALL reads the definition from the CSD, calls the builders and builds the definition in CICS DSA, and updates the CICS global catalog data set for future recovery.

EXEC CICS CREATE builds the same record that would be obtained from the CSD and then calls the builders just like CEDA INSTALL.
EXEC CICS DISCARD calls the builders with a pointer to the TCTTE entry that is to be deleted. The builders then freemain the TCTTE, remove index entries and the catalog record.

**Modules**

DFHZCQ handles all requests for the dynamic add and delete of terminal control resources. It contains the following CSECTs:

- DFHBS1B3
- DFHBSZM
- DFHBSTP3
- DFHBSTZ1
- DFHBS1Z1
- DFHBSZP
- DFHBSTS
- DFHBSTZ2
- DFHBS1Z3
- DFHBSZR
- DFHBSTT
- DFHBSTZ3
- DFHBSMIR
- DFHBSZS
- DFHBSTZ
- DFHBSXGS
- DFHBSMPP
- DFHBSZ6
- DFHBSTZA
- DFHBSZZ
- DFHBSM61
- DFHBST
- DFHBSTZB
- DFHBSZZS
- DFHBSM62
- DFHBSTB
- DFHBSTC
- DFHBSZZV
- DFHBSS
- DFHBSTBL
- DFHBSTZD
- DFHZCQCH
- DFHBSSA
- DFHBSTB3
- DFHBSTZH
- DFHZCQL
- DFHBSSF
- DFHBSTC
- DFHBSTZL
- DFHZC1N
- DFHBSSF
- DFHBSTD
- DFHBSTZ0
- DFHZC1Q
- DFHBSSS
- DFHBSTE
- DFHBSTZP
- DFHZC1S
- DFHBSSZ
- DFHBSTH
- DFHBSTZR
- DFHZCQT
- DFHBSSZ6
- DFHBSTI
- DFHBSTZS
- DFHZCQR
- DFHBSSZE
- DFHBSTM
- DFHBSTZV
- DFHZCQRT
- DFHBSSZL
- DFHBSTO
- DFHBSTZ
- DFHZCQ00

**Note:** The term “node” refers either to a TCTTE or to one of its subsidiary parts, such as the NIB descriptor.

Subroutines that are found in the builders:

**BSEBUILD**

**BUILD:** Create the node. For example, obtain the shared storage for the node.

**BSECON**

**CONNECT:** Connect the higher node to the lower. For example, make the TCTTE point to the NIB descriptor.

**BSEDESTR**

**DESTROY:** Abolish a deleted node. For example, free the storage removed from TMP’s chains.

**BSEFINDF**

**FINDFIRST:** Find the first subsidiary node of a higher node. For example, BSFINDF(TCTTE) returns the NIBD being built.

**BSEFINDN**

**FINDNEXT:** Find the next subsidiary node of the one just found. For example, return the address of the next model TCTTE.

**BSEFLAT**

**FLATTEN:** Build the catalog or log record segment for each part of the TCTTE. This is passed back to the caller to create a complete “flattened” TCTTE.

**BSEMAKEY**

**MAKEKEY:** Create a key that is used to write out the new node to the global catalog.

**BSENQIRE**

**ENQUIRE:** The converse of BUILD, it creates a BPS from a TCTTE. The BPS can then be shipped to another system.
Builders

BSEReady
READY: Make a node ready to use. For example, add to TMP’s chains.

BSEReset
RESET: Build the TCTTE from the CICS global catalog. (RESET is a cut-down version of UNFLATTEN.)

BSEUnflatten
UNFLATTEN: Build the TCTTE from the CICS global catalog.

BSEUnready
UNREADY: Check that a node can be deleted. For example, ensure that no AIDs are queued on a TCTTE before deleting.

Not all subroutines are found in all builders. Certain subroutines are required, but do nothing other than return to the caller. The subroutine names are the same in each builder.

Module entry
Consider a module entry to be a router that does some housekeeping and then branches to the appropriate subroutine:

• Enter the builder at offset X’18’.
• The first X’17’ bytes are taken up by the standard DFHVM macro expansion.
• Save DFHTBS’s registers (DFHTBS calls each builder).
• Save the first two entries in the parameter list:
  1. The address of LIFO storage
  2. The index number of the subroutine to call.
• Increase the value of register 1 by 8 to get past the first two entries.
• Branch to the appropriate subroutine of the builder using the index number passed.
• Return from the builder subroutine.
• Restore registers.
• Return to DFHTBS.

Subroutine entry

• Register 1 points to the parameter list.
• Store Register 14 (return address) at Register 2 + X’nn’ (varies by entry point).
• Store the parameter list into Register 2 + X’nn’ (varies by entry point).
• The length of the parameter list varies.

Subroutine exit (return to module entry)

• Exit from the subroutine only through an “official” exit point.
• The exit point is usually the end of the subroutine.
• The end of the subroutine is indicated with “*/BUILD */”.
• In some cases, the end of the subroutine branches back to the exit point somewhere within the subroutine.
• Return (BR R14) from within the subroutine.
• Reload Register 14 from Register 2 + X’nn’ and return to caller.
Patterns

In DFHZCQRT, a series of patterns define the flow through the builder modules. (See Figure 35.) For each kind of terminal, there is a different pattern.

If installing, DFHZCQIS selects the pattern and calls DFHTBS (table builder service). If deleting, DFHZCQDL does the selection.

DFHTBS interprets the pattern and calls each builder that the pattern calls out. DFHTBS knows nothing about the terminal or whether you are installing or deleting. It simply does what the pattern tells it to do.

DFHTBS passes a BPS as it calls each builder. The BPS allows one builder to be used for many different kinds of terminals. For example, DFHBSTC obtains the user area for all terminal types. The BPS contains the length to be obtained.

Calling sequence of builders for a 3277 remote terminal

1. DFHZCQRT contains a series of comments followed by the patterns. The comment appears as:

```/ ************* /
/* 3277 REMOTE */
/ ************* /
```

2. Shortly afterwards is a Declare (DCL) followed by a level-1 name:

```DCL 1 P145002 STATIC```

This is the name of the pattern that drives the build process for a 3277 remote terminal.

- DFHBSTZ is indicated to be the first builder called.
- One pattern is used to drive the building process.
- 18 subpatterns are to be used.
- Three of these 18 subpatterns each call one additional pattern.
- The terms “pattern” and “builder” mean the same thing. Therefore:

```
DFHBSTZ + DFHBSxx + DFHBSxx = 22
(1) + (18) + (3) = 22
pattern + sub- + sub-sub- = 22
patterns patterns
```

Thus we have to go through 22 builder modules to build a 3277 remote terminal.

3. Go to the cross-reference at the back of the dump and find where P145002 is defined in assembler language. Go to that address.
Builders

4. This states that the first builder to be called is DFHBSTZ. This is the main one.
5. Drop down to the 2-byte fields that follow: these state the names of the builders that are to be called, in sequence (18 should be listed).
6. The first one is IAATZ1 which does not sound familiar:
   - Go to the cross-reference at the back of the dump, look up IAATZ1, and go to where it is defined.
   - You see that this is DFHBSTZ1.
   - You can also see a close resemblance between IAATZ1 and DFHBSTZ1, but do not count on this to be always true.
7. Now you know that the second builder to be called is DFHBSTZ1.
8. The next two builders to be called are IAATCV (DFHBSTV) and IAATCB (DFHBSTB).
9. The fifth builder to be called according to the pattern needs to be looked at:
   - The pattern says that IACTZ3 should be called.
   - When you go to where IACTZ3 is defined, you find that this is DFHBSIZ3.
   - You also see that DFHBSIZ3 calls IAATM.
   - Look up IAATM and you see that it is DFHBSTM.
   - This is a sub to a subpattern, and this is how nesting of builder calls occurs.
   - Thus, DFHBSIZ3 calls DFHBSTM when building a local 3277.
   - DFHBSTM accounts for one of the “other” three mentioned in step 2.
10. If you continue through this pattern, you can identify the names of the 22 builders that would be called to build a 3270 local TCTTE.

Here is the complete list, in order, of the builders that are called:

1  DFHBSTZ  12  DFHBSTH
2  DFHBSTZ1  13  DFHBSTI
3  DFHBSTZV  14  DFHBSTS
4  DFHBSTZB  15  DFHBSTT
5  DFHBSIZ3  16  DFHBSTZA
6  DFHBSTM  17  DFHBSTP3
7  DFHBSTB  18  DFHBSSZ
8  DFHBSTB3  19  DFHBSTB3
9  DFHBSTO  20  DFHBSTZE
10 DFHBSTC  21  DFHBSSZV
11 DFHBSTE  22  DFHBSTZ3

A look at “Pattern Trace” supports this flow. Note that the first ZCP TBSB(P) BUILD and its matching return (the return has no builder suffix) should be ignored.

Builder parameter list

As each builder is called by DFHTBS, a parameter list is passed. Unique data is passed to enable one builder module to be called for a variety of terminal types. The length of the builder parameter list is fixed for each kind of subroutine; for example, the parameter list passed to BSEBUILD is always X'23' bytes long, regardless of the builder involved.

<table>
<thead>
<tr>
<th>Subroutine</th>
<th>Length of parameter list</th>
</tr>
</thead>
<tbody>
<tr>
<td>BSEBUILD</td>
<td>23</td>
</tr>
<tr>
<td>BSECON</td>
<td>13</td>
</tr>
<tr>
<td>BSEDESTR</td>
<td>7</td>
</tr>
<tr>
<td>BSEMAKEY</td>
<td>8</td>
</tr>
<tr>
<td>BSEReady</td>
<td>3</td>
</tr>
<tr>
<td>BSEUNRDY</td>
<td>17</td>
</tr>
</tbody>
</table>
When the builders are called

Builders are called during:
- Cold start
- Warm start
- Emergency restart
- After emergency restart
- Autoinstall logon and logoff
- APPC autoinstall
- CEDA INSTALL
- EXEC CICS CREATE
- EXEC CICS DISCARD
- Transaction routing
- Non-immediate shutdown.

Cold start
- Read information from the CSD and call builders to build RDO-defined terminals.
- Load in DFHTCT for non-VTAM terminals. Builders are not called.

Warm start

Note: A warm start is identical to an emergency restart from the builders perspective. The only difference is that Recovery Manager has no forward-recovery records to pass to DFHAPRDR.
- Read information from the global catalog and call builders to restore RDO-defined terminals.
- Load in DFHTCT for non-VTAM terminals. Builders are not called.

Emergency restart
- Read information from the global catalog and call builders to restore RDO-defined terminals.

Note: Auto-installed terminals will not have a catalog entry if AIRDELAY=0
- Recovery Manager calls DFHAPRDR which calls the builders to restore in-flight terminals installs from the system log.
- Load in DFHTCT for non-VTAM resources. Builders are not called.

After emergency restart
Delete autoinstalled terminals after the time period has expired as specified in the AIRDELAY parameter (if the user has not logged back on before then).

APPC autoinstall
- Inquire on the model supplied by the autoinstall user program
- Install an APPC connection created from the above inquire.

Autoinstall logon and logoff
- Logon: Install terminal entry using model entry in the AMT.
- Logoff: Delete terminal entry.
CEDA INSTALL
Install VTAM terminal resources. (There is no builder process for CEDA DEFINE or ALTER.)

EXEC CICS CREATE
Install VTAM terminal resources.

EXEC CICS DISCARD
Delete VTAM terminal resources.

Transaction routing
If a TCTTE is defined as shippable, its definition is shipped to the remote system and installed there. The definition is obtained by an INQUIRE call to the builders in the Terminal Owning Region and built with an INSTALL call in the Application Owning Region.

Shutdown
Delete autoinstalled terminals from the catalog (if they had entries, and are not LU6.2 parallel connections). On a warm start, therefore, autoinstalled terminals are not recovered.

Diagnosing problems with the builders
When working on a problem associated with a builder (for example, abend or loop), it may be helpful to ask yourself the following questions:

- Why am I in a DFHBS* module? Am I doing CEDA GRPLIST install, CEDA GROUP install, autoinstall, logon, logoff, catalog, uncatalog, create or discard?
- What is the termid/sysid of the terminal I am working with (the one I am installing, deleting, cataloging or uncataloging)?
- Is this resource part of an resource definition atom?
- How is this terminal defined?
- Are there any messages associated with this terminal?

Exits
No global user exit points are provided for this function.

Trace
The following point IDs are provided for the DFHZCQxx modules:
- AP FCB0 - FCBF, for which the trace level is 1.

The following point IDs are provided for the DFHTBSx modules:
- AP FCC0 - FCC9, for which the trace level is 1.

The following point IDs are provided for the DFHTBSxP modules:
- AP 0630 - 0644, exception trace.
- AP FCD0 - FCD9, for which the trace level is 1.
- AP FCDA - FCDB, for which the trace level is 2.

The following point IDs are provided for the DFHTBSS module:
- AP 0620 - 0621, for which the trace level is 1.
- AP 0622 - 062E, and 0645 exception trace.
The following point IDs are provided for the DFHTONR module:
- AP 0648 - 0649, for which the trace level is 1.
- AP 064A - 064C, exception trace.

The following point IDs are provided for the DFHAPRDR module:
- AP 0601 - 0602, for which the trace level is 1.
- AP 0603 - 061E, exception trace.

The following point IDs are provided for the DFHZGTA module:
- AP FA80 - FA81, for which the trace level is 1.
- AP FA82 - FA9A, exception trace.

The following point ID is provided for message set production:
- AP FCDD, exception trace.

The following point ID is provided for DFHBSTZA:
- AP FCDE, exception trace.

See the CICS Trace Entries for further information.

Messages

Builder modules issue messages in the DFHZC59xx, DFHZC62xx, and DFHZC63xx series.

Message sets

If a builder finds an error, it adds a message to a message set. This set is then printed by the caller; for example:

DFHTCRP    Cold start (local system entry
and error console only)
DFHAMTP    CEDA, EXEC CICS CREATE
DFHEIQSC   EXEC CICS DISCARD CONNECTION
DFHEIQST   EXEC CICS DISCARD TERMINAL
DFHZATA    Autoinstall
dFHZATD    Autoinstall delete
dFHZATS     Install and delete transaction routed terminals

How messages show up in a trace

If a message is issued from a builder module (that is, those with a prefix of DFHZC59xx, DFHZC62xx, or DFHZC63xx), it appears in the trace as a TBSM trace entry with the following point ID:
- AP FCDD, exception trace.

This trace entry is produced when a message is added to the message set and indicates there was a problem in building or deleting a terminal or connection.

For more information about the trace points, see the CICS Trace Entries. For more information about using traces in problem determination, see the CICS Problem Determination Guide.
Chapter 12. Built-in functions

CICS provides the application programmer with two commonly used functions: field edit and phonetic conversion.

These are functions that generally used to be coded as separate subroutines by the programmer. They are referred to as built-in functions.

The field edit function is provided by the BIF DEEDIT command of the CICS application programming interface.

The phonetic conversion function is provided as a subroutine that can be called by CICS application programs, and also by any offline programs.

Design overview

The built-in functions component includes field edit and phonetic conversion, both of which are available to a CICS application program. Also, the phonetic conversion subroutine can be used offline.

Field edit (DEEDIT)

The field edit function allows the application program to pass a field containing EBCDIC digits (0 through 9) intermixed with other values, and receive a result with all non-numeric characters removed.

For further details of this function, see the CICS Application Programming Reference.

Phonetic conversion

This facility allows the user to organize a file according to name (or similar alphabetic key), and access the file using search arguments that may be misspelled.

The phonetic conversion subroutine (DFHPHN) converts a name into a partial key, which can then be used to access a database file. The generated key is based upon the sound of the name. This means that names sounding similar, but spelled differently, generally produce identical keys. For example, the names SMITH, SMYTH, and SMYTHE all produce a phonetic key of 5530. Likewise, the names ANDERSON, ANDRESEN, and ANDRESENN produce a phonetic key of A536. The encoding routine ignores embedded blanks in a name, so you can write names prefixed by ‘Mc’ with or without a blank between the prefix and the rest of the name, for example, ‘McEWEN’ or ‘Mc EWEN’.

For details of how to code a CALL statement for the DFHPHN subroutine according to the language of the application program, see the CICS Application Programming Guide.

Modules

<table>
<thead>
<tr>
<th>Module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHEBF</td>
<td>EXEC interface processor for BIF DEEDIT command</td>
</tr>
<tr>
<td>DFHPHN</td>
<td>Phonetic conversion subroutine</td>
</tr>
</tbody>
</table>
Built-in functions

Exits

No global user exit points are provided for these functions.

Trace

No tracing is performed for the phonetic conversion subroutine.

The following point ID is provided for DFHEBF:
• AP 00FB, for which the trace level is BF 1.

For more information about the trace points, see the CICS Trace Entries. For more information about using traces in problem determination, see the CICS Problem Determination Guide.
Chapter 13. Business Application Manager domain (BAM)

The business application manager domain (also sometimes known simply as “business application manager”) is responsible for managing CICS business transaction services’ (BTS) processes, process types and activities. It deals with the hardening of the associated data to BTS repository files. Along with scheduler services domain and event manager domain it forms the CICS BTS function.

Business application manager domain’s specific gate

Table 7 summarizes the business application manager domain’s specific gate. It shows the level-1 trace point IDs of the modules providing the functions for the gate, the functions provided by the gate, and whether or not the functions are available through the exit programming interface (XPI).

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>XPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>BATT</td>
<td>BA 0160</td>
<td>ADD_REPLACE_PROCSSTYPE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>BA 0161</td>
<td>INQUIRE_PROCSSTYPE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>START_BROWSE_PROCSSTYPE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GET_NEXT_PROCSSTYPE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>END_BROWSE_PROCSSTYPE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SET_PROCSSTYPE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DISCARD_PROCSSTYPE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COMMIT_PROCSSTYPE_TABLE</td>
<td>NO</td>
</tr>
<tr>
<td>BAXM</td>
<td>BA 0170</td>
<td>INIT_ACTIVITY_REQUEST</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>BA 0171</td>
<td>BIND_ACTIVITY_REQUEST</td>
<td>NO</td>
</tr>
<tr>
<td>BAPR</td>
<td>BA 0110</td>
<td>ADD_PROCESS</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>BA 0111</td>
<td>RUN_PROCESS</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LINK_PROCESS</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ACQUIRE_PROCESS</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CANCEL_PROCESS</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SUSPEND_PROCESS</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RESUME_PROCESS</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CHECK_PROCESS</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RESET_PROCESS</td>
<td>NO</td>
</tr>
<tr>
<td>BAAC</td>
<td>BA 0120</td>
<td>ADD_ACTIVITY</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>BA 0121</td>
<td>RUN_ACTIVITY</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CHECK_ACTIVITY</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RETURN_END_ACTIVITY</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DELETE_ACTIVITY</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SUSPEND_ACTIVITY</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RESUME_ACTIVITY</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CANCEL_ACTIVITY</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LINK_ACTIVITY</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ACQUIRE_ACTIVITY</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RESET_ACTIVITY</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADD_TIMER_REQUEST</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADD_REALTACH_ACQUIRED</td>
<td>NO</td>
</tr>
</tbody>
</table>
### Business application manager domain (BAM)

**Table 7. Business application manager domain’s specific gate (continued)**

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>XPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>BABR</td>
<td>BA 0150</td>
<td>STARTBR_ACTIVITY</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>BA 0151</td>
<td>GETNEXT_ACTIVITY</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ENDBR_ACTIVITY</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INQUIRE_ACTIVITY</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>STARTBR_CONTAINER</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GETNEXT_CONTAINER</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ENDBR_CONTAINER</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INQUIRE_CONTAINER</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>STARTBR_PROCESS</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GETNEXT_PROCESS</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ENDBR_PROCESS</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INQUIRE_PROCESS</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INQUIRE_ACTIVATION</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COMMIT_BROWSE</td>
<td>NO</td>
</tr>
<tr>
<td>BACR</td>
<td>BA 0130</td>
<td>DELETE_CONTAINER</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>BA 0131</td>
<td>GET_CONTAINER_INTO</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GET_CONTAINER_SET</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PUT_CONTAINER</td>
<td>NO</td>
</tr>
<tr>
<td>BAGD</td>
<td>BA 0401</td>
<td>INQUIRE_DATA_LENGTH</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>BA 0402</td>
<td>GET_DATA</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DESTROY_TOKEN</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDRESS_DATA</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RELEASE_DATA</td>
<td>NO</td>
</tr>
</tbody>
</table>

**BATT gate, ADD_REPLACE_PROCESSTYPE function**

The ADD_REPLACE_PROCESSTYPE function of the BATT gate is used to add a new process type definition or replace an existing process type definition. Process types are defined using RDO.

**Input parameters**

**PROCESSTYPE_NAME**

- is an 8-character name.

**FILE_NAME**

- is an 8-character name of the repository file to be associated with this process type. The file is defined using RDO.

**AUDITLOG_NAME**

- is an 8-character name of the audit log to be associated with this process type. The log is defined using RDO.

**AUDITLEVEL**

determines the level of auditing to be undertaken for this process type. It can take the values:

- OFF
- PROCESS
- ACTIVITY
- FULL

**USERRECORDS**

- indicates whether user audit records are to be written to the log. It can take the values:
  - YES
  - NO

**CATALOG_PTDEF**

- indicates whether the definition should be written to the global catalog. It can take the values:
  - YES
  - NO
STATUS
indicates whether the process type definition should be installed in a disabled or enabled state. It can take the values:
DISABLED|ENABLED

Output parameters
RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NOT_DISABLED, INSUFFICIENT_STORAGE</td>
</tr>
</tbody>
</table>

BATT gate, INQUIRE_PROCESSTYPE function
The INQUIRE_PROCESSTYPE function of the BATT gate is used to return information on the named process type.

Input parameters
PROCESSTYPE_NAME
is the 8-character name of the process type to be inquired upon.

Output parameters
FILE_NAME
is the 8-character name of the repository file associated with this process type.

AUDITLOG_NAME
is an 8-character name of the audit log associated with this process type.

AUDITLEVEL
identifies the level of auditing for this process type. It can take the values:
OFF|PROCESS|ACTIVITY|FULL

USERRECORDS
indicates whether user audit records are to being written to the log. It can take the values:
YES|NO

STATUS
indicates the status of the process type. It can take the values:
DISABLED|ENABLED

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|PURGED|INVALID|DISASTER|KERNERROR

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>ENTRY_NOT_FOUND</td>
</tr>
</tbody>
</table>
BATT gate, START_BROWSE_PROCESSTYPE function

The START_BROWSE_PROCESSTYPE function of the BATT gate is used to initiate a browse of the process types known to this region.

Input parameters
None

Output parameters

BROWSE_TOKEN
is the token used to identify this browse.

RESPONSE
is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

BATT gate, GET_NEXT_PROCESSTYPE function

The GET_NEXT_PROCESSTYPE function of the BATT gate is used to return the name of the next process type in the browse, identified by the browse token.

Input parameters

BROWSE_TOKEN
is the token returned to the caller on the START_BROWSE_PROCESSTYPE call.

Output parameters

PROCESSTYPE_NAME
the 8-character process type name.

RESPONSE
is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NO_MORE_DATA_AVAILABLE</td>
</tr>
</tbody>
</table>

BATT gate, END_BROWSE_PROCESSTYPE function

The END_BROWSE_PROCESSTYPE function of the BATT gate is used to end the browse identified by the browse token.

Input parameters

BROWSE_TOKEN
is the token returned to the caller on the START_BROWSE_PROCESSTYPE call.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED
BATT gate, SET_PROCESSTYPE function

The SET_PROCESSTYPE function of the BATT gate is used to alter the named process type definition.

Input parameters

- **PROCESSTYPE_NAME** is the 8-character process type name.
- **FILE_NAME** is an 8-character name of the repository file to be associated with this process type.
- **AUDITLEVEL** determines the level of auditing to be undertaken for this process type. It can take the values: OFF|PROCESS|ACTIVITY|FULL
- **USERRECORDS** indicates whether user audit records are to be written to the log. It can take the values: YES|NO
- **STATUS** indicates whether the status of the process type. It can take the values: DISABLED|ENABLED

Output parameters

- **RESPONSE** is the domain's response to the call. It can have any of these values: OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>ENTRY_NOT_FOUND, NOT_DISABLED</td>
</tr>
</tbody>
</table>

BATT gate, DISCARD_PROCESSTYPE function

The DISCARD_PROCESSTYPE function of the BATT gate is used to discard the named process type definition.

Input parameters

- **PROCESSTYPE_NAME** is the 8-character process type name.

Output parameters

- **RESPONSE** is the domain’s response to the call. It can have any of these values: OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>ENTRY_NOT_FOUND, NOT_DISABLED</td>
</tr>
</tbody>
</table>
BATT gate, COMMIT_PROCESSSTYPE_TABLE function

The COMMIT_PROCESSSTYPE_TABLE function of the BATT gate is used to commit the process type definitions to the global catalog.

Input parameters

TOKEN
is the token identifying the table of process type definitions.

Output parameters

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

BAXM gate, INIT_ACTIVITY_REQUEST function

The INIT_ACTIVITY_REQUEST function of the BAXM gate is used when the transaction requires a 3270 bridge facility, in which case the named bridge exit program is invoked.

Input parameters

REQUEST_BLOCK
a block used to hold the request data.

BRIDGE_EXIT
the 8-character name of the bridge exit program.

Output parameters

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

BAXM gate, BIND_ACTIVITY_REQUEST function

The BIND_ACTIVITY_REQUEST function of the BAXM gate is used to make the current UOW an activation of the activity specified in the activity request. This activation could be used to mark the activity complete abended because the previous activation failed, hence the abend information.

Input parameters

ABEND_CODE
the 4-character abend code.

ABEND_PROG
the 8-character abend program name.

ABEND_MSG
the 6-character abend message number.

REQUEST_BLOCK
a block used to hold the activity request data.

Output parameters

PROGRAM
is the 8-character program name.
**RUN_PROGRAM**

is used to indicate if a program is to be invoked on the program manager INITIAL_LINK. It can take the values:

YES | NO

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | INVALID | DISASTER | KERNERROR | PURGED

**[REASON]**

is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>ACTIVITY_NOT_FOUND, TIMEOUT, READ_FAILURE</td>
</tr>
</tbody>
</table>

**BAPR gate, ADD_PROCESS function**

The ADD_PROCESS function of the BAPR gate is used to define a new process in response to an EXEC CICS DEFINE PROCESS call.

**Input parameters**

**PROCESS_NAME**

the 36-character process name.

**PROCESSTYPE**

the 8-character process type.

**TRANID**

the 4-character transaction id.

**PROGRAM**

the 8-character program name associated with the root activity.

**USERID**

the 8-character userid.

**Output parameters**

**PROCESS_TOKEN**

a token representing this process internally.

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | INVALID | DISASTER | KERNERROR | PURGED

**[REASON]**

is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>DUPLICATE_PROCESS_NAME, FILE_NOT_AUTH, PROCESS_ALREADY_ACQUIRED, PROCESSTYPE_NOT_ENABLED, PROCESSTYPE_NOT_FOUND, WRITE_FAILED</td>
</tr>
</tbody>
</table>

**BAPR gate, RUN_PROCESS function**

The RUN_PROCESS function of the BAPR gate is used to execute the acquired process (invoke the root activity), either asynchronously or synchronously i.e. with a context switch.
Business application manager domain (BAM)

**Input parameters**

**MODE**
- can take the values: 
  - SYNC|ASYNC

**INPUT_EVENT**
- the 16-character name of the input event.

**FACILITY_TOKEN**
- the 8-character facility token.

**Output parameters**

**RESPONSE**
- is the domain’s response to the call. It can have any of these values: 
  - OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

**[REASON]**
- is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>PROCESS_NOT_FOUND, PROCESSTYPE_NOT_FOUND, PROCESS_SUSPENDED, OTHER_PROCESS_CURRENT, INVALID_EVENT, INVALID_MODE, AUTOINSTALL_FAILED, AUTOINSTALL_INVALID_DATA, AUTOINSTALL_MODEL_NOT_DEF, AUTOINSTALL_URM_FAILED, PROGRAM_NOT_AUTHOURISED, PROGRAM_NOT_DEFINED, PROGRAM_NOT_ENABLED, PROGRAM_NOT_LOADABLE, REMOTE_PROGRAM, SECOND_JVM_PROGRAM, RUN_SYNC_ABENDED, RECORD_BUSY, REMOTE_TRAN, TRAN_NOT_AUTH</td>
</tr>
</tbody>
</table>

**BAPR gate, LINK_PROCESS function**

The LINK_PROCESS function of the BAPR gate is used to invoke the acquired process synchronously, without a context switch.

**Input parameters**

**INPUT_EVENT**
- the 16-character name of the input event.

**Output parameters**

**RESPONSE**
- is the domain’s response to the call. It can have any of these values: 
  - OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

**[REASON]**
- is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>PROCESS_NOT_FOUND, PROCESSTYPE_NOT_FOUND, PROCESS_SUSPENDED, OTHER_PROCESS_CURRENT, INVALID_EVENT, INVALID_MODE, AUTOINSTALL_FAILED, AUTOINSTALL_INVALID_DATA, AUTOINSTALL_MODEL_NOT_DEF, AUTOINSTALL_URM_FAILED, PROGRAM_NOT_AUTHOURISED, PROGRAM_NOT_DEFINED, PROGRAM_NOT_ENABLED, PROGRAM_NOT_LOADABLE, REMOTE_PROGRAM, SECOND_JVM_PROGRAM, NO_EVENTS_PROCESSED, PENDING_ACTIVITY_EVENTS</td>
</tr>
</tbody>
</table>
BAPR gate, ACQUIRE_PROCESS function
The ACQUIRE_PROCESS function of the BAPR gate is used to acquire the named process.

Input parameters
PROCESS_NAME
the 36-character process name.

PROCESSTYPE
the 8-character process type.

Output parameters
RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>PROCESS_NOT_FOUND, PROCESSTYPE_NOT_FOUND, FILE_NOT_AUTH, OTHER_PROCESS_CURRENT, RECORD_BUSY</td>
</tr>
</tbody>
</table>

BAPR gate, CANCEL_PROCESS function
The CANCEL_PROCESS function of the BAPR gate is used to synchronously cancel the acquired process.

Input parameters
None

Output parameters
RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>PROCESS_NOT_FOUND, PROCESSTYPE_NOT_FOUND, FILE_NOT_AUTH, RECORD_BUSY</td>
</tr>
</tbody>
</table>

BAPR gate, SUSPEND_PROCESS function
The SUSPEND_PROCESS function of the BAPR gate is used to suspend the acquired process.

Input parameters
None

Output parameters
RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED
Business application manager domain (BAM)

[REASON]

is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>PROCESS_NOT_FOUND, RECORD_BUSY</td>
</tr>
</tbody>
</table>

BAPR gate, RESUME_PROCESS function

The RESUME_PROCESS function of the BAPR gate is used to resume a previously suspended process.

Input parameters

None

Output parameters

RESPONSE

is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON]

is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>PROCESS_NOT_FOUND, RECORD_BUSY</td>
</tr>
</tbody>
</table>

BAPR gate, CHECK_PROCESS function

The CHECK_PROCESS function of the BAPR gate is used to establish how the acquired process completed.

Input parameters

None

Output parameters

COMPLETION_STATUS

is the completion status of the process. It can have any of these values:

NORMAL|ABENDED|FORCEDCOMPLETE|INCOMPLETE

ABEND_CODE

the 4-character abend code.

ABEND_PROGRAM

the 8-character name of the program which abended.

SUSPENDED

indicates whether the process is suspended. It can take the value:

YES|NO

ACTMODE

the active mode of the process. It can take the value:

INITIAL|ACTIVE|DORMANT|CANCELLING|COMPLETE

RESPONSE

is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON]

is returned when RESPONSE is EXCEPTION. Possible values are:
**BAPR gate, REST_PROCESS function**

The `REST_PROCESS` function of the BAPR gate is used to reset the state of the acquired root activity to initial, so it may be run again.

**Input parameters**

None

**Output parameters**

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

- OK
- EXCEPTION
- INVALID
- DISASTER
- KERNERROR
- PURGED

**[REASON]**

is returned when **RESPONSE** is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>PROCESS_NOT_FOUND, RECORD_BUSY</td>
</tr>
</tbody>
</table>

**BAAC gate, ADD_ACTIVITY function**

The `ADD_ACTIVITY` function of the BAAC gate is used to define a new activity in response to an EXEC CICS DEFINE ACTIVITY call.

**Input parameters**

**ACTIVITY_NAME**

the 16-character activity name.

**COMPLETION_EVENT**

the 16-character completion event.

**TRANID**

the 4-character transaction id.

**PROGRAM**

the 8-character program name associated with the root activity.

**USERID**

the 8-character userid.

**ACTIVITYID**

the buffer containing the activity identifier.

**Output parameters**

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

- OK
- EXCEPTION
- INVALID
- DISASTER
- KERNERROR
- PURGED

**[REASON]**

is returned when **RESPONSE** is EXCEPTION. Possible values are:
**BAAC gate, RUN_ACTIVITY function**

The RUN_ACTIVITY function of the BAAC gate is used to execute the named child activity or the acquired activity either asynchronously or synchronously i.e. with a context switch.

**Input parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTIVITY_NAME</td>
<td>the 16-character activity name.</td>
</tr>
<tr>
<td>MODE</td>
<td>can take the values:</td>
</tr>
<tr>
<td></td>
<td>SYNC</td>
</tr>
<tr>
<td>INPUT_EVENT</td>
<td>the 16-character name of the input event.</td>
</tr>
<tr>
<td>FACILITY_TOKEN</td>
<td>the 8-character facility token.</td>
</tr>
</tbody>
</table>

**Output parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESPONSE</td>
<td>is the domain’s response to the call. It can have any of these values:</td>
</tr>
<tr>
<td></td>
<td>OK</td>
</tr>
<tr>
<td>[REASON]</td>
<td>is returned when RESPONSE is EXCEPTION. Possible values are:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>DUPLICATE_ACTIVITY_NAME, NO_CURRENT_ACTIVITY, UNKNOWN_TRANSACTION_ID, INVALID_NAME</td>
</tr>
</tbody>
</table>

**BAAC gate, LINK_ACTIVITY function**

The LINK_PROCESS function of the BAAC gate is used to invoke the named child activity or acquired activity synchronously, without a context switch.

**Input parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTIVITY_NAME</td>
<td>the 16-character name of the activity.</td>
</tr>
<tr>
<td>INPUT_EVENT</td>
<td>the 16-character name of the input event.</td>
</tr>
</tbody>
</table>

**Output parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESPONSE</td>
<td>is the domain’s response to the call. It can have any of these values:</td>
</tr>
<tr>
<td></td>
<td>OK</td>
</tr>
<tr>
<td>[REASON]</td>
<td>is returned when RESPONSE is EXCEPTION. Possible values are:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>ACTIVITY_NOT_FOUND, INVALID_EVENT, INVALID_MODE, NO_CURRENT_ACTIVITY, NO_COMPLETION_EVENT, REMOTE_PROGRAM, ACTIVITY_SUSPENDED, RUN_SYNC_ABENDED, READ_FAILURE, RECORD_BUSY, REMOTE_TRAN, TRAN_NOT_AUTH</td>
</tr>
</tbody>
</table>
RESPONSE | Possible REASON values
--- | ---
EXCEPTION | ACTIVITY_NOT_FOUND, NO_CURRENT_ACTIVITY, NO_COMPLETION_EVENT, INVALID_EVENT, INVALID_MODE, AUTOINSTALL_FAILED, AUTOINSTALL_INVALID_DATA, AUTOINSTALL_MODEL_NOT_DEF, AUTOINSTALL_URM_FAILED, PROGRAM_NOT_AUTHOURISED, PROGRAM_NOT_DEFINED, PROGRAM_NOT_ENABLED, PROGRAM_NOT_LOADABLE, REMOTE_PROGRAM, SECOND_JVM_PROGRAM, NO_EVENTS_PROCESSED, PENDING_ACTIVITY_EVENTS

**BAAC gate, CANCEL_ACTIVITY function**
The CANCEL_ACTIVITY function of the BAAC gate is used to synchronously cancel the named child activity or the acquired activity.

**Input parameters**
ACTIVITY_NAME
the 16-character activity name.

**Output parameters**
RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>ACTIVITY_NOT_FOUND, NO_CURRENT_ACTIVITY, INVALID_MODE, INVALID_ACTIVITYID, FILE_NOT_AUTH, RECORD_BUSY</td>
</tr>
</tbody>
</table>

**BAAC gate, SUSPEND_ACTIVITY function**
The SUSPEND_ACTIVITY function of the BAAC gate is used to suspend the named child activity or the acquired activity.

**Input parameters**
ACTIVITY_NAME
the 16-character activity name.

**Output parameters**
RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>ACTIVITY_NOT_FOUND, NO_ACQUIRED_ACTIVITY, INVALID_MODE, READ_FAILURE, RECORD_BUSY</td>
</tr>
</tbody>
</table>

**BAAC gate, RESUME_ACTIVITY function**
The RESUME_ACTIVITY function of the BAAC gate is used to resume a previously suspended activity.
Business application manager domain (BAM)

Input parameters
ACTIVITY_NAME
    the 16-character activity name.

Output parameters
RESPONSE
    is the domain’s response to the call. It can have any of these values:
        OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON]
    is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>ACTIVITY_NOT_FOUND, NO_ACQUIRED_ACTIVITY, INVALID_MODE, READ_FAILURE, RECORD_BUSY</td>
</tr>
</tbody>
</table>

BAAC gate, CHECK_ACTIVITY function

The CHECK_ACTIVITY function of the BAAC gate is used to establish how the named child activity or acquired activity completed.

Input parameters
ACTIVITY_NAME
    the 16-character activity name.

Output parameters
COMPLETION_STATUS
    is the completion status of the activity. It can have any of these values:
        NORMAL|ABENDED|FORCEDCOMPLETE|INCOMPLETE

ABEND_CODE
    the 4-character abend code.

ABEND_PROGRAM
    the 8-character name of the program which abended.

SUSPENDED
    indicates whether the process is suspended. It can take the value:
        YES|NO

ACTMODE
    the active mode of the process. It can take the value:
        INITIAL|ACTIVE|DORMANT|CANCELLING|COMPLETE

RESPONSE
    is the domain’s response to the call. It can have any of these values:
        OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON]
    is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>ACTIVITY_NOT_FOUND, NO_CURRENT_ACTIVITY, READ_FAILURE, RECORD_BUSY</td>
</tr>
</tbody>
</table>
BAAC gate, RESET_ACTIVITY function
The RESET_ACTIVITY function of the BAAC gate is used to reset the state of the named child activity to initial, so it may be run again.

Input parameters
ACTIVITY_NAME
the 16-character activity name.

Output parameters
RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>ACTIVITY_NOT_FOUND, NO_CURRENT_ACTIVITY, FILE_NOT_AUTH, INVALID_MODE, READ_FAILURE, RECORD_BUSY</td>
</tr>
</tbody>
</table>

BAAC gate, RETURN_END_ACTIVITY function
The RETURN_END_ACTIVITY function of the BAAC gate is used to indicate the completion of the current activity and so raise the completion event.

Input parameters
None

Output parameters
RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NO_CURRENT_ACTIVITY</td>
</tr>
</tbody>
</table>

BAAC gate, DELETE_ACTIVITY function
The DELETE_ACTIVITY function of the BAAC gate is used to delete the named child activity from the repository.

Input parameters
ACTIVITY_NAME
the 16-character activity name.

Output parameters
ACTMODE
the active mode of the process. It can take the value:
INITIAL|ACTIVE|DORMANT|CANCELLING|COMPLETE

RESPONSE
is the domain’s response to the call. It can have any of these values:
Business application manager domain (BAM)

\textbf{OK}\mid\textbf{EXCEPTION}\mid\textbf{INVALID}\mid\textbf{DISASTER}\mid\textbf{KERNERROR}\mid\textbf{PURGED}

\[\text{[REASON]}\]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>ACTIVITY_NOT_FOUND, NO_CURRENT_ACTIVITY, INVALID_MODE, READ_FAILURE, RECORD_BUSY</td>
</tr>
</tbody>
</table>

\textbf{BAAC gate, ACQUIRE\_ACTIVITY function}

The ACQUIRE\_ACTIVITY function of the BAAC gate is used to acquire the specified activity.

\textbf{Input parameters}

\textbf{ACTIVITYID}

the buffer for the activity identifier.

\textbf{Output parameters}

\textbf{RESPONSE}

is the domain’s response to the call. It can have any of these values:

\textbf{OK}\mid\textbf{EXCEPTION}\mid\textbf{INVALID}\mid\textbf{DISASTER}\mid\textbf{KERNERROR}\mid\textbf{PURGED}

\[\text{[REASON]}\]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>ACTIVITY_NOT_FOUND, ACTIVITY_ALREADY_ACQUIRED, READ_FAILURE, RECORD_BUSY</td>
</tr>
</tbody>
</table>

\textbf{BAAC gate, ADD\_TIMER\_REQUEST function}

The ADD\_TIMER\_REQUEST function of the BAAC gate is used to add a delayed request to BAM domain in response to an EXEC CICS DEFINE TIMER call.

\textbf{Input parameters}

\textbf{REQUEST\_TOKEN}

the token representing the request.

\textbf{TIMER\_EVENT}

the timer event name.

\textbf{EVENT\_VERSION}

the version of the event.

\textbf{DATETIME}

the time at which the timer expires.

\textbf{Output parameters}

\textbf{RESPONSE}

is the domain’s response to the call. It can have any of these values:

\textbf{OK}\mid\textbf{EXCEPTION}\mid\textbf{INVALID}\mid\textbf{DISASTER}\mid\textbf{KERNERROR}\mid\textbf{PURGED}

\[\text{[REASON]}\]
is returned when RESPONSE is EXCEPTION. Possible values are:
### BAAC gate, ADD_REATTACH_ACQUIRED function

The ADD_REATTACH_ACQUIRED function of the BAAC gate is used to reattach an activity.

#### Input parameters
None

#### Output parameters
- **RESPONSE**
  is the domain's response to the call. It can have any of these values: OK, EXCEPTION, INVALID, DISASTER, KERNERROR, PURGED

  - **[REASON]**
    is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NO_CURRENT_ACTIVITY</td>
</tr>
</tbody>
</table>

### BABR gate, STARTBR_ACTIVITY function

The STARTBR_ACTIVITY function of the BABR gate is used to initiate a browse of activities from the specified activity identifier or from the root activity of the specified process.

#### Input parameters
- **ACTIVITYID**
  is a buffer containing the activity identifier.
- **PROCESS_NAME**
  is a buffer containing the process name.
- **PROCESS_TYPE**
  is the 8-character process type.

#### Output parameters
- **BROWSE_TOKEN**
  is the token identifying the browse.

- **RESPONSE**
  is the domain's response to the call. It can have any of these values: OK, EXCEPTION, INVALID, DISASTER, KERNERROR, PURGED

  - **[REASON]**
    is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>ACTIVITY_NOT_FOUND, FILE_NOT_AUTH, NO_CURRENT_ACTIVITY, PROCESS_NOT_FOUND, PROCESSTYPE_NOT_FOUND, RECORD_BUSY</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_ACTIVITYID_LEN, INVALID_PROCESSNAME_LEN</td>
</tr>
</tbody>
</table>
Business application manager domain (BAM)

**BABR gate, GETNEXT_ACTIVITY function**

The GETNEXT_ACTIVITY function of the BABR gate is used to return the next activity in the specified browse.

**Input parameters**

- **RETURNED_ACTIVITYID**
  - is a buffer containing the activity identifier.

- **BROWSE_TOKEN**
  - is the browse token.

**Output parameters**

- **ACTIVITY_NAME**
  - is the 16-character activity name.

- **LEVEL**
  - is the level into the activity tree.

- **RESPONSE**
  - is the domain’s response to the call. It can have any of these values:
    - OK
    - EXCEPTION
    - INVALID
    - DISASTER
    - KERNERROR
    - PURGED

**[REASON]**

- is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>BROWSE_END, INVALID_BROWSE_TOKEN, INVALID_BROWSE_TYPE, RECORD_BUSY</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_BUFFER_LENGTH</td>
</tr>
</tbody>
</table>

**BABR gate, ENDBR_ACTIVITY function**

The ENDBR_ACTIVITY function of the BABR gate is used to end the specified activity browse.

**Input parameters**

- **BROWSE_TOKEN**
  - is the browse token.

**Output parameters**

- **RESPONSE**
  - is the domain’s response to the call. It can have any of these values:
    - OK
    - EXCEPTION
    - INVALID
    - DISASTER
    - KERNERROR
    - PURGED

**[REASON]**

- is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>INVALID_BROWSE_TOKEN, INVALID_BROWSE_TYPE</td>
</tr>
</tbody>
</table>

**BABR gate, INQUIRE_ACTIVITY function**

The INQUIRE_ACTIVITY function of the BABR gate is used to obtain information about the specified activity.
Input parameters

ACTIVITYID
is a buffer containing the identifier of the activity which is to be inquired upon.

RETURNED_ACTIVITYID
is a buffer containing the returned activity identifier.

RETURNED_PROCESS_NAME
is a buffer containing the returned process name.

Output parameters

ABEND_CODE
is the 4-character abend code.

ABEND_PROGRAM
is the 8-character name of the program which abended.

ACTIVITY_NAME
is the 16-character activity name.

COMPLETION_STATUS
is the completion status. It can take the values:
ABENDED | FORCED | INCOMPLETE | NORMAL

EVENT_NAME
is the 16-character event name.

MODE
is the mode of the activity. It can take the values:
INITIAL | ACTIVE | DORMANT | CANCELLING | COMPLETE

PROCESS_TYPE
is the 8-character process type.

PROGRAM
is the 8-character name of the activity program.

TRANSID
is the 4-character transaction identifier.

INIT_TRANSID
is the 4-character transaction identifier of the transaction under which the activity was initiated.

USERID
is the 8-character userid.

SUSPENDED
indicates whether the activity is currently suspended. It can take the values:
YES | NO

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK | EXCEPTION | INVALID | DISASTER | KERNERROR | PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:
RESPONSE Possible REASON values

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>ACTIVITY_NOT_FOUND, FILE_NOT_AUTH, NO_CURRENT_ACTIVITY, RECORD_BUSY</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_ACTIVITYID_LEN, INVALID_BUFFER_LEN</td>
</tr>
</tbody>
</table>

BABR gate, STARTBR_CONTAINER function

The STARTBR_CONTAINER function of the BABR gate is used to initiate a browse of containers associated with a specified activity or process.

**Input parameters**

- **ACTIVITYID**
  is a buffer containing the activity identifier.

- **PROCESS_NAME**
  is a buffer containing the process name.

- **PROCESS_TYPE**
  is the 8-character process type.

**Output parameters**

- **BROWSE_TOKEN**
  is the token identifying the browse.

- **RESPONSE**
  is the domain’s response to the call. It can have any of these values:
    - OK
    - EXCEPTION
    - INVALID
    - DISASTER
    - KERNERROR
    - PURGED
  [REASON] is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>ACTIVITY_NOT_FOUND, FILE_NOT_AUTH, NO_CURRENT_ACTIVITY, PROCESS_NOT_FOUND, PROCESSTYPE_NOT_FOUND, RECORD_BUSY</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_ACTIVITYID_LEN, INVALID_PROCESSNAME_LEN</td>
</tr>
</tbody>
</table>

BABR gate, GETNEXT_CONTAINER function

The GETNEXT_CONTAINER function of the BABR gate is used to return the next container in the specified browse.

**Input parameters**

- **BROWSE_TOKEN**
  is the browse token.

**Output parameters**

- **CONTAINER_NAME**
  is the 16-character container name.

- **RESPONSE**
  is the domain’s response to the call. It can have any of these values:
    - OK
    - EXCEPTION
    - INVALID
    - DISASTER
    - KERNERROR
    - PURGED
  [REASON] is returned when RESPONSE is EXCEPTION. Possible values are:
### BABR gate, ENDBR_CONTAINER function

The ENDBR_CONTAINER function of the BABR gate is used to end the specified container browse.

#### Input parameters

**BROWSE_TOKEN**

is the browse token.

#### Output parameters

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

- OK
- EXCEPTION
- INVALID
- DISASTER
- KERNERROR
- PURGED

**[REASON]**

is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>BROWSE_END, INVALID_BROWSE_TOKEN, INVALID_BROWSE_TYPE, RECORD_BUSY</td>
</tr>
</tbody>
</table>

### BABR gate, INQUIRE_CONTAINER function

The INQUIRE_CONTAINER function of the BABR gate is used to obtain information about the specified container.

#### Input parameters

**CONTAINER_NAME**

the 16-character container name.

**ACTIVITYID**

is a buffer containing the activity identifier.

**PROCESS_NAME**

is a buffer containing the process name.

**PROCESS_TYPE**

is the 8-character process type.

#### Output parameters

**DATA_LENGTH**

is the length of the container data.

**DATA_ADDRESS**

is the address of the container data.

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

- OK
- EXCEPTION
- INVALID
- DISASTER
- KERNERROR
- PURGED

**[REASON]**

is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:
BUSINESS APPLICATION MANAGER DOMAIN (BAM)

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>ACTIVITY_NOT_FOUND, CONTAINER_NOT_FOUND, PROCESS_NOT_FOUND, PROCESSTYPE_NOT_FOUND, FILE_NOT_AUTH, NO_CURRENT_ACTIVITY, RECORD_BUSY</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_ACTIVITYID_LEN, INVALID_PROCESSNAME_LEN</td>
</tr>
</tbody>
</table>

BABR gate, STARTBR_PROCESS function

The STARTBR_PROCESS function of the BABR gate is used to initiate a browse of the processes of a certain type.

**Input parameters**

**PROCESS_TYPE**

is the 8-character process type.

**Output parameters**

**BROWSE_TOKEN**

is the token identifying the browse.

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | INVALID | DISASTER | KERNERROR | PURGED

[REASON]

is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>FILE_NOT_AUTH, FILE_UNAVAILABLE, NO_CURRENT_ACTIVITY, PROCESSTYPE_NOT_FOUND, RECORD_BUSY</td>
</tr>
</tbody>
</table>

BABR gate, GETNEXT_PROCESS function

The GETNEXT_PROCESS function of the BABR gate is used to return the next process in the specified browse.

**Input parameters**

**BROWSE_TOKEN**

is the browse token.

**RETURNED_ACTIVITYID**

is a buffer containing the activity identifier.

**RETURNED_PROCESS_NAME**

is a buffer containing the process name.

**Output parameters**

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | INVALID | DISASTER | KERNERROR | PURGED

[REASON]

is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>BROWSE_END, INVALID_BROWSE_TOKEN, INVALID_BROWSE_TYPE, RECORD_BUSY</td>
</tr>
</tbody>
</table>
BABR gate, ENDBR_PROCESS function

The ENDBR_PROCESS function of the BABR gate is used to end the specified process browse.

**Input parameters**

- BROWSE_TOKEN
  - is the browse token.

**Output parameters**

- RESPONSE
  - is the domain’s response to the call. It can have any of these values:
    - OK
    - EXCEPTION
    - INVALID
    - DISASTER
    - KERNERROR
    - PURGED

  [REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>INVALID_BROWSE_TOKEN, INVALID_BROWSE_TYPE</td>
</tr>
</tbody>
</table>

BABR gate, INQUIRE_PROCESS function

The INQUIRE_PROCESS function of the BABR gate is used to obtain information about the specified process.

**Input parameters**

- RETURNED_ACTIVITYID
  - is a buffer containing the activity identifier.
- PROCESS_NAME
  - is a buffer containing the process name.
- PROCESS_TYPE
  - is the 8-character process type.

**Output parameters**

- RESPONSE
  - is the domain’s response to the call. It can have any of these values:
    - OK
    - EXCEPTION
    - INVALID
    - DISASTER
    - KERNERROR
    - PURGED

  [REASON] is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>PROCESS_NOT_FOUND, PROCESSTYPE_NOT_FOUND, FILE_NOT_AUTH, RECORD_BUSY</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_BUFFER_LENGTH</td>
</tr>
</tbody>
</table>
BAMR gate, INQUIRE_ACTIVATION function

The INQUIRE_ACTIVATION function of the BAMR gate is used to obtain information about the activation associated with a running transaction, if there is one.

Input parameters

TRANSACTION_TOKEN
is a token representing an instance of a transaction.

RETURNED_ACTIVITYID
is a buffer containing the activity identifier.

RETURNED_PROCESS_NAME
is a buffer containing the process name.

Output parameters

ACTIVITY_NAME
is the 16-character activity name.

PROCESS_TYPE
is the 8-character process type.

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>ACTIVITY_NOT_FOUND,</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_BUFFER_LENGTH</td>
</tr>
</tbody>
</table>

BAMR gate, COMMIT_BROWSE function

The COMMIT_BROWSE function of the BAMR gate is used to release any CICS BTS browses associated with this UOW.

Input parameters

CHAIN_HEAD
pointer to the head of the browse chain.

Output parameters

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

BACR gate, DELETE_CONTAINER function

The DELETE_CONTAINER function of the BACR gate is used to delete a named container and its associated data.

Input parameters

CONTAINER_NAME
is the 16-character container name.
ACTIVITY_NAME
is the 16-character activity name.

CONTAINER_SCOPE
identifies the scope of this container. It can the values:
CHILD_ACTIVITY|ACTIVITY|PROCESS|
ACQUIRED_ACTIVITY|ACQUIRED_PROCESS

Output parameters
RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>ACTIVITY_NOT_FOUND, CONTAINER_NOT_FOUND, NO_ACQUIRED_PROCESS,</td>
</tr>
<tr>
<td></td>
<td>NO_ACQUIRED_ACTIVITY, NO_CURRENT_PROCESS, NO_CURRENT_ACTIVITY, RECORD_BUSY, CONTAINER_READONLY</td>
</tr>
</tbody>
</table>

BACR gate, GET_CONTAINER_INTO function
The GET_CONTAINER_INTO function of the BACR gate is used to place the data in a named container into an area provided by the caller.

Input parameters
CONTAINER_NAME
is the 16-character container name.

ACTIVITY_NAME
is the 16-character activity name.

CONTAINER_SCOPE
identifies the scope of this container. It can the values:
CHILD_ACTIVITY|ACTIVITY|PROCESS|
ACQUIRED_ACTIVITY|ACQUIRED_PROCESS

ITEM_BUFFER
is the buffer into which the container data is placed.

Output parameters
RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>ACTIVITY_NOT_FOUND, CONTAINER_NOT_FOUND, LENGTH_ERROR,</td>
</tr>
<tr>
<td></td>
<td>NO_ACQUIRED_PROCESS, NO_ACQUIRED_ACTIVITY, NO_CURRENT_ACTIVITY, RECORD_BUSY</td>
</tr>
<tr>
<td></td>
<td>NO_CURRENT_PROCESS, RECORD_BUSY</td>
</tr>
</tbody>
</table>
BACR gate, GET_CONTAINER_SET function

The GET_CONTAINER_SET function of the BACR gate is used to place the data in a named container into an area provided by BAM domain and return this area to the caller.

Input parameters

CONTAINER_NAME
is the 16-character container name.

ACTIVITY_NAME
is the 16-character activity name.

CONTAINER_SCOPE
identifies the scope of this container. It can the values:

CHILD_ACTIVITY|ACTIVITY|PROCESS|
ACQUIRED_ACTIVITY|ACQUIRED_PROCESS

Output parameters

ITEM_DATA
a block holding the named container’s data.

RESPONSE
is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>ACTIVITY_NOT_FOUND, CONTAINER_NOT_FOUND, NO_ACQUIRED_PROCESS, NO_ACQUIRED_ACTIVITY, NO_CURRENT_ACTIVITY, NO_CURRENT_PROCESS, RECORD_BUSY</td>
</tr>
</tbody>
</table>

BACR gate, PUT_CONTAINER function

The PUT_CONTAINER function of the BACR gate is used to place data into a named container.

Input parameters

CONTAINER_NAME
is the 16-character container name.

ACTIVITY_NAME
is the 16-character activity name.

CONTAINER_SCOPE
identifies the scope of this container. It can the values:

CHILD_ACTIVITY|ACTIVITY|PROCESS|
ACQUIRED_ACTIVITY|ACQUIRED_PROCESS

ITEM_DATA
a block holding the data to be placed in the named container.

Output parameters

RESPONSE
is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED
[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>ACTIVITY_NOT_FOUND, CONTAINER_NOT_FOUND, LENGTH_ERROR, NO_ACQUIRED_PROCESS, NO_ACQUIRED_ACTIVITY, NO_CURRENT_ACTIVITY, NO_CURRENT_PROCESS, INVALID_CONTAINER_NAME, CONTAINER_READONLY, RECORD_BUSY</td>
</tr>
</tbody>
</table>

**BAGD format, INQUIRE_DATA_LENGTH function**

The INQUIRE_DATA_LENGTH function of the BAGD format is used by BAM domain to query the called domain as to the size of the flattened data which is to be included in the activity record.

**Input parameters**

DATA_TOKEN
a token representing the data.

**Output parameters**

DATA_LENGTH
the length of the flattened data.

RESPONSE
is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | INVALID | PURGED | KERNERROR | DISASTER

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>INVALID_TOKEN</td>
</tr>
</tbody>
</table>

**BAGD format, DESTROY_TOKEN function**

The DESTROY_TOKEN function of the BAGD format is used by BAM domain to tell interested parties (EM domain) to destroy their data token.

**Input parameters**

DATA_TOKEN
a token representing the data.

**Output parameters**

RESPONSE
is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | INVALID | PURGED | KERNERROR | DISASTER

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>INVALID_TOKEN</td>
</tr>
</tbody>
</table>
Business application manager domain (BAM)

**BAGD format, ADDRESS_DATA function**

The ADDRESS_DATA function of the BAGD format is a call made to BAM domain which returns the length of the calling domain’s data in the activity record.

**Input parameters**

- **ACTIVITYID**
  
a block to hold the activity identifier.

- **ACQUIRED_ACTIVITY**
  
indicates if this is an acquired activity. It can take the values:

  YES|NO

**Output parameters**

- **DATA_BLOCK**
  
a block containing the flattened data.

- **ACTIVITY_TOKEN**
  
a token representing the activity.

- **RESPONSE**
  
is the domain’s response to the call. It can have any of these values:

  OK|EXCEPTION|INVALID|PURGED|KERNERROR|DISASTER

  [REASON]

  is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>ACTIVITY_NOT_FOUND, NO_CURRENT_ACTIVITY, FILE_NOT_AUTH</td>
</tr>
</tbody>
</table>

**BAGD format, RELEASE_DATA function**

The RELEASE_DATA function of the BAGD format is a call made to BAM domain which releases the calling domain’s storage associated with the identified activity.

**Input parameters**

- **ACTIVITY_TOKEN**
  
a token representing the activity.

**Output parameters**

- **RESPONSE**
  
is the domain’s response to the call. It can have any of these values:

  OK|EXCEPTION|INVALID|PURGED|KERNERROR|DISASTER

  [REASON]

  is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>INVALID_TOKEN</td>
</tr>
</tbody>
</table>

**Business application manager domain’s generic gates**

[Table 8 on page 231] summarizes the business application manager domain’s generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.
Table 8. Business application manager domain’s generic gates

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMDM</td>
<td>BA 0101</td>
<td>PRE_INITIALISE</td>
<td>DMDM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INITIALISE_DOMAIN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>QUIESCE_DOMAIN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TERMINATE_DOMAIN</td>
<td></td>
</tr>
<tr>
<td>RMRO</td>
<td>BA 0140</td>
<td>PERFORM_PREPARE</td>
<td>RMRO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PERFORM_COMMIT</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>START_BACKOUT</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>DELIVER_BACKOUT_DATA</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>END_BACKOUT</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PERFORM_SHUNT</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PERFRM_UNSHUNT</td>
<td></td>
</tr>
<tr>
<td>RMKP</td>
<td>BA 0140</td>
<td>TAKE_KEYPOINT</td>
<td>RMKP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RMDE</td>
<td>BA 0140</td>
<td>START_DELIVERY</td>
<td>RMDE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DELIVER_RECOVERY</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>END_DELIVERY</td>
<td></td>
</tr>
<tr>
<td>APUE</td>
<td>BA 0180</td>
<td>SET_EXIT_STATUS</td>
<td>APUE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For descriptions of these functions and their input and output parameters, refer to the §§ dealing with the corresponding generic formats:

- **Functions and parameters**
  - Format DMDM—“Domain manager domain’s generic formats” on page 377
  - Format RMRO—“Recovery Manager domain’s call back formats” on page 955
  - Format RMKP—“Recovery Manager domain’s call back formats” on page 955
  - Format RMDE—“Recovery Manager domain’s call back formats” on page 955
  - Format APUE—“Application domain’s generic formats” on page 106

**Modules**

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHBADM</td>
<td>DFHBADM is the gate module for the following requests:</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>PRE_INITIALISE</td>
</tr>
<tr>
<td></td>
<td>INITIALISE_DOMAIN</td>
</tr>
<tr>
<td></td>
<td>QUIESCE_DOMAIN</td>
</tr>
<tr>
<td></td>
<td>TERMINATE_DOMAIN</td>
</tr>
</tbody>
</table>
### Business application manager domain (BAM)

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
</table>
| DFHBATT | DFHBATT is the gate module for the following requests:  
|         | ADD_REPLACE_PROCESSTYPE  
|         | INQUIRE_PROCESSTYPE  
|         | START_BROWSE_PROCESSTYPE  
|         | GET_NEXT_PROCESSTYPE  
|         | END_BROWSE_PROCESSTYPE  
|         | DISCARD_PROCESSTYPE  
|         | COMMIT_PROCESSTYPE_TABLE |
| DFHBAAC | DFHBAAC is the gate module for the following requests:  
|         | ADD_ACTIVITY  
|         | RUN_ACTIVITY  
|         | CHECK_ACTIVITY  
|         | RETURN_END_ACTIVITY  
|         | DELETE_ACTIVITY  
|         | SUSPEND_ACTIVITY  
|         | RESUME_ACTIVITY  
|         | CANCEL_ACTIVITY  
|         | LINK_ACTIVITY  
|         | ACQUIRE_ACTIVITY  
|         | RESET_ACTIVITY  
|         | ADD_TIMER_REQUEST  
|         | ADD_REATTACH_ACQUIRED |
| DFHBAPR | DFHBAPR is the gate module for the following requests:  
|         | ADD_PROCESS  
|         | RUN_PROCESS  
|         | CHECK_PROCESS  
|         | SUSPEND_PROCESS  
|         | RESUME_PROCESS  
|         | CANCEL_PROCESS  
|         | LINK_PROCESS  
|         | ACQUIRE_PROCESS  
|         | RESET_PROCESS |
| DFHBACR | DFHBACR is the gate module for the following requests:  
|         | DELETE_CONTAINER  
|         | GET_CONTAINER_INTO  
|         | GET_CONTAINER_SET  
|         | PUT_CONTAINER |
| DFHBAXM | DFHBAXM is the gate module for the following requests:  
|         | INIT_ACTIVITY_REQUEST  
|         | BIND_ACTIVITY_REQUEST |
| DFHBagD | DFHBagD is the gate module for the following requests:  
|         | INQUIRE_DATA_LENGTH  
|         | GET_DATA  
|         | DESTROY_TOKEN  
|         | ADDRESS_DATA  
|         | RELEASE_DATA |
### Business application manager domain (BAM)

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHBABR</td>
<td>DFHBABR is the gate module for the following requests:</td>
</tr>
<tr>
<td></td>
<td>STARTBR_ACTIVITY</td>
</tr>
<tr>
<td></td>
<td>GETNEXT_ACTIVITY</td>
</tr>
<tr>
<td></td>
<td>ENDBR_ACTIVITY</td>
</tr>
<tr>
<td></td>
<td>INQUIRE_ACTIVITY</td>
</tr>
<tr>
<td></td>
<td>STARTBR_CONTAINER</td>
</tr>
<tr>
<td></td>
<td>GETNEXT_CONTAINER</td>
</tr>
<tr>
<td></td>
<td>ENDBR_CONTAINER</td>
</tr>
<tr>
<td></td>
<td>INQUIRE_CONTAINER</td>
</tr>
<tr>
<td></td>
<td>STARTBR_PROCESS</td>
</tr>
<tr>
<td></td>
<td>GETNEXT_PROCESS</td>
</tr>
<tr>
<td></td>
<td>ENDBR_PROCESS</td>
</tr>
<tr>
<td></td>
<td>INQUIRE_PROCESS</td>
</tr>
<tr>
<td></td>
<td>INQUIRE_ACTIVATION</td>
</tr>
<tr>
<td></td>
<td>COMMIT_BROWSE</td>
</tr>
<tr>
<td>DFHBASP</td>
<td>DFHBASP is the gate module for the following requests:</td>
</tr>
<tr>
<td></td>
<td>PERFORM_PREPARE</td>
</tr>
<tr>
<td></td>
<td>PERFORM_COMMIT</td>
</tr>
<tr>
<td></td>
<td>PERFORM_SHUNT</td>
</tr>
<tr>
<td></td>
<td>PERFORM_UNSHUNT</td>
</tr>
<tr>
<td></td>
<td>START_BACKOUT</td>
</tr>
<tr>
<td></td>
<td>DELIVER_BACKOUT_DATA</td>
</tr>
<tr>
<td></td>
<td>END_BACKOUT</td>
</tr>
<tr>
<td></td>
<td>START_RECOVERY</td>
</tr>
<tr>
<td></td>
<td>DELIVER_RECOVERY</td>
</tr>
<tr>
<td></td>
<td>END_RECOVERY</td>
</tr>
<tr>
<td></td>
<td>TAKE_KEYPOINT</td>
</tr>
<tr>
<td>DFHBAUE</td>
<td>DFHBAUE is the gate module for the following requests:</td>
</tr>
<tr>
<td></td>
<td>SET_EXIT_STATUS</td>
</tr>
<tr>
<td>DFHBAC0</td>
<td>Implements general activity class methods.</td>
</tr>
<tr>
<td>DFHBAC1</td>
<td>Initialises the activity class.</td>
</tr>
<tr>
<td>DFHBAC2</td>
<td>Implements the prepare method of the activity class.</td>
</tr>
<tr>
<td>DFHBAC3</td>
<td>Implements the commit method of the activity class.</td>
</tr>
<tr>
<td>DFHBAC4</td>
<td>Implements the delete method of the activity class.</td>
</tr>
<tr>
<td>DFHBAC5</td>
<td>Implements the set_complete method of the activity class.</td>
</tr>
<tr>
<td>DFHBAC6</td>
<td>Implements the invoke_exit method of the activity class.</td>
</tr>
<tr>
<td>DFHBAA10</td>
<td>Implements the read_activity method of the activity class.</td>
</tr>
<tr>
<td>DFHBAA11</td>
<td>Implements the get_activity_instance method of the activity class.</td>
</tr>
<tr>
<td>DFHBAA12</td>
<td>Implements the run_sync method of the activity class.</td>
</tr>
<tr>
<td>DFHBAA1R1</td>
<td>Initialises the audit class.</td>
</tr>
<tr>
<td>DFHBAA1R2</td>
<td>Implements the write method of the audit class.</td>
</tr>
<tr>
<td>DFHBAPR0</td>
<td>Implements general process class methods.</td>
</tr>
<tr>
<td>DFHBAVP1</td>
<td>Initialises the variable length subpool class.</td>
</tr>
<tr>
<td>DFHBAOFI</td>
<td>Initialises the object factory class.</td>
</tr>
<tr>
<td>DFHBABU1</td>
<td>Initialises the buffer class.</td>
</tr>
<tr>
<td>DFHBAP1T1</td>
<td>Initialises the processtype class.</td>
</tr>
<tr>
<td>DFHBAP1T2</td>
<td>Implements the rebuild_table method of the procsesstype class.</td>
</tr>
</tbody>
</table>
### Business application manager domain (BAM)

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHBAPT3</td>
<td>Implements the purge_catalog method of the processtype class.</td>
</tr>
<tr>
<td>DFHBALR2</td>
<td>Implements the create_key method of the logical record class.</td>
</tr>
<tr>
<td>DFHBALR3</td>
<td>Implements the write_buffer method of the logical record class.</td>
</tr>
<tr>
<td>DFHBALR4</td>
<td>Implements the read_key method of the logical record class.</td>
</tr>
<tr>
<td>DFHBALR5</td>
<td>Implements the read_record method of the logical record class.</td>
</tr>
<tr>
<td>DFHBALR6</td>
<td>Implements the delete_record method of the logical record class.</td>
</tr>
<tr>
<td>DFHBALR7</td>
<td>Implements the get_browse_token method of the logical record class.</td>
</tr>
<tr>
<td>DFHBALR8</td>
<td>Implements the read_next_record method of the logical record class.</td>
</tr>
<tr>
<td>DFHBALR9</td>
<td>Implements the release_browse_token method of the logical record class.</td>
</tr>
<tr>
<td>DFHBARUP</td>
<td>The BTS repository utility program.</td>
</tr>
<tr>
<td>DFHBARUC</td>
<td>The BTS repository utility program.</td>
</tr>
<tr>
<td>DFHBARUD</td>
<td>The BTS repository utility program.</td>
</tr>
<tr>
<td>DFHBADUF</td>
<td>Formats the BAM domain control blocks</td>
</tr>
<tr>
<td>DFHBADU1</td>
<td>Formats the BAM domain control blocks</td>
</tr>
<tr>
<td>DFHBATRI</td>
<td>Interprets BAM domain trace entries</td>
</tr>
</tbody>
</table>

### Exits

There are two user exit points in BAM domain, XRSINDI and XBADEACT. See the CICS Customization Guide for further details.

### Trace

The point IDs for the business application manager domain are of the form BA xxxx; the corresponding trace levels are BA 1, BA 2, and Exc.

For more information about the trace points, see the CICS User’s Handbook. For more information about using traces in problem determination, see the CICS Problem Determination Guide.
Chapter 14. CICS-AD/Cycle Language Environment/370 interface

This section describes the run-time interface between CICS and IBM Systems Application Architecture (SAA) AD/Cycle Language Environment/370 that supports the execution of CICS application programs written to use AD/Cycle Language Environment/370, hereafter usually abbreviated to “Language Environment/370”.

Design overview

Communication between CICS and Language Environment/370 is made by calling a special Language Environment/370 interface module (CEECCICS) and passing to it a parameter list (addressed by register 1), which consists of an indication of the function to be performed and a set of address pointers to data values or areas.

Module CEECCICS is distributed in the Language Environment/370 library, but must be copied to an authorized library defined in the STEPLIB concatenation of the CICS startup job stream (see the CICS System Definition Guide).

All calls to Language Environment/370 are made directly from the CICS language interface module DFHAPLI. This module is called by several components of CICS to perform specific functions. Table 9 lists those functions, and shows the name of the CICS module initiating each function call and the Language Environment/370 call made by DFHAPLI to support the function. The format of each call parameter list is given in “External interfaces” on page 240.

Table 9. CICS-AD/Cycle Language Environment/370 interface calls

<table>
<thead>
<tr>
<th>Function</th>
<th>Module</th>
<th>LE/370 call</th>
</tr>
</thead>
<tbody>
<tr>
<td>Terminate Languages</td>
<td>DFHSTP</td>
<td>Partition Termination</td>
</tr>
<tr>
<td>Establish Language</td>
<td>DFHPGLK,</td>
<td>Establish Ownership Type</td>
</tr>
<tr>
<td></td>
<td>DFHPGLU,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DFHPGPG</td>
<td></td>
</tr>
<tr>
<td>Start Program</td>
<td>DFHPGLK,</td>
<td>Thread Initialization Run Unit Initializa</td>
</tr>
<tr>
<td></td>
<td>DFHPGLU</td>
<td>tion Run Unit Begin Invocation Run Unit End In</td>
</tr>
<tr>
<td></td>
<td></td>
<td>vocation Run Unit End Termination</td>
</tr>
<tr>
<td>Goto</td>
<td>DFHEIP</td>
<td>Perform Goto</td>
</tr>
<tr>
<td>Find Program Attributes</td>
<td>DFHEDFX</td>
<td>Determine Working Storage</td>
</tr>
<tr>
<td>Initialize Languages</td>
<td>DFHSIJ1</td>
<td>Partition Initialization</td>
</tr>
</tbody>
</table>

The logical relationship between the different calls is shown in Figure 36 on page 236.
Partition initialization

CICS

Establish ownership type

Once only per program

Thread initialization

Task initialization

Run-unit initialization

Run-unit begin invocation

Determine working storage

Perform GOTO

Run-unit end invocation

Run-unit termination

Thread termination

Partition termination

Figure 36. CICS-AD/Cycle Language Environment/370 interface components
CICS-AD/Cycle Language Environment/370 interface

**Note:** The actual passing of control to CEECCICS is made from the CICS language interface program (DFHAPLI), which provides a single point of contact between CICS and Language Environment/370. Other modules call DFHAPLI to initiate the desired function.

All calls to DFHAPLI use either the DFHAPLIM macro (for calls from outside the CICS application domain), or the DFHLILIM macro (for calls made from within the application domain).

**Establishing the connection**

The procedure for establishing the initial connection with Language Environment/370 is as follows:

1. **Load CEECCICS.** At CICS startup, DFHSIJ1 invokes DFHAPLI to “initialize languages”. DFHAPLI issues a BLDL for CEECCICS, followed by an MVS LOAD macro.

2. **Initialize contact with Language Environment/370.** Contact is first made with Language Environment/370 by having CICS drive the partition initialization function. DFHAPLI attempts partition initialization only if the earlier load of CEECCICS was successful; otherwise, the logic is bypassed.

   If the Language Environment/370 partition initialization is successful, and Language Environment/370 indicates that it can support the running of programs in languages supported by CICS, a flag is set and no further processing takes place.

   If the partition initialization function fails, CICS issues error message DFHAP1200. CICS then attempts initialization of the VS COBOL II and C/370 environments separately.

**Application program contact with Language Environment/370.** Whenever a program written in a supported language is run, the application’s attempt to make contact with Language Environment/370 fails if the “Language Environment/370 initialization bits” flag is not set. CICS then tries to run the program itself using the basic support for the language. If this fails, CICS then abends the transaction and sets the associated installed resource definition as disabled.

The following sections describe application programs that can use the Language Environment/370 support as being “Language Environment/370 enabled”. This term means that the program has been defined in the CSD as LANGUAGE(LE370) or the program has been compiled by a Language Environment/370-enabled compiler. For further information about enabling application programs to use Language Environment/370 support, see the CICS System Definition Guide.

**Storage for the transaction**

A set of work areas is required during the lifetime of any task that includes one or more programs supported by Language Environment/370. This set is known as the “language interface work area”. It is shared by any additional VS COBOL II or C/370 programs that form part of the task.

The language interface work area contains storage for the following:

- The largest possible CICS-Language Environment/370 interface parameter list (currently 15 parameter elements, but with space allowed for a further three elements)
- A general-purpose register save area for use by DFHAPLI
- A general-purpose register save area for use by Language Environment/370
CICS-AD/Cycle Language Environment/370 interface

- A 240-byte special work area for use by Language Environment/370 as the equivalent of DFHEISTG for CICS
- A 4-byte Language Environment/370 reason code field
- The IOINFO area (see page 244)
- The PGMINFO1 area (see page 245)
- The program termination block (see page 247).

Also, a thread work area is required if Language Environment/370 is involved in the running of the task. The length of a thread work area is a constant value that is notified to CICS by Language Environment/370 during the partition initialization processing. This additional work area is built contiguous with the language interface work area if the transaction is known to contain one or more programs that are Language Environment/370 enabled. When such a program is first encountered, DFHAPLI:

1. Gets from the transaction manager the address of the transaction-related instance data.
2. Flags the data to tell the transaction manager that the transaction runs Language Environment/370 application programs.
3. Adds the length of the language interface work area to the total user storage length for that transaction.

This forces the transaction manager to acquire extra storage, during task initialization, as an extension to the language interface work area. For the first occurrence only, DFHAPLI acquires the thread work area.

Further areas known as run-unit work areas (RUWAs) are required at run time if the transaction includes one or more programs that are Language Environment/370 enabled. The length of an RUWA varies for each program. The lengths required for work areas above and below the 16MB line by Language Environment/370 are notified to CICS during the processing to establish ownership type for that program; thereafter they can be found in the program’s installed resource definition. CICS adds to the length for the RUWA above the 16MB line a fixed amount for its own purposes before acquiring the storage.

Storage acquisition

During task initialization, the transaction manager acquires an area of storage, the language interface work area, which is large enough to hold all required data for calls to Language Environment/370. This area is contiguous with the EXEC interface storage (EIS), and its address is saved in TCACEEPT in the TCA.

The thread work area is usually contiguous with the language interface work area. Its address is always held in CEE_TWA in the language interface work area.

For every link level entered during the execution of the application, a run-unit work area must be acquired by CICS and its address passed to Language Environment/370 during run-unit initialization. Its address is placed in EIORUSTG in the EXEC interface storage (EIS).
Control blocks

The main control block is the language interface work area. This area is shared by any additional VS COBOL II or C/370 programs that form part of the task. It is addressed by TCACEEPT in the TCA. For programs supported by Language Environment/370, the work area is mapped by the Language_Interface_Workarea DSECT.

Modules

The CICS-AD/Cycle Language Environment/370 interface is accessed in the language interface program (DFHAPLI) in response to calls from the following modules:

DFHSIJ1, DFHEIP, DFHEDFX, and DFHSTP.

Exits

No global user exit points are provided for this interface.

Trace

Trace entries are made on entry to and exit from DFHAPLI.

Point IDs AP 1940 to AP 1945, with a trace level of PC 1, correspond to these trace entries.

The function information is always interpreted.

For entry trace records, the program name and link level are also interpreted where applicable.

For exit trace records, the returned reason code is interpreted.

Also, all calls into and out of the language environments are traced at level 1. The point IDs are: AP1948 to AP 1952.

For more information about the trace points, see the CICS Trace Entries. For more information about using traces in problem determination, see the CICS Problem Determination Guide.

(The CICS Trace Entries includes tables that show, for entry and exit trace records, the ERTLI function together with any other data items traced.)

The ERTLI function named in the DFHAPLI entry trace is the function requested on the call, while that named in the DFHAPLI exit trace is the ERTLI function most recently processed. There are some situations in which a trace record made on entry to DFHAPLI is not matched by a corresponding exit trace for the same ERTLI function. In particular, after making a call to Language Environment/370 for thread initialization, DFHAPLI does not return to the caller, but proceeds with “run-unit initialization” and “run-unit begin invocation” before finally returning. Another example is the successful execution of a “perform GOTO” function, which results in DFHAPLI not returning to the caller.

Note: ERTLI refers to the Extended Run-Time Language Interface. This is an extension of the Run-Time Language Interface (RTLI) protocols that were
CICS-AD/Cycle Language Environment/370 interface

defined to assist communication between CICS and both VS COBOL II and C/370. ERTLI includes communication between CICS and Language Environment/370.

External interfaces

This section describes the parameter lists and work areas used for the functions provided by the CICS-AD/Cycle Language Environment/370 interface.

CICS-AD/Cycle Language Environment/370 interface parameter lists

The following tables show the layout and contents of the parameter lists for the functions provided by the Language Environment/370 interface module CEECCICS.

<table>
<thead>
<tr>
<th>No.</th>
<th>Parameter name</th>
<th>Description</th>
<th>Receiver field</th>
<th>Data length</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>FUNCTION</td>
<td>F&quot;10&quot; (= Partition initialization)</td>
<td>Yes</td>
<td>F’word</td>
</tr>
<tr>
<td>2</td>
<td>RSNCODE</td>
<td>Reason code</td>
<td>Yes</td>
<td>F’word</td>
</tr>
<tr>
<td>3</td>
<td>SYSEIB</td>
<td>Address of system EIB</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>PREASA</td>
<td>Preallocated save area</td>
<td>240</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>PTOKEN</td>
<td>Language Environment/370 partition token</td>
<td>Yes</td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>EIBLEN</td>
<td>Length of CICS EIB</td>
<td>Yes</td>
<td>F’word</td>
</tr>
<tr>
<td>7</td>
<td>TWALEN</td>
<td>Thread work area length</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>CELLEVEL</td>
<td>Language Environment/370-CICS interface level</td>
<td>Yes</td>
<td>F’word</td>
</tr>
<tr>
<td>9</td>
<td>GETCAA</td>
<td>Get-CAA routine address</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>SETCAA</td>
<td>Set-CAA routine address</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>LANGDEF</td>
<td>Language modules defined</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>LANGBITS</td>
<td>Language availability bits</td>
<td>Yes</td>
<td>F’word</td>
</tr>
</tbody>
</table>

Table 11. CICS-Language Environment/370 ESTABLISH_OWNERSHIP_TYPE parameter list

<table>
<thead>
<tr>
<th>No.</th>
<th>Parameter name</th>
<th>Description</th>
<th>Receiver field</th>
<th>Data length</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>FUNCTION</td>
<td>F&quot;50&quot; (= Establish ownership type)</td>
<td>Yes</td>
<td>F’word</td>
</tr>
<tr>
<td>2</td>
<td>RSNCODE</td>
<td>Reason code</td>
<td>Yes</td>
<td>F’word</td>
</tr>
<tr>
<td>3</td>
<td>SYSEIB</td>
<td>Address of system EIB</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>PREASA</td>
<td>Preallocated save area</td>
<td>240</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>PTOKEN</td>
<td>Language Environment/370 partition token</td>
<td>Yes</td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>reserved</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>reserved</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>PGMINFO1</td>
<td>CICS-Language Environment/370 program information</td>
<td>48</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>PGMINFO2</td>
<td>Language Environment/370-CICS program information</td>
<td>Yes</td>
<td>20</td>
</tr>
</tbody>
</table>
### Table 12. CICS-Language Environment/370 THREAD_INITIALIZATION parameter list

<table>
<thead>
<tr>
<th>No.</th>
<th>Parameter name</th>
<th>Description</th>
<th>Receiver field</th>
<th>Data length</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>FUNCTION</td>
<td>F&quot;20&quot; (= Thread initialization)</td>
<td></td>
<td>F'word</td>
</tr>
<tr>
<td>2</td>
<td>RSNCODE</td>
<td>Reason code</td>
<td>Yes</td>
<td>F'word</td>
</tr>
<tr>
<td>3</td>
<td>SYSEIB</td>
<td>Address of system EIB</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>PREASA</td>
<td>Preallocated save area</td>
<td></td>
<td>240</td>
</tr>
<tr>
<td>5</td>
<td>PTOKEN</td>
<td>Language Environment/370 partition token</td>
<td></td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>TTOKEN</td>
<td>Thread token</td>
<td>Yes</td>
<td>8</td>
</tr>
<tr>
<td>7</td>
<td>PREATWA</td>
<td>Address of preallocated thread work area</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>PGMINFO1</td>
<td>CICS-Language Environment/370 program information</td>
<td></td>
<td>48</td>
</tr>
<tr>
<td>9</td>
<td>PGMINFO2</td>
<td>Language Environment/370-CICS program information</td>
<td></td>
<td>20</td>
</tr>
</tbody>
</table>

### Table 13. CICS-Language Environment/370 RUNUNIT_INITIALIZATION parameter list

<table>
<thead>
<tr>
<th>No.</th>
<th>Parameter name</th>
<th>Description</th>
<th>Receiver field</th>
<th>Data length</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>FUNCTION</td>
<td>F&quot;30&quot; (= Run-unit initialization)</td>
<td></td>
<td>F'word</td>
</tr>
<tr>
<td>2</td>
<td>RSNCODE</td>
<td>Reason code</td>
<td>Yes</td>
<td>F'word</td>
</tr>
<tr>
<td>3</td>
<td>SYSEIB</td>
<td>Address of system EIB</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>PREASA</td>
<td>Preallocated save area</td>
<td></td>
<td>240</td>
</tr>
<tr>
<td>5</td>
<td>PTOKEN</td>
<td>Language Environment/370 partition token</td>
<td></td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>TTOKEN</td>
<td>Thread token</td>
<td></td>
<td>8</td>
</tr>
<tr>
<td>7</td>
<td>RTOKEN</td>
<td>Run-unit token</td>
<td>Yes</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>PGMINFO1</td>
<td>CICS-Language Environment/370 program information</td>
<td></td>
<td>48</td>
</tr>
<tr>
<td>9</td>
<td>PGMINFO2</td>
<td>Language Environment/370-CICS program information</td>
<td></td>
<td>20</td>
</tr>
</tbody>
</table>
### CICS-AD/Cycle Language Environment/370 interface

#### Table 14. CICS-Language Environment/370 RUNUNIT_BEGIN_INVOCATION parameter list

<table>
<thead>
<tr>
<th>No.</th>
<th>Parameter name</th>
<th>Description</th>
<th>Receiver field</th>
<th>Data length</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>FUNCTION</td>
<td>F”32” (= Run-unit begin invocation)</td>
<td>F’word</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>RSNCODE</td>
<td>Reason code</td>
<td>Yes</td>
<td>F’word</td>
</tr>
<tr>
<td>3</td>
<td>SYSEIB</td>
<td>Address of system EIB</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>PREASA</td>
<td>Preallocated save area</td>
<td>240</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>PTOKEN</td>
<td>Language Environment/370 partition token</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>TTOKEN</td>
<td>Thread token</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>RTOKEN</td>
<td>Run-unit token</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>PGMINFO1</td>
<td>CICS-Language Environment/370 program information</td>
<td>48</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>PGMINFO2</td>
<td>Language Environment/370-CICS program information</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>IOINFO</td>
<td>Input/output queue details</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>RSA</td>
<td>RSA at last EXEC CICS command</td>
<td>F’word</td>
<td></td>
</tr>
</tbody>
</table>

#### Table 15. CICS-Language Environment/370 DETERMINE_WORKING_STORAGE parameter list

<table>
<thead>
<tr>
<th>No.</th>
<th>Parameter name</th>
<th>Description</th>
<th>Receiver field</th>
<th>Data length</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>FUNCTION</td>
<td>F”60” (= Determine working storage)</td>
<td>F’word</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>RSNCODE</td>
<td>Reason code</td>
<td>Yes</td>
<td>F’word</td>
</tr>
<tr>
<td>3</td>
<td>SYSEIB</td>
<td>Address of system EIB</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>PREASA</td>
<td>Preallocated save area</td>
<td>240</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>PTOKEN</td>
<td>Language Environment/370 partition token</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>TTOKEN</td>
<td>Thread token</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>RTOKEN</td>
<td>Run-unit token</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>LANG</td>
<td>Program language bits</td>
<td>F’word</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>PGMRSA</td>
<td>Register save area address</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>WSA</td>
<td>Working storage address</td>
<td>Yes</td>
<td>4</td>
</tr>
<tr>
<td>11</td>
<td>WSL</td>
<td>Working storage length</td>
<td>Yes</td>
<td>F’word</td>
</tr>
<tr>
<td>12</td>
<td>SSA</td>
<td>Static storage address (reserved)</td>
<td>Yes</td>
<td>4</td>
</tr>
<tr>
<td>13</td>
<td>SSL</td>
<td>Static storage length (reserved)</td>
<td>Yes</td>
<td>F’word</td>
</tr>
<tr>
<td>14</td>
<td>EP</td>
<td>Program entry point</td>
<td>Yes</td>
<td>4</td>
</tr>
</tbody>
</table>
### Table 16. CICS-Language Environment/370 PERFORM_GOTO parameter list

<table>
<thead>
<tr>
<th>No.</th>
<th>Parameter name</th>
<th>Description</th>
<th>Receiver field</th>
<th>Data length</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>FUNCTION</td>
<td>F’70’ (= Perform GOTO)</td>
<td>F’word</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>RSNCODE</td>
<td>Reason code</td>
<td>Yes</td>
<td>F’word</td>
</tr>
<tr>
<td>3</td>
<td>SYSEIB</td>
<td>Address of system EIB</td>
<td>F’word</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>PREASA</td>
<td>Preallocated save area</td>
<td>F’word</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>PTOKEN</td>
<td>Language Environment/370 partition token</td>
<td>F’word</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>TTOKEN</td>
<td>Thread token</td>
<td>Yes</td>
<td>F’word</td>
</tr>
<tr>
<td>7</td>
<td>RTOKEN</td>
<td>Run-unit token</td>
<td>F’word</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>LANG</td>
<td>Program language bits</td>
<td>F’word</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>LABEL</td>
<td>Label argument at Handle</td>
<td>F’word</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>RSA</td>
<td>RSA at last EXEC CICS command</td>
<td>F’word</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>CALLERR</td>
<td>Cross call error flag</td>
<td>Yes</td>
<td>F’word</td>
</tr>
<tr>
<td>12</td>
<td>ABCODE</td>
<td>Address of TACB abend code</td>
<td>F’word</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>R13</td>
<td>Register 13 value at abend</td>
<td>F’word</td>
<td></td>
</tr>
</tbody>
</table>

### Table 17. CICS-Language Environment/370 RUNUNIT_END_INVOCATION parameter list

<table>
<thead>
<tr>
<th>No.</th>
<th>Parameter name</th>
<th>Description</th>
<th>Receiver field</th>
<th>Data length</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>FUNCTION</td>
<td>F’33’ (= Run-unit end invocation)</td>
<td>F’word</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>RSNCODE</td>
<td>Reason code</td>
<td>Yes</td>
<td>F’word</td>
</tr>
<tr>
<td>3</td>
<td>SYSEIB</td>
<td>Address of system EIB</td>
<td>F’word</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>PREASA</td>
<td>Preallocated save area</td>
<td>F’word</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>PTOKEN</td>
<td>Language Environment/370 partition token</td>
<td>F’word</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>TTOKEN</td>
<td>Thread token</td>
<td>Yes</td>
<td>F’word</td>
</tr>
<tr>
<td>7</td>
<td>RTOKEN</td>
<td>Run-unit token</td>
<td>F’word</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>PGMINFO1</td>
<td>CICS-Language Environment/370 program information</td>
<td>F’word</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>PGMINFO2</td>
<td>Language Environment/370-CICS program information</td>
<td>F’word</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>PTB</td>
<td>Program termination block</td>
<td>64</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>RSA</td>
<td>RSA at last EXEC CICS command</td>
<td>F’word</td>
<td></td>
</tr>
</tbody>
</table>
CICS-AD/Cycle Language Environment/370 interface

Table 18. CICS-Language Environment/370 RUNUNIT_TERMINATION parameter list

<table>
<thead>
<tr>
<th>No.</th>
<th>Parameter name</th>
<th>Description</th>
<th>Receiver field</th>
<th>Data length</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>FUNCTION</td>
<td>F&quot;31&quot; (= Run-unit termination)</td>
<td>F’word</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>RSNCODE</td>
<td>Reason code</td>
<td>Yes</td>
<td>F’word</td>
</tr>
<tr>
<td>3</td>
<td>SYSEIB</td>
<td>Address of system EIB</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>PREASA</td>
<td>Preallocated save area</td>
<td></td>
<td>240</td>
</tr>
<tr>
<td>5</td>
<td>PTOKEN</td>
<td>Language Environment/370 partition token</td>
<td></td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>TTOKEN</td>
<td>Thread token</td>
<td></td>
<td>8</td>
</tr>
<tr>
<td>7</td>
<td>RTOKEN</td>
<td>Run-unit token</td>
<td>Yes</td>
<td>8</td>
</tr>
</tbody>
</table>

Table 19. CICS-Language Environment/370 THREAD_TERMINATION parameter list

<table>
<thead>
<tr>
<th>No.</th>
<th>Parameter name</th>
<th>Description</th>
<th>Receiver field</th>
<th>Data length</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>FUNCTION</td>
<td>F&quot;21&quot; (= Thread termination)</td>
<td>F’word</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>RSNCODE</td>
<td>Reason code</td>
<td>Yes</td>
<td>F’word</td>
</tr>
<tr>
<td>3</td>
<td>SYSEIB</td>
<td>Address of system EIB</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>PREASA</td>
<td>Preallocated save area</td>
<td></td>
<td>240</td>
</tr>
<tr>
<td>5</td>
<td>PTOKEN</td>
<td>Language Environment/370 partition token</td>
<td></td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>TTOKEN</td>
<td>Thread token</td>
<td>Yes</td>
<td>8</td>
</tr>
</tbody>
</table>

Table 20. CICS-Language Environment/370 PARTITION_TERMINATION parameter list

<table>
<thead>
<tr>
<th>No.</th>
<th>Parameter name</th>
<th>Description</th>
<th>Receiver field</th>
<th>Data length</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>FUNCTION</td>
<td>F&quot;11&quot; (= Partition termination)</td>
<td>F’word</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>RSNCODE</td>
<td>Reason code</td>
<td>Yes</td>
<td>F’word</td>
</tr>
<tr>
<td>3</td>
<td>SYSEIB</td>
<td>Address of system EIB</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>PREASA</td>
<td>Preallocated save area</td>
<td></td>
<td>240</td>
</tr>
<tr>
<td>5</td>
<td>PTOKEN</td>
<td>Language Environment/370 partition token</td>
<td></td>
<td>8</td>
</tr>
</tbody>
</table>

Work areas

The following sections describe the work areas required during the lifetime of any task that includes one or more programs that use the CICS-Language Environment/370 interface.

IOINFO

The IOINFO area, which is built by DFHAPLI in the CICS-Language Environment/370 work area, is passed to Language Environment/370 on a RUNUNIT_BEGIN_INVOCATION call.

CICS applications cannot use the SYSIN and SYSPRINT data streams because such usage would conflict with the way CICS handles I/O. However, an application may require a general input or output data stream in some situations, for example,
where it is necessary to output a message to a program and the program has not
been written to expect such output under normal operation.

Three such data streams are architected for this purpose: input, output (normal),
and error output. The destinations must be either spools or queues. CICS uses
queues, so the file type is always set to "Q". Table 21 shows the transient data
queue names that are passed to Language Environment/370 (abbreviated here to
LE/370). For completeness, the table also shows the equivalent transient data
queue names that are passed to C/370 when CICS interfaces directly with C/370.

Table 21. Transient data queues for use by Language Environment/370 (and C/370)

<table>
<thead>
<tr>
<th>File type</th>
<th>LE/370 queue name</th>
<th>C/370 queue name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>CESI</td>
<td>CCSI</td>
</tr>
<tr>
<td>Output</td>
<td>CESO</td>
<td>CCSO</td>
</tr>
<tr>
<td>Error output</td>
<td>CESE</td>
<td>CCSE</td>
</tr>
</tbody>
</table>

Each data stream is identified by a 6-byte control block, and the three control
blocks are concatenated to form the IOINFO area, which CICS passes to Language
Environment/370.

IOINFO has this format (in assembler-language code):

```
IOINFO DS 0CL18 Input/output queue details
STD_IN DS 0CL6 Standard input file
QORS_IN DS CL1 ..file type - "Q" = transient data
TDQ_IN DS CL4 ..queue name
SPO_IN DS CL1 ..spool class - not used

STD_OUT DS 0CL6 Standard output file
QORS_OUT DS CL1 ..file type - "Q" = transient data
TDQ_OUT DS CL4 ..queue name
SPO_OUT DS CL1 ..spool class - not used

STD_ERR DS 0CL6 Standard error output file
QORS_ERR DS CL1 ..file type - "Q" = transient data
TDQ_ERR DS CL4 ..queue name
SPO_ERR DS CL1 ..spool class - not used
```

PGMINFO1

The PGMINFO1 area, which is built by DFHAPLI in the CICS-Language
Environment/370 work area, is passed to Language Environment/370 during these
interface calls:

```
ESTABLISH_OWNERSHIP_TYPE
THREAD_INITIALIZATION
RUNUNIT_INITIALIZATION
RUNUNIT_BEGIN_INVOCATION
RUNUNIT_END_INVOCATION
```

When both CICS and LE are capable of supporting it, the separate calls to LE for
Rununit Initialisation and Rununit Begin Invocation are combined into a single
call. This single call is a Rununit Initialisation call with additional parameters
indicating

1. make the combined call
2. whether CICS believes the RUWA being passed has already been passed to LE,
   and so need not be completely initialised by LE/370

PGMINFO1 has this format (in assembler-language code):
CICS-AD/Cycle Language Environment/370 interface

PGMINFO2

The PGMINFO2 area, which forms part of the PPT entry for the running program, is filled in by Language Environment/370 on successful completion of an ESTABLISH_OWNERSHIP_TYPE call; and is subsequently passed by CICS to Language Environment/370 during these interface calls:

- THREAD_INITIALIZATION
- RUNUNIT_INITIALIZATION
- RUNUNIT_BEGIN_INVOCATION
- RUNUNIT_END_INVOCATION

PGMINFO2 has this format (in assembler-language code):

```
PGMINFO2 DS 0F
PGMINLEN DS FL4 Length of PGMINFO2 extension
PLBRWA31 DS F Length of 31-bit RUWA
PLBRWAA EQU X'80' ..31-bit storage required (C/370)
PLBRWAL DS FL3 ..Length of 31-bit RUWA
PLBRWA24 DS F Length of 24-bit RUWA

PLBLANG DS 0CL4 Language availability byte
PLBLANG1 DS X
PLBCEENE EQU X'80' ..Language Environment/370 enabled
PLBCEELA EQU X'40' ..Language Environment/370 language known
PLBMIXED EQU X'20' ..Mixed/single language
PLBCOMPT EQU X'10' ..Compatibility
PLBEXECU EQU X'08' ..Language Environment/370 executable
PLBASSEM EQU X'04' ..Assembler-language program
PLBC370 EQU X'02' ..C/370 program
PLBCOBL2 EQU X'01' ..VS COBOL II program
PLBBLANG2 DS X
PLBOSCOB EQU X'80' ..OS/VS COBOL program
PLBPLI EQU X'40' ..OS PLI program
```
CICS-AD/Cycle Language Environment/370 interface

Program termination block

The program termination block (PTB), which is built by DFHAPLI in the CICS-Language Environment/370 work area, is passed to Language Environment/370 on a RUNUNIT_END_INVOCATION call.

It has this format (in Assembler-language code):

```
CELINFO DS 0F
PCHK DS OCL32 Abend information
   DS CL8
PCHK_PSW DS CL8 ..PSW
PCHKINTS DS OCL8 ..Interrupt data
PCHK_LEN DS XL2 ....Instruction length
PCHK_INT DS XL2 ....Interrupt code
PCHK_ADR DS FL4 ..Exception address
PCHK_GR DS AL4 ..A(GP registers at abend)
PCHK_FR DS AL4 ..A(FP registers at abend)
PCHK_AR DS AL4 ..A(AX registers at abend)
PCHK_EX DS AL4 ..A(Registers at the last time a CICS command was issued)
CNTCODE DS OCL4 Continuation code
CONT1 EQU X'40' ..retry using registers
CONT2 EQU X'20' ..retry using PSW
   DS BL3 Reserved
RTRY DS OCL20
RTRY_ADR DS FL4 ..Retry address
RTRY_PM DS AL4 ..A(Program mask)
RTRY_GR DS AL4 ..A(GP registers)
RTRY_FR DS AL4 ..A(FP registers)
RTRY_AR DS AL4 ..A(AX registers)
```
CICS-AD/Cycle Language Environment/370 interface
Chapter 15. CICS catalog domains (GC/LC)

The two CICS catalog domains, namely the local catalog (LC) domain and the
global catalog (GC) domain, are repositories used by other domains to hold
information to allow an orderly restart. They enable CICS code to read, write, and
purge records on the local and global catalog data sets so that a record of the CICS
state can be maintained when CICS is not running.

These domains use a common set of programs to provide a domain interface to
VSAM KSDS data sets for the local catalog (DFHLCD) and for the global catalog
(DFHGCD). They also conceal, from the user domain, the underlying VSAM
operations.

The local catalog is initialized with the DFHCCUTL utility to contain information
that is relevant to a particular CICS system, including a list of domains.

The global catalog is used to hold information that is applicable to a whole CICS
system. Thus, in an XRF system consisting of one active and one alternate CICS
system, there are two local catalogs and one global catalog. Conversely, in a
non-XRF system, there is one local catalog and one global catalog.

The descriptions that follow relate to the common set of programs for both the
local and the global catalog domains.

CICS catalog domains’ specific gate

Table 22 summarizes the CICS catalog domains’ specific gate. It shows the level-1
trace point IDs of the modules providing the functions for the gate, the functions
provided by the gate, and whether or not the functions are available through the
exit programming interface (XPI).

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace¹</th>
<th>Function</th>
<th>XPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCCC</td>
<td>CC 2010</td>
<td>ADD</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>CC 2050</td>
<td>DELETE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GET</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WRITE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GET_UPDATE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PUT_REPLACE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>START_BROWSE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GET_NEXT</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>END_BROWSE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TYPE_PURGE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>START_WRITE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WRITE_NEXT</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>END_WRITE</td>
<td>NO</td>
</tr>
</tbody>
</table>

¹ The domain identifier part of the point ID, shown in the table as CC, appears in
a trace as either LC (local catalog domain) or GC (global catalog domain).

In many of the functions to be described, an input parameter NAME is listed. This
name is used in the construction of a VSAM key which is then used to identify a
specific record in the catalog. The record may, or may not, already exist. The key is
a string concatenation of the calling domain, the type, and the name. The type is a block of records for a domain. The choice of type and name for a specific domain is at the discretion of the calling domain.

**CCCC gate, ADD function**

The ADD function of the CCCC gate is used to add a record.

**Input parameters**

- **DATA_IN** is the data to be added to the record.
- **TYPE** identifies a block of data.
- **NAME** is used to construct a record key, together with the domain and the type.

**Output parameters**

- **RESPONSE** is the domain’s response to the call. It can have any of these values:
  - OK
  - EXCEPTION
  - DISASTER
  - INVALID
  - KERNERROR
  - PURGED

  **[REASON]** is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>DUPLICATE, INVALID_DATA_LENGTH, IO_ERROR, CATALOG_FULL</td>
</tr>
</tbody>
</table>

**CCCC gate, DELETE function**

The DELETE function of the CCCC gate is used to delete a record.

**Input parameters**

- **TYPE** identifies a block of data.
- **NAME** is used to construct a record key, together with the domain and the type.
- **[WRITE_TOKEN]** is an optional token corresponding to a START_WRITE. This avoids the need for additional connects or disconnects.

**Output parameters**

- **RESPONSE** is the domain’s response to the call. It can have any of these values:
  - OK
  - EXCEPTION
  - DISASTER
  - INVALID
  - KERNERROR
  - PURGED

  **[REASON]** is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>RECORD_NOT_FOUND, IO_ERROR, BAD_TOKEN</td>
</tr>
</tbody>
</table>

**CCCC gate, GET function**

The GET function of the CCCC gate is used to get a record.
Input parameters
DATA_OUT
  If the response is OK, this contains a copy of the specified record.
TYPE  identifies a block of data.
NAME  is used to construct a record key, together with the domain and the type.

Output parameters
RESPONSE
  is the domain’s response to the call. It can have any of these values:
  OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
  is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>RECORD_NOT_FOUND, INVALID_DATA_LENGTH, IO_ERROR</td>
</tr>
</tbody>
</table>

CCCC gate, WRITE function
The WRITE function of the CCCC gate is used to write a record.

Input parameters
DATA_OUT
  is the data to be written to the specified record.
TYPE  identifies a block of data.
NAME  is used to construct a record key, together with the domain and the type.

Output parameters
RESPONSE
  is the domain’s response to the call. It can have any of these values:
  OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
  is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>INVALID_DATA_LENGTH, IO_ERROR, CATALOG_FULL</td>
</tr>
</tbody>
</table>

CCCCC gate, GET_UPDATE function
The GET_UPDATE function of the CCCC gate is used to get a record and to establish a thread. This thread, identified by a token, is used in a corresponding PUT_REPLACE.

Input parameters
DATA_OUT
  If response is OK, this contains a copy of the record.
TYPE  identifies a block of data.
CICS catalog domains (GC/LC)

NAME

is used to construct a record key, together with the domain and the type.

Output parameters

UPDATE_TOKEN

Token to be used by the corresponding PUT_REPLACE.

RESPONSE

is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]

is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>RECORD_NOT_FOUND, INVALID_DATA_LENGTH, IO_ERROR</td>
</tr>
</tbody>
</table>

CCCCC gate, PUT_REPLACE function

The PUT_REPLACE function of the CCCC gate is used to replace a record.

Input parameters

DATA_IN

is the data to be copied to the record.

UPDATE_TOKEN

is the token obtained from a previous GET_UPDATE, used to identify an existing record in the catalog.

Output parameters

RESPONSE

is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]

is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>BAD_TOKEN, INVALID_DATA_LENGTH, IO_ERROR, CATALOG_FULL</td>
</tr>
</tbody>
</table>

CCCCC gate, START_BROWSE function

The START_BROWSE function of the CCCC gate is used to start a browse session.

Input parameters

TYPE

identifies a block of data. The browse positions itself before the first record for that type.

Output parameters

BROWSE_TOKEN

is the token identifying this browse session.

RESPONSE

is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED
[REASON]
is returned when RESPONSE is EXCEPTION. It has this value:
IO_ERROR

**CCCC gate, GET_NEXT function**
The GET_NEXT function of the CCCC gate is used to get the next record.

**Input parameters**
BROWSE_TOKEN
is the token identifying this browse session.

DATA_OUT
is a copy of the next record within the browsed type.

**Output parameters**
RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>INVALID_DATA_LENGTH, BAD_TOKEN, BROWSE_END, IO_ERROR</td>
</tr>
</tbody>
</table>

**CCCC gate, END_BROWSE function**
The END_BROWSE function of the CCCC gate is used to end a browse session.

**Input parameters**
BROWSE_TOKEN
is the token identifying this browse session.

**Output parameters**
RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>BAD_TOKEN, IO_ERROR</td>
</tr>
</tbody>
</table>

**CCCC gate, TYPE_PURGE function**
The TYPE_PURGE function of the CCCC gate is used to purge records. This deletes all records within the specified TYPE block for that domain.

**Input parameters**
TYPE identifies a block of data.
CICS catalog domains (GC/LC)

Output parameters

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>TYPE_NOT_FOUND, IO_ERROR</td>
</tr>
</tbody>
</table>

CCCC gate, START_WRITE function

The START_WRITE function of the CCCC gate is used to start a write session.

Input parameters
None.

Output parameters

WRITE_TOKEN
is the token identifying a unique file string (thread).

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. It has this value:
IO_ERROR

CCCC gate, WRITE_NEXT function

The WRITE_NEXT function of the CCCC gate is used to write the next record.

Input parameters

WRITE_TOKEN
is the token corresponding to the token from START_WRITE.

DATA_IN
is the data to be copied to the record.

TYPE
identifies a block of data.

NAME
is used to construct a record key, together with the domain and the type.

Output parameters

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>INVALID_DATA_LENGTH, IO_ERROR, CATALOG_FULL, BAD_TOKEN</td>
</tr>
</tbody>
</table>
**CICS catalog domains (GC/LC)**

**CCCC gate, END_WRITE function**

The END_WRITE function of the CCCC gate is used to end a write session.

**Input parameters**

**WRITE_TOKEN**

Token corresponding to a START_WRITE.

**Output parameters**

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON]

is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>IO_ERROR, BAD_TOKEN</td>
</tr>
</tbody>
</table>

---

**Table 23. CICS catalog domains’ generic gate**

Table 23 summarizes the CICS catalog domains’ generic gate. It shows the level-1 trace point IDs of the modules providing the functions for the gate, the functions provided by the gate, and the generic formats for calls to the gates.

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace1</th>
<th>Function</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMDM</td>
<td>CC 1010</td>
<td>PRE_INITIALISE</td>
<td>DMDM</td>
</tr>
<tr>
<td></td>
<td>CC 1040</td>
<td>INITIALISE_DOMAIN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>QUIESCE_DOMAIN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TERMINATE_DOMAIN</td>
<td></td>
</tr>
</tbody>
</table>

1 The domain identifier part of the point ID, shown in the table as CC, appears in a trace as either LC (local catalog domain) or GC (global catalog domain).

Descriptions of these functions and their input and output parameters are given in the section dealing with the corresponding generic formats. This is in format DMDM—see “Domain manager domain’s generic formats” on page 377.

In preinitialization processing, the local catalog domain opens the CICS local catalog, DFHLCD. (There is no preinitialization processing for the global catalog domain.)

In initialization processing, the global catalog domain opens the CICS global catalog, DFHGCD.

In quiesce processing, the local and global catalog domains close their respective catalog data sets.

In termination processing, the CICS catalog domains perform no termination processing. They do not close either the local catalog or the global catalog; the operating system closes these data sets.
CICS catalog domains (GC/LC)

Modules

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHCCCC</td>
<td>Handles the following functions:</td>
</tr>
<tr>
<td></td>
<td>ADD</td>
</tr>
<tr>
<td></td>
<td>DELETE</td>
</tr>
<tr>
<td></td>
<td>GET</td>
</tr>
<tr>
<td></td>
<td>WRITE</td>
</tr>
<tr>
<td></td>
<td>GET_UPDATE</td>
</tr>
<tr>
<td></td>
<td>PUT_REPLACE</td>
</tr>
<tr>
<td></td>
<td>START_BROWSE</td>
</tr>
<tr>
<td></td>
<td>GET_NEXT</td>
</tr>
<tr>
<td></td>
<td>END_BROWSE</td>
</tr>
<tr>
<td></td>
<td>TYPE_PURGE</td>
</tr>
<tr>
<td></td>
<td>START_WRITE</td>
</tr>
<tr>
<td></td>
<td>WRITE_NEXT</td>
</tr>
<tr>
<td></td>
<td>END_WRITE</td>
</tr>
<tr>
<td>DFHCCDM</td>
<td>Handles the initialization and termination of the CICS catalog domains.</td>
</tr>
<tr>
<td>DFHCCDUF</td>
<td>Catalog dump formatting routine.</td>
</tr>
<tr>
<td>DFHCCTRI</td>
<td>Trace interpreter routine for the catalog domains.</td>
</tr>
<tr>
<td>DFHCCUTL</td>
<td>Offline utility to initialize the local catalog.</td>
</tr>
</tbody>
</table>

Exits

No global user exit points are provided in these domains.

Trace

The point IDs for the local catalog domain are of the form LC xxxx; the corresponding trace levels are LC 1 and Exc.

The point IDs for the global catalog domain are of the form GC xxxx; the corresponding trace levels are GC 1 and Exc.

For more information about the trace points, see the CICS Trace Entries. For more information about using traces in problem determination, see the CICS Problem Determination Guide.
Chapter 16. CICS-C/370 interface

This section describes the run-time interface between CICS and C/370 that supports the execution of CICS application programs written in the C language. The design has much in common with CICS support for VS COBOL II.

Design overview

If IBM SAA AD/Cycle Language Environment/370 is installed and is able to support the running of C/370 application programs, CICS interfaces with AD/Cycle Language Environment/370 and not directly with C/370.

Otherwise, communication between CICS and C/370 is made by calling a special C/370 interface module (EDCCICS) and passing to it a parameter list (addressed by register 1), which consists of an indication of the function to be performed and a set of address pointers to data values or areas.

Module EDCCICS is distributed in the C/370 library, but must be copied to an authorized library defined in the STEPLIB concatenation of the CICS startup job stream (see the CICS System Definition Guide).

Table 24 lists the functions that are driven via this interface and shows the name of the CICS module initiating each function call. The format of each distinct call parameter list is given in “External interfaces” on page 261.

Table 24. CICS-C/370 interface calls

<table>
<thead>
<tr>
<th>Function</th>
<th>Module</th>
<th>C/370 call</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initialize Languages</td>
<td>DFHSIJ1</td>
<td>Partition Initialization</td>
</tr>
<tr>
<td>Terminate Languages</td>
<td>DFHSTP</td>
<td>Partition Termination</td>
</tr>
<tr>
<td>Establish Language</td>
<td>DFHPGLK,</td>
<td>Determine Program Type</td>
</tr>
<tr>
<td></td>
<td>DFHPGLU,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DFHPGPG</td>
<td></td>
</tr>
<tr>
<td>Start Program</td>
<td>DFHPGLK,</td>
<td>Thread Initialization Run Unit Initializatio</td>
</tr>
<tr>
<td></td>
<td>DFHPGGLU</td>
<td>Unit Termination Thread Termination</td>
</tr>
<tr>
<td>Find Program Attributes</td>
<td>DFHEDFX</td>
<td>Working Storage Locate</td>
</tr>
</tbody>
</table>

The logical relationship between the different calls is shown in Figure 37 on page 258.
CICS-C/370 interface

Partition initialization

Determine program type

Thread initialization

Run-unit initialization

Working storage locate

Run-unit termination

Thread termination

Partition termination

Figure 37. CICS-C/370 interface components
Note: The actual passing of control to EDCCICS is made from the CICS language interface program (DFHAPLI), which provides a single point of contact between CICS and C/370. Other modules call DFHAPLI to perform the desired function.

All calls to DFHAPLI use either the DFHAPLIM or DFHLILIM domain calls.

Establishing the connection

The procedure for establishing the initial connection with C/370 is as follows:

1. **Try to establish connection to Language Environment/370.** At CICS startup, DFHSIJ1 invokes DFHAPLI for “partition initialization”. DFHAPLI attempts to perform Language Environment/370 partition initialization. If this is successful, and Language Environment/370 indicates that it can support the running of C/370 programs, no further processing takes place for this call to DFHAPLI. Otherwise, processing continues as follows.

2. **Load EDCCICS.** DFHAPLI issues an MVS LOAD macro.

3. **Initialize contact with C/370.** Contact is made with C/370 by having CICS drive the partition initialization function. DFHAPLI attempts C/370 partition initialization only if the earlier load of EDCCICS was successful; otherwise, the logic is bypassed.

   If the C/370 partition initialization function completes, a flag is set; if it fails, CICS issues error message DFHAP1202.

**Application program contact with C/370.** Whenever a C program runs directly under CICS, the application’s attempt to make contact with C/370 fails if the “C is initialized” flag is not set. CICS then abends the transaction with abend code APCK and sets the transaction disabled.

Storage for the transaction

During the process of interfacing between CICS and C/370, storage is required for the following:

- Parameter list for CICS-C/370 calls
- Register save area for use by DFHAPLI
- Register save area for use by C/370
- C/370 special work area
- C/370 reason code
- C/370 input/output queue details
- Thread storage
- Run-unit storage

The lengths of the last two areas are initially unknown to CICS. The length of a thread work area is a constant value that is notified to CICS by C/370 during the partition initialization processing. The length of a run-unit work area varies for each program. It is notified to CICS by C/370 during the “determine program type” processing for that program; thereafter it can be found in the program language block (PLB) entry.

The determine program type call is made by the program manager domain which calls DFHAPLI when program fetch completes successfully. The information returned by C/370 is valid while the program is not refreshed with the NEWCOPY option; in that event, a new determine program type call must be made.
CICS-C/370 interface

Storage acquisition

For partition initialization and partition termination, the necessary storage for the calls is taken from the functional module’s own storage areas.

All other calls are application program related. Sufficient storage for these calls is acquired by CICS before the first task-related call to C/370, and is retained throughout the lifetime of the task. The storage area caters for all the required storage detailed above except for run-unit storage, which is acquired separately. The storage area is contiguous with the EXEC interface storage (EIS).

The address of this storage area for calls is held in TCACEEPT in the TCA.

During the current CICS lifetime, on the first occasion only that a unique C program is executed, a special “determine program type” call is made to C/370. The purpose of this call is to verify that the program is in fact written in C, and to tell CICS the length of the run-unit work area that has to be provided before control is actually passed to the program.

Thus, for every link level entered during the execution of the application, a run-unit storage area must be acquired by CICS and its address passed to C/370 during the run-unit initialization process. The length of this area will have been saved into the PLB entry by C/370 as part of the determine program type function. CICS places the address of the run-unit area into EIORUSTG from where it is freed during the processing of run-unit termination.

Control blocks

The main control block is the combined CICS-C/370 work area and call parameter list, also known as the “language interface work area”. This area is shared by any additional programs that are supported by Language Environment/370, and form part of the same task. It is addressed by TCACEEPT in the TCA. For C programs running directly under CICS, the work area is mapped by the LANGUAGE_INTERFACE_WORKAREA dsect.

Modules

The CICS-C/370 interface is accessed in the language interface program (DFHAPLI) in response to calls from the following modules:

DFHSIJ1, DFHEDFX, DFHPGLK, DFHPGLU, DFHPGPG, and DFHSTP.

Exits

No global user exit points are provided for this interface.

Trace

Trace entries are made on entry to and exit from DFHAPLI for C/370. Also, calls to and returns from the interface module EDCCICS are traced.

Point IDs AP 1940 to AP 1945, with a trace level of PC 1, correspond to these trace entries (as for the CICS-Language Environment/370 interface).
External interfaces

The DFHAPLI module builds the parameter lists that are required for invoking unique C/370 functions, and passes control to the C/370 interface module, EDCCICS.

The DFHAPLI module supports the following C/370 interface functions:

- PARTITION_INITIALIZATION
- DETERMINE_PROGRAM_TYPE
- THREAD_INITIALIZATION
- RUNUNIT_INITIALIZATION
- WORKING_STORAGE_LOCATE
- RUNUNIT_TERMINATION
- THREAD_TERMINATION
- PARTITION_TERMINATION

CICS-C/370 interface parameter lists

The following tables show the layout and contents of the parameter lists for the functions provided by the C/370 interface module EDCCICS.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Data addressed by the pointer</th>
<th>CICS field name</th>
<th>Data length</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>F&quot;10&quot; (= Partition initialization)</td>
<td></td>
<td>F&quot;word&quot;</td>
</tr>
<tr>
<td>04</td>
<td>Reason code</td>
<td>C_REASON_CODE</td>
<td>F&quot;word&quot;</td>
</tr>
<tr>
<td>08</td>
<td>Address of system EIB</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>0C</td>
<td>240-byte work area</td>
<td>C_WORKAREA</td>
<td>240</td>
</tr>
<tr>
<td>10</td>
<td>C/370 partition token storage area (receiver)</td>
<td>CSACELPT</td>
<td>8</td>
</tr>
<tr>
<td>14</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>F&quot;85&quot; (Length of EIB)</td>
<td></td>
<td>F&quot;word&quot;</td>
</tr>
<tr>
<td>1C</td>
<td>Length of preallocated thread storage (receiver)</td>
<td>CSACELTL</td>
<td>F&quot;word&quot;</td>
</tr>
<tr>
<td>20</td>
<td>C/370 interface level storage area (receiver)</td>
<td>CSACELIL</td>
<td>F&quot;word&quot;</td>
</tr>
</tbody>
</table>
### Table 26. CICS-C/370 DETERMINE_PROGRAM_TYPE parameter list

<table>
<thead>
<tr>
<th>Offset</th>
<th>Data addressed by the pointer</th>
<th>CICS field name</th>
<th>Data length</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>F&quot;50&quot; (= Determine program type)</td>
<td></td>
<td>F'word</td>
</tr>
<tr>
<td>04</td>
<td>Reason code</td>
<td>C_REASON_CODE</td>
<td>F'word</td>
</tr>
<tr>
<td>08</td>
<td>Address of system EIB</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>0C</td>
<td>240-byte work area</td>
<td>C_WORKAREA</td>
<td>240</td>
</tr>
<tr>
<td>10</td>
<td>C/370 partition token storage area</td>
<td>CSACELPT</td>
<td>8</td>
</tr>
<tr>
<td>14</td>
<td>C/370 thread token storage area</td>
<td>EIOCTHRT</td>
<td>8</td>
</tr>
<tr>
<td>18</td>
<td>Program load point</td>
<td>TCAPCLA</td>
<td>4</td>
</tr>
<tr>
<td>1C</td>
<td>Program entry point</td>
<td>TCAPCRS</td>
<td>4</td>
</tr>
<tr>
<td>20</td>
<td>Program type (receiver)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>Run-unit work area length (receiver)</td>
<td>PPTCISA</td>
<td>F'word</td>
</tr>
</tbody>
</table>

### Table 27. CICS-C/370 THREAD_INITIALIZATION parameter list

<table>
<thead>
<tr>
<th>Offset</th>
<th>Data addressed by the pointer</th>
<th>CICS field name</th>
<th>Data length</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>F&quot;20&quot; (= Thread initialization)</td>
<td></td>
<td>F'word</td>
</tr>
<tr>
<td>04</td>
<td>Reason code</td>
<td>C_REASON_CODE</td>
<td>F'word</td>
</tr>
<tr>
<td>08</td>
<td>Address of system EIB</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>0C</td>
<td>240-byte work area</td>
<td>C_WORKAREA</td>
<td>240</td>
</tr>
<tr>
<td>10</td>
<td>C/370 partition token storage area</td>
<td>CSACELPT</td>
<td>8</td>
</tr>
<tr>
<td>14</td>
<td>C/370 thread token storage area (receiver)</td>
<td>EIOCTHRT</td>
<td>8</td>
</tr>
<tr>
<td>18</td>
<td>Preallocated thread work area</td>
<td></td>
<td>F'word</td>
</tr>
</tbody>
</table>

### Table 28. CICS-C/370 RUNUNIT_INITIALIZATION parameter list

<table>
<thead>
<tr>
<th>Offset</th>
<th>Data addressed by the pointer</th>
<th>CICS field name</th>
<th>Data length</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>F&quot;32&quot; (= Run-unit initialization)</td>
<td></td>
<td>F'word</td>
</tr>
<tr>
<td>04</td>
<td>Reason code</td>
<td>C_REASON_CODE</td>
<td>F'word</td>
</tr>
<tr>
<td>08</td>
<td>Address of system EIB</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>0C</td>
<td>240-byte work area</td>
<td>C_WORKAREA</td>
<td>240</td>
</tr>
<tr>
<td>10</td>
<td>C/370 partition token storage area</td>
<td>CSACELPT</td>
<td>8</td>
</tr>
<tr>
<td>14</td>
<td>C/370 thread token storage area</td>
<td>EIOCTHRT</td>
<td>8</td>
</tr>
<tr>
<td>18</td>
<td>C/370 run-unit token storage area (receiver)</td>
<td>EIORUNTK</td>
<td>8</td>
</tr>
<tr>
<td>1C</td>
<td>Preallocated run-unit work area</td>
<td>EIORUSTG</td>
<td>F'word</td>
</tr>
<tr>
<td>20</td>
<td>Run-unit entry address</td>
<td>EIOARG4</td>
<td>4</td>
</tr>
<tr>
<td>24</td>
<td>Application program argument list</td>
<td>EISARG1</td>
<td>F'word</td>
</tr>
<tr>
<td>28</td>
<td>Input/output queue details</td>
<td>C_IOINFO (see page 244)</td>
<td>18</td>
</tr>
</tbody>
</table>
### Table 29. CICS-C/370 WORKING_STORAGE_LOCATE parameter list

<table>
<thead>
<tr>
<th>Offset</th>
<th>Data addressed by the pointer</th>
<th>CICS field name</th>
<th>Data length</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>F&quot;60&quot; (= Working storage locate)</td>
<td>C_REASON_CODE</td>
<td>F'word</td>
</tr>
<tr>
<td>04</td>
<td>Reason code</td>
<td>C_REASON_CODE</td>
<td>F'word</td>
</tr>
<tr>
<td>08</td>
<td>Address of system EIB</td>
<td>C_WORKAREA</td>
<td>4</td>
</tr>
<tr>
<td>0C</td>
<td>240-byte work area</td>
<td>CSACELPT</td>
<td>8</td>
</tr>
<tr>
<td>10</td>
<td>C/370 partition token storage area</td>
<td>EIOCTHRT</td>
<td>8</td>
</tr>
<tr>
<td>14</td>
<td>C/370 thread token storage area</td>
<td>EISARSA</td>
<td>240</td>
</tr>
<tr>
<td>18</td>
<td>Application RSA address</td>
<td>EISARSA</td>
<td>4</td>
</tr>
<tr>
<td>1C</td>
<td>Working storage address (receiver)</td>
<td>EDFUASTG</td>
<td>4</td>
</tr>
<tr>
<td>20</td>
<td>Working storage length (receiver)</td>
<td>EDFWSLN</td>
<td>F'word</td>
</tr>
</tbody>
</table>

### Table 30. CICS-C/370 RUNUNIT_TERMINATION parameter list

<table>
<thead>
<tr>
<th>Offset</th>
<th>Data addressed by the pointer</th>
<th>CICS field name</th>
<th>Data length</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>F&quot;31&quot; (= Run-unit termination)</td>
<td>C_REASON_CODE</td>
<td>F'word</td>
</tr>
<tr>
<td>04</td>
<td>Reason code</td>
<td>C_REASON_CODE</td>
<td>F'word</td>
</tr>
<tr>
<td>08</td>
<td>Address of system EIB</td>
<td>C_WORKAREA</td>
<td>4</td>
</tr>
<tr>
<td>0C</td>
<td>240-byte work area</td>
<td>CSACELPT</td>
<td>8</td>
</tr>
<tr>
<td>10</td>
<td>C/370 partition token storage area</td>
<td>EIOCTHRT</td>
<td>8</td>
</tr>
<tr>
<td>14</td>
<td>C/370 thread token storage area</td>
<td>EIOCTHRT</td>
<td>8</td>
</tr>
<tr>
<td>18</td>
<td>C/370 run-unit token storage area</td>
<td>EIORUNTK</td>
<td>8</td>
</tr>
<tr>
<td>1C</td>
<td>Termination data</td>
<td>CELINFO</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td>(see page 247)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 31. CICS-C/370 THREAD_TERMINATION parameter list

<table>
<thead>
<tr>
<th>Offset</th>
<th>Data addressed by the pointer</th>
<th>CICS field name</th>
<th>Data length</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>F&quot;21&quot; (= Thread termination)</td>
<td>C_REASON_CODE</td>
<td>F'word</td>
</tr>
<tr>
<td>04</td>
<td>Reason code</td>
<td>C_REASON_CODE</td>
<td>F'word</td>
</tr>
<tr>
<td>08</td>
<td>Address of system EIB</td>
<td>C_WORKAREA</td>
<td>4</td>
</tr>
<tr>
<td>0C</td>
<td>240-byte work area</td>
<td>CSACELPT</td>
<td>8</td>
</tr>
<tr>
<td>10</td>
<td>C/370 partition token storage area</td>
<td>EIOCTHRT</td>
<td>8</td>
</tr>
<tr>
<td>14</td>
<td>C/370 thread token storage area</td>
<td>EIOCTHRT</td>
<td>8</td>
</tr>
</tbody>
</table>
### CICS-C/370 interface

#### Table 32. CICS-C/370 PARTITION_TERMINATION parameter list

<table>
<thead>
<tr>
<th>Offset</th>
<th>Data addressed by the pointer</th>
<th>CICS field name</th>
<th>Data length</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>F&quot;11&quot; (= Partition termination)</td>
<td></td>
<td>F’word</td>
</tr>
<tr>
<td>04</td>
<td>Reason code</td>
<td>C_REASON_CODE</td>
<td>F’word</td>
</tr>
<tr>
<td>08</td>
<td>Address of system EIB</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>0C</td>
<td>240-byte work area</td>
<td>C_WORKAREA</td>
<td>240</td>
</tr>
<tr>
<td>10</td>
<td>C/370 partition token storage area</td>
<td>CSACELPT</td>
<td>8</td>
</tr>
<tr>
<td>14</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Chapter 17. CICS-DB2 Attachment Facility

The CICS-DB2 Attachment facility allows applications programs to access and update data held in DB2 tables managed by the DB2 for OS/390 product. It also allows applications to send operator commands to a DB2 subsystem.

Design overview

The CICS-DB2 Attachment facility allows connection to a DB2 subsystem using the CICS resource manager interface (RMI) also known as the task related user exit interface. The Attachment facility interfaces to DB2 through a series of requests to three components of DB2, each of which processes specific types of requests:

- Subsystem Support Subcomponent (SSSC) for thread and system control requests
- Advanced Database Management Facility (ADMF) for SQL requests
- Instrumentation Facility Component (IFC) for IFI requests

There are no DB2 release dependencies within the attachment facility, it can connect to a DB2 subsystem running any supported level of DB2.

The architecture of the CICS-DB2 interface is shown in Figure 38 on page 266.
CICS Initialization

During CICS Initialization the following modules are invoked:

**CICS-DB2 initialization gate DFHD2IN1**

DFHD2IN1 first receives control from DFHSII1 during CICS initialization by means of a DFHROINM INITIALISE call. When invoked with this function DFHD2IN1 attaches a system task CSSY to run program DFHD2IN2.

DFHD2IN1 is invoked a second time later by DFHSII1 by means of a DFHROINM WAIT_FOR_INITIALIZATION call for which DFHD2IN1 issues a CICS wait to wait for DFHD2IN2 processing to complete.
CICS-DB2 recovery task DFHD2IN2
DFHD2IN2 runs under CICS system task CSSY attached by DFHD2IN1.
DFHD2IN2 links to program DFHD2RP, the CICS-DB2 restart program. On return
from DFHD2RP, DFHD2IN2 posts the ecb waited on by DFHD2IN1 so that CICS
Initialization can continue.

CICS-DB2 restart program DFHD2RP
DFHD2RP runs under system task CSSY during CICS initialization. DFHD2RP
performs the following functions:
• Adds storage manager subpools for the DFHD2ENT, DFHD2TRN and
  DFHD2CSB control blocks.
• Issues lock manager domain ADD_LOCK requests to add the necessary locks
  required by the CICS-DB2 Attachment facility to manage the dynamic chains of
  DFHD2LOT and DFHD2CSB control blocks, plus locks to manipulate the
  DFHD2GLB, DFHD2ENT and DFHD2TRN control blocks.
• Loads CICS-DB2 modules DFHD2CC, DFHD2STR, DFHD2STP and DFHD2TM
• Activates the DFHD2TM gate with the kernel.
• For cold and Initial CICS starts:
  – Purges the Global catalog of DFHD2GLB, DFHD2ENT and DFHD2TRN
    control blocks
• For warm and emergency CICS starts:
  – Installs DFHD2GLB, DFHD2ENT and DFHD2TRN blocks found on the global
    catalog

CICS-DB2 Attachment startup
The CICS-DB2 Attachment facility can be started using one of the following
methods:
• specifying program DFHD2CM0 in PLTP
• specifying SIT parameter DB2CONN=YES
• Issuing the DSNC STRT command
• Issuing the CEMT or EXEC CICS SET DB2CONN CONNECTED command

All of the above ways result in an EXEC CICS SET DB2CONN CONNECTED
command being issued and the CICS-DB2 startup program DFHD2STR getting
control.

CICS-DB2 startup program DFHD2STR
The startup program starts by reading a temporary storage queue to obtain any
parameters passed if a DSNC STRT command has been issued. It also retrieves any
parameters specified via the INITPARM SIT parameter by linking to program
DFHD2INI.

Next DFHD2STR must ensure the necessary DFHD2GLB block is installed. If a
DFHD2GLB is already installed, representing an installed DB2CONN, then it is
checked to make sure interface is currently shut before startup can proceed. If no
DFHD2GLB block exists, then program DFHD2CNV is called to locate and load a
macro RCT, and then convert it to RDO form, installing the necessary control
blocks.

The remainder of DFHD2STR processing is as follows:
• Initialise the DFHD2GLB and set the state to ‘connecting’
• MVS load the DB2 program request handler
CICS-DB2 Attachment Facility

- Attach a CICS system task to run the CICS DB2 service task CEX2
- Issue an MVS Attach for the CICS-DB2 master subtask program DFHD2MSB
- Wait for DFHD2MSB initialization processing to complete
- Enable the CICS-DB2 TRUE DFHD2EX1
- Set the status of the connection to ‘connected’
- Process any indoubts passed from DB2
- Update state in the temporary storage queue to pass back to a DSNC STRT command

CICS-DB2 Attachment shutdown

The CICS-DB2 Attachment facility can be stopped using one of the following methods:
- Issuing the DSNC STOP command
- Issuing the CEMT or EXEC CICS SET DB2CONN NOT CONNECTED command
- Running the CDBQ or CDBF transactions
- Shutting down CICS

All of the above ways result in an EXEC CICS SET DB2CONN NOTCONNECTED command being issued and the CICS-DB2 shutdown program DFHD2STP getting control.

CICS-DB2 Shutdown program DFHD2STP

Processing in DFHD2STP is as follows:
- If CDB2SHUT is set in the dump table, take a system dump (serviceability aid)
- Post CICS-DB2 service task CEX2 to terminate all subtasks then terminate itself.
  Wait for service task to complete.
- Post master subtask DFHD2MSB to terminate. Wait for it to terminate
- Detach master subtask TCB.
- Call DFHD2CC to write out shutdown statistics
- If the CICS-DB2 attachment is to go into ‘standbymode’:
  - Re-initialise the DFHD2GLB, set the state to ‘connecting’
  - Post any tasks who are waiting for shutdown to complete
  - Issues ‘Waiting for DB2 attach’ message
- If the CICS-DB2 attachment is not to go into ‘standbymode’:
  - Disable the CICS-DB2 TRUE DFHD2EX1
  - MVS delete the program request handler
  - Re-initialise the DFHD2GLB, set the state to ‘shut’
  - Discard the control blocks if they originated from a macro RCT conversion
  - Issue the shutdown complete message and post any tasks who are waiting for
  shutdown to complete

CICS-DB2 mainline processing

CICS-DB2 Task Related User Exit (TRUE) DFHD2EX1

Control is passed to the TRUE via the CICS RMI. The TRUE manages the relationship between a CICS task represented by a LOT control block, and a CICS-DB2 subtask represented by a CSB control block. DFHD2EX1 uses parameters set in the DB2CONN and DB2ENTRY definitions to manage use of the CICS DB2 threads, each thread running under a subtask. It is the subtask running program
DFHD2EX3 which issues requests to DB2 on behalf of a CICS task. A wait/post protocol is executed between the CICS task running in the CICS-DB2 TRUE, and the subtask in program DFHD2EX3.

The CICS-DB2 TRUE DFHD2EX1 gets invoked by the RMI for the following events:

- EXEC SQL commands and DB2 IFI commands from application programs
- syncpoint
- end of task
- INQUIRE EXITPROGRAM commands for the DB2 TRUE with the CONNECTST or QUALIFIER keywords (RMI SPI calls)
- EDF - when EDving EXEC SQL commands
- CICS shutdown

**CICS-DB2 Master subtask program DFHD2MSB**

The DFHD2MSB TCB is attached by DFHD2STR during startup of the Attachment facility. It runs as a ‘daughter’ of the main CICS TCB. It is ‘mother’ to all the subtask TCBs which process the DB2 work. The DFHD2MSB TCB is detached by DFHD2STP during CICS-DB2 Attachment shutdown.

The main functions of DFHD2MSB are:

- To issue the initial IDENTIFY to DB2 to connect CICS to DB2
- To find out from DB2 what indoubts it has
- To service resynchronisation requests from CICS to DB2
- To provide a shutdown listening exit to DB2 to listen out for DB2 shutdown
- To attach thread subtasks as required
- To detach thread subtasks as required
- To provide a recovery routine to cleanup if a thread subtask fails

**CICS-DB2 subtask program DFHD2EX3**

A CICS-DB2 subtask TCB is attached by DFHD2MSB when required by DFHD2EX1. It runs as a daughter of the DFHD2MSB TCB and a granddaughter of the main CICS TCB. A CICS-DB2 subtask TCB normally remains active for the lifetime of the CICS Attachment facility and terminates as part of CICS-DB2 Attachment facility shutdown. Exception conditions that cause a subtask TCB to be detached are:

- if the DB2CONN TCBLIMIT parameter is lowered
- if a CICS task is forcepurged whilst its associated subtask is active in DB2
- If a failure occurs during syncpoint processing during the indoubt window requiring the thread to be released.

The DFHD2EX3 program issues requests to DB2 using the DB2 SSSC, ADMF and IFC interfaces communicating via the DB2 program request handler DSNAPRH. In order to process DB2 requests a TCB first has to IDENTIFY to DB2, secondly it has to SIGNON to DB2 to establish authorization ids to DB2. Thirdly a thread has to be created. Once a thread has been created API and syncpoint requests can flow to DB2. Subsequent SIGNON requests can occur for a thread to change authorization ids to DB2 or for the purposes of DB2 cutting accounting records (partial SIGNON) When a thread is nolonger required it is terminated. The TCB remains identified and signed on to DB2 and awaits another request requiring it to create a thread again.
CICS-DB2 Attachment Facility

Each DB2 subtask runs an instance of program DFHD2EX3 and each is represented by a DFHD2CSB control block. A CSB control block is anchored to one of three CSB chains depending on its state (an active thread within a UOW, a thread waiting for work, or an identified, signed on TCB with no thread). The CICS-DB2 TRUE DFHD2EX1 manages the CSB chains.

CICS-DB2 service task program DFHD2EX2
The CICS-DB2 service task program DFHD2EX2 runs as a CICS system task under transaction CEX2. Its mains functions are:

- To wait for DB2 to startup if DB2 is down when connection is attempted if STANDBYMODE=RECONNECT or CONNECT is specified in the DB2CONN.
- To initiate shutdown of the CICS-DB2 Attachment facility if posted to do so by DFHD2MSB.
- To perform the protected thread purge cycle.
- To terminate all subtasks during CICS-DB2 Attachment facility shutdown.

CICS-DB2 PLTPI program DFHD2CM0
Used in PLTPI or as a result of DB2CONN=YES being set in the SIT. It issues an EXEC CICS SET DB2CONN CONNECTED command to start up the CICS DB2 Attachment facility.

CICS-DB2 Command processor DFHD2CM1
DFHD2CM1 processes commands issues via the DSNC command. The following commands are processed:

- DSNC STRT - EXEC CICS SET DB2CONN CONNECTED command issued
- DSNC STOP - EXEC CICS SET DB2CONN NOTCONNECTED command issued
- DSNC MODIFY DEST - EXEC CICS SET DB2CONN MSGQUEUEn command issued
- DSNC MODIFY TRAN - EXEC CICS SET DB2CONN THREADLIMIT or EXEC CICS SET DB2ENTRY THREADLIMIT command issued.
- DSNC DISC - call passed to DFHD2CC to disconnect threads
- DSNC DISP PLAN - call passed to DFHD2CC to display information on threads for a particular DB2 plan
- DSNC DISP TRAN - call passed to DFHD2CC to display information on threads for a transaction.
- DSNC DISP STAT - call passed to DFHD2CC to write out statistics
- DSNC -db2command - DB2 IFI command issued to send operator command to the connected DB2 subsystem.

CICS-DB2 shutdown quiesce program DFHD2CM2
Runs under transaction CDBQ. Issues an EXEC CICS SET DB2CONN NOTCONNECTED WAIT command to shutdown the CICS-DB2 Attachment facility.

CICS-DB2 shutdown force program DFHD2CM3
Runs under transaction CDBF. Issues an EXEC CICS SET DB2CONN NOTCONNECTED FORCE command to shutdown the CICS-DB2 Attachment facility.

CICS-DB2 Table manager DFHD2TM
Handles installs, discards, inquire and set requests for the DFHD2GLB, DFHD2ENT and DFHD2TRN control blocks representing the DB2CONN, DB2ENTRY and DB2TRAN resources. Callers of DFHD2TM are:

- DFHAMD2 - for CEDA install and EXEC CICS CREATE
DFHD2CNV - to install DB2 objects as a result of dynamic conversion from a macro RCT.

DFHD2EX1 - to complete disablement of a DB2ENTRY or to complete Attachment facility shutdown

DFHD2RP - to install objects from the Global Catalog during CICS restart

DFHD2STP - to discard DB2 objects during Attachment shutdown if they originated from a macro RCT.

DFHEIQD2 - for EXEC CICS INQUIRE, SET and DISCARD of DB2 objects

DFHESE - for inquiry during EXEC CICS QUERY SECURITY processing.

CICS DB2 statistics program DFHD2ST
Called by AP domain statistics program DFHAPST to process CICS-DB2 statistics for EXEC CICS COLLECT STATISTICS and EXEC CICS PERFORM STATISTICS commands.

CICS DB2 connection control program DFHD2CC
DFHD2CC processes the following requests:
- Start_db2_attachment - request routed on to DFHD2STR
- Stop_db2_attachment - request routed on to DFHD2STP
- Write_db2_statistics - statistics collected from control blocks and are written out to the terminal, to transient data or to SMF.
- disconnect_threads - CSB control blocks searched and marked so that threads are terminated when they are next released.
- display_plan and display_tran - thread information collected from control blocks and output to the terminal.

CICS DB2 EDF processor DFHD2EDF
Receives control from CICS-DB2 TRUE DFHD2EX1 when the TRUE is invoked for an EDF request. DFHD2EDF uses the RMI provided parameters to format the screen to be output by EDF before and after an EXEC SQL request is issued.

Control blocks

DFHD2SS (CICS-DB2 static storage)
CICS-DB2 static storage (D2SS) is acquired by DFHSIB1 and anchored off field SSZDB2 in the static storage address list DFHSSADS. The static storage is initialized by the CICS-DB2 restart program DFHD2RP. Its lifetime is that of the CICS region. CICS-DB2 static storage holds information such as storage manager, lock manager and directory manager tokens acquired during restart processing before any other CICS-DB2 control blocks are installed.

DFHD2GLB (CICS-DB2 Global block)
The DFHD2GLB block represents an installed DB2CONN definition. It is getmaintained by DFHD2TM when a DB2CONN is installed and freemained by DFHD2TM when a DB2CONN is discarded. It holds CICS-DB2 state data global to the connection and also the state data for pool threads and commands threads. The pool and command sections of the DFHD2GLB are mapped by a common type definition DFHD2RCT which is also used to map the DFHD2ENT control block.

The DFHD2GLB block is anchored off CICS-DB2 static storage in field D2S_DFHD2GLB.
DFHD2ENT (CICS-DB2 DB2ENTRY block)
The DFHD2ENT block represents an installed DB2ENTRY definition. It is getmained by DFHD2TM when a DB2ENTRY is installed and freemained by DFHD2TM when a DB2ENTRY is discarded. It uses a type definition DFHD2RCT in common with the pool and command sections of the DFHD2GLB block to achieve a common layout for all three areas. A DFHD2ENT block is located using a directory manager index that is keyed off the RDO name of the DB2ENTRY.

DFHD2TRN (CICS-DB2 DB2TRAN block)
The DFHD2TRN block represents an installed DB2TRAN definition. It is getmained by DFHD2TM when a DB2TRAN is installed and freemained by DFHD2TM when a DB2TRAN is discarded. A DB2TRAN can be located in two ways. Firstly by a directory manager index keyed off the RDO name of the DB2TRAN. Secondly by a directory manager index keyed off the transaction id associated with the DB2TRAN.

DFHD2CSB (CICS-DB2 subtask block)
The DFHD2CSB block represents a CICS-DB2 subtask running program DFHDEX3. A DFHD2CSB is getmained by DFHD2EX1 prior to the subtask being attached. It is passed to the subtask DFHD2EX3 as an attach parm. A DFHD2CSB is freemained by DFHD2EX1 after the DFHD2EX3 program has returned to MVS. A DFHD2EX3 block is anchored off one of several CSB chains from a DB2ENTRY or the DFHD2GLB depending on the state of the TCB and the DB2 thread.

DFHD2GWA (CICS-DB2 global work area)
The DFHD2GWA block is the global work area of the CICS-DB2 task related user exit (TRUE) DFHD2EX1. It is getmained when the TRUE is enabled, and freemained when the TRUE is disabled. The D2GWA holds a chain of LOT control blocks representing the tasks currently using the CICS-DB2 interface.

DFHD2LOT (CICS-DB2 life of task block)
The DFHD2LOT block is the task local work area of the CICS-DB2 task related user exit (TRUE) DFHD2EX1. It is getmained by DFHERM when a task first calls the CICS-DB2 TRUE. It is freemained by DFHERM at end of task. Its address is passed to DFHD2EX1 by DFHERM in parameter UEPTAA in the DFHUEPAR RMI parameter list.

The DFHD2LOT holds CICS-DB2 state information for a CICS task using the CICS-DB2 interface.

### Modules

<table>
<thead>
<tr>
<th>Module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHD2CC</td>
<td>CICS-DB2 connection control program</td>
</tr>
<tr>
<td>DFHD2CM0</td>
<td>CICS-DB2 PLTPI startup program</td>
</tr>
<tr>
<td>DFHD2CM1</td>
<td>CICS-DB2 command processor</td>
</tr>
<tr>
<td>DFHD2CM2</td>
<td>CICS-DB2 quiesce shutdown program</td>
</tr>
<tr>
<td>DFHD2CM3</td>
<td>CICS-DB2 force shutdown program</td>
</tr>
<tr>
<td>DFHD2EDF</td>
<td>CICS-DB2 EDF processor</td>
</tr>
<tr>
<td>DFHD2EX1</td>
<td>CICS-DB2 task related user exit (TRUE)</td>
</tr>
<tr>
<td>DFHD2EX2</td>
<td>CICS-DB2 service task program</td>
</tr>
<tr>
<td>DFHD2EX3</td>
<td>CICS-DB2 subtask program</td>
</tr>
</tbody>
</table>
Module Description

<table>
<thead>
<tr>
<th>Module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHD2INI</td>
<td>CICS-DB2 Initparm processor</td>
</tr>
<tr>
<td>DFHD2IN1</td>
<td>CICS-DB2 initialization gate</td>
</tr>
<tr>
<td>DFHD2IN2</td>
<td>CICS-DB2 recovery task</td>
</tr>
<tr>
<td>DFHD2MSB</td>
<td>CICS-DB2 master subtask program</td>
</tr>
<tr>
<td>DFHD2RP</td>
<td>CICS-DB2 restart program</td>
</tr>
<tr>
<td>DFHD2STP</td>
<td>CICS-DB2 shutdown program</td>
</tr>
<tr>
<td>DFHD2STR</td>
<td>CICS-DB2 startup program</td>
</tr>
<tr>
<td>DFHD2ST</td>
<td>CICS-DB2 statistics program</td>
</tr>
<tr>
<td>DFHD2TM</td>
<td>CICS-DB2 table manager</td>
</tr>
<tr>
<td>DSNCUEXT</td>
<td>CICS-DB2 sample dynamic plan exit</td>
</tr>
</tbody>
</table>

Exits

There are no Global user exits provided by the CICS DB2 Interface.

The CICS DB2 interface does however provide a dynamic plan ‘exit’ in the form of a user replaceable module. A sample default exit is provided called DSNCUEXT. A dynamic plan exit allows the name of the plan to chosen dynamically at execution time. For further information about dynamic plan exits see the CICS DB2 Guide.

Trace

The CICS-DB2 Attachment facility outputs trace entries in the range AP 3100 to AP 33FF. Trace output from the CICS-DB2 TRUE DFHD2EX1 and GTF trace from the CICS-DB2 subtask is controlled by the RI (RMI) trace flag. Trace from the rest of the attachment and other CICS-DB2 modules is controlled by the FC (File Control) trace flag.

Statistics

A limited set of CICS-DB2 statistics can be obtained by issuing the DSNC DISP STAT command, which will output the statistics to a CICS terminal. The same format of statistics is output to a nominated transient data queue when the CICS-DB2 Attachment facility is shut down. For more information see the CICS DB2 Guide.

A more comprehensive set of CICS-DB2 statistics can be obtained by issuing an EXEC CICS PERFORM STATISTICS RECORD command with the DB2 keyword, or by issuing the EXEC CICS COLLECT STATISTICS command with the DB2CONN or DB2ENTRY keywords. CICS-DB2 Global statistics are mapped by DSECT DFHD2GDS. CICS-DB2 resource statistics are mapped by DSECT DFHD2RDS. For more information see the CICS Performance Guide.
Chapter 18. Command interpreter

The command interpreter demonstrates to the application programmer the syntax of CICS commands and the effects of their execution. It can also be used to perform simple one-off tasks whose nature does not justify the writing of a permanent application.

Design overview

The command interpreter is invoked by the CECI transaction and is an interactive, display-oriented tool that checks the syntax of CICS commands and executes them. Another transaction, CECS, performs only syntax checking.

The user enters a command that is analyzed in the same way as it would be by the command translator, which processes it as if it were part of an application program. The results of this analysis, including any messages, an indication of defaults assumed, and the entire syntax of the command, are then displayed.

When the command is syntactically valid, the user can request its execution. The interpreter calls DFHEIP, passing a parameter list precisely as would be passed during the execution of a program that contained the command.

The interpreter does all this using the same command-language tables as are used by the command translator. These tables contain data that define the syntax of CICS commands and the contents of the parameter lists required by DFHEIP to execute them.

Modules

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHECIP</td>
<td>Invoked by CECI. Checks that the terminal is suitable. Obtains and initializes working storage. Loads the language tables. Links to DFHECID</td>
</tr>
<tr>
<td>DFHECSP</td>
<td>Same as DFHECIP, but invoked by CECS</td>
</tr>
<tr>
<td>DFHECID</td>
<td>Receives data from the terminal and sends back a display. Analyzes commands. Constructs parameter lists for DFHEIP, which it calls. Deals with PF keys</td>
</tr>
<tr>
<td>DFHEITAB</td>
<td>Command-language table (application programmer commands)</td>
</tr>
<tr>
<td>DFHEITBS</td>
<td>Command-language table (system programmer commands).</td>
</tr>
</tbody>
</table>

Exits

No global user exit points are provided for this function.

Trace

No trace points are provided for this function.
Chapter 19. CSD utility program (DFHCSDUP)

The CSD utility program, DFHCSDUP, provides offline services for you to list and modify the resource definitions in the CICS system definition (CSD) file. DFHCSDUP can be invoked as a batch program, or from a user-written program running either in batch mode or under TSO. The second method provides a more flexible interface to the utility, allowing for the specification of up to five user exit routines to be called at various points during DFHCSDUP processing.


The following commands can be used with DFHCSDUP:

**ADD**
**ALTER**
**APPEND**
**COPY**
**DEFINE**
**DELETE**
**EXTRACT**
**INITIALIZE**
**LIST**
**MIGRATE**
**REMOVE**
**SERVICE**
**UPGRADE**
**VERIFY**

These commands are described in the CICS Resource Definition Guide and the CICS Operations and Utilities Guide.

Design overview

When DFHCSDUP is invoked, control passes to the utility command processor (DFHCUCP), which validates commands and invokes the appropriate routine to execute the requested function. Unless DFHCSDUP has been invoked from a user program specifying a get-command exit, DFHCUCP takes a command from the input data set, using DFHCUCB to obtain the command and DFHCUCAB to analyze and parameterize it. When supplied, the get-command exit is invoked from the point during DFHCUCB’s processing where commands would otherwise be read from SYSIN (or an alternatively named input data set when DFHCSDUP is invoked from a user program).

Some syntax errors are diagnosed and reported by DFHCUCAB, and further contextual validation takes place in DFHCUCV. Valid commands are then passed to the relevant service routine for execution; for example, a MIGRATE command is handled by DFHCUMIG. If command execution is successful, the next command is processed.

All commands are validated, but the execution of commands from the input data set stops when an incorrect command is encountered, and execution of subsequent commands is also suppressed if an error of severity 8 or higher occurs when the command is executed. When commands are supplied by a get-command exit,
CSD utility program (DFHCSDUP)

however, DFHCSDUP attempts to execute all commands, even if an error is
detected in the command syntax or during processing (unless the error is serious
enough to warrant an ABEND).

If errors occur while processing commands, error messages in the DFH51xx,
DFH52xx, DFH55xx, and DFH56xx series are written to SYSPRINT (or an
alternatively named output data set when DFHCSDUP is invoked from a user
program).

An ESTAE environment is established by DFHCUCP shortly after the start of
DFHCSDUP processing. If an operating system abend subsequently occurs, control
passes to the ESTAE exit routine, which then returns to MVS requesting a dump
and scheduling a retry routine to get control. This retry routine attempts cleanup
processing before returning to the caller of DFHCSDUP with a return code of ‘16’.

To protect the integrity of the CSD, DFHCUCP issues a STAX macro to defer the
handling of any attention interrupts that may occur in a TSO environment until all
processing associated with the current command has been completed.

DFHCSDUP uses batch versions of RDO routines from the parameter utility
program (DFHPUP) and the CSD management program (DFHDMP) to read, write,
and update resource definitions on the CSD file. All CSD control functions use the
batch environment adapter (DFHDMPBA), which performs environment-
dependent VSAM operations on the CSD file. DFHDMPBA also processes all
interactions with operating system services.

Modules

DFHCSDUP is link-edited from a number of object modules, including batch
versions of routines from DFHPUP and DFHDMP.

Exits

When invoked as a conventional batch program, DFHCSDUP supports only one
user exit: the EXTRACT exit, which is invoked at various stages during the
processing of an EXTRACT command. The name of the user-written program to
get control must be specified by the USERPROGRAM keyword of the EXTRACT
command. Details of selected CSD objects are passed to the user exit program so
that users can analyze the contents of their CSD in any way they may choose.

When invoked from a user program, DFHCSDUP supports the following five user
exits, the addresses of which can be specified in the EXITS parameter of
DFHCSDUP’s entry linkage:
1. Initialization exit—invoked by DFHCUCP
2. Termination exit—invoked by DFHCUCP
3. EXTRACT exit—invoked by DFHCULIS
4. Get-command exit—invoked by DFHCUCB
5. Put-message exit—invoked by DFHBEP.

Note: A user exit routine specified by the USERPROGRAM keyword of an
EXTRACT command is used in preference to any EXTRACT exit routine
specified on the entry linkage.

For further information about these user exits, see the CICS Customization Guide.
Trace

Trace points are not applicable to offline utilities.

Statistics

The following statistics are maintained by DFHCSDUP, and are written, when appropriate, to SYSPRINT (or alternatively named output data set):

- **CMDSEXOK** Commands executed OK
- **CMDSINER** Commands in error
- **CMDSNOTX** Commands not executed
- **CMDSWARN** Commands with warning messages.

All the above statistics are kept in DFHCUCP’s static storage and are always output at the end of processing.

All the following statistics are kept in DFHCUMIG’s static storage and the appropriate statistics are also output to SYSPRINT (or its replacement). For example, if a user migrates an FCT, only TOTFILE and TOTLSRP are output.

- **TOTCONS** Total connections migrated
- **TOTFILE** Total files migrated
- **TOTLSRP** Total LSR pools migrated
- **TOTMAPS** Total map sets migrated
- **TOTPGMS** Total programs migrated
- **TOTPRFG** Total profiles generated
- **TOTPRFM** Total profiles migrated
- **TOTPSTS** Total partition sets migrated
- **TOTSESS** Total sessions migrated
- **TOTTRAS** Total transactions migrated
- **TOTTRMS** Total terminals migrated
- **TOTTYPS** Total typeterms migrated
CSD utility program (DFHCSUDP)
Chapter 20. Database control (DBCTL)

An overall description of DL/I database support is given in "Chapter 26. DL/I database support" on page 343. This section gives information that is specific to database control (DBCTL).

Design overview

The CICS support that enables connection to DBCTL, via the database resource adapter (DRA), is based on the CICS resource manager interface (RMI), also known as the task-related user exit interface. However, because it is necessary to provide compatibility with the existing CICS-DL/I implementation (in terms of link-edit stubs, API return codes, and so on), a limited amount of support within CICS itself is provided, but there are no DBCTL release dependencies within the CICS modules.

The main components of the CICS-DBCTL interface are shown in Figure 39:

- **CICS address space**
- **IMS/ESA address space**
- **DRA control exit**

Figure 39. The major components of the CICS-DBCTL interface

- The connection process (CICS-DBCTL)
  - **CICS-DBCTL connection and disconnection programs**
    These programs are used for establishing and terminating the connection with the DRA.
  - **CICS-DBCTL control program**
    This program is responsible for resolving in-doubt units of work after a CICS or DBCTL failure. It also outputs messages when DBCTL notifies CICS of a change in the status of the CICS-DBCTL interface.
    When the CICS disconnects from DBCTL, the control program is responsible for invoking the disable program which performs cleanup.

- **DRA control exit**
  This exit is invoked by the DRA, when connection has been established with the DBCTL address space, to initiate the resynchronization process,
Database control (DBCTL)

that is, to initiate the resolution of in-doubt units of work. It is also
invoked to handle cases where connection to DBCTL cannot be achieved
or when the connection has failed.

DBCTL user-replaceable program
This program is invoked whenever CICS successfully connects to DBCTL
and whenever CICS disconnects from DBCTL.

Disable program
This program is invoked when CICS disconnects from DBCTL.

• The DBCTL call processor program
The function of this program is to issue an RMI call to DBCTL and to maintain
compatibility with the existing CICS-DL/I interface in areas such as application
program return codes, and so on.

• The interface layer

The adapter
The adapter’s primary responsibility is interfacing the RMI and DRA
parameter lists. Other responsibilities include the issuing of DRA
initialization and termination calls, when invoked by the CICS
connection and disconnection programs, and the management of CICS
tasks, in order to effect an orderly shutdown of the CICS-DBCTL
interface.

DRA suspend and resume exits
These exits are invoked by the DRA in order to suspend and resume a
CICS task while a DL/I call is processed by DBCTL.

Adapter exits
There are four exits for use by the adapter:
– The statistics exit
– The token exit
– The monitoring exit
– The status exit.

Details of these components are described in the following sections.

Note: CICS documentation uses the term “connecting and disconnecting from
DBCTL”. The DRA documentation refers to “initializing and terminating the
CICS-DBCTL interface”. In general, these two terms are synonymous.

The connection process

Connection and disconnection programs
In order to initialize, terminate, and inquire on the status of the interface, a set of
four programs is available:
1. Menu program
2. Connection program
3. Disconnection program
4. Inquiry program.

Menu program (DFHDBME): This permits a terminal user to display a menu,
which offers the option of connecting and disconnecting from DBCTL.

The menu program passes control to either the connection or the disconnection
program, as appropriate, using the COMMAREA to pass any overrides and
parameters.
In the case of connection, it offers the ability to supply the suffix of the DRA startup parameter table and the name of the DBCTL region. The DRA startup parameter table contains various parameters, mostly relating to the initialization of the CICS-DBCTL interface, including the name of the DBCTL region and the minimum and maximum number of CICS-DBCTL threads. It also contains the length of time in seconds that the DRA waits after an unsuccessful attempt to connect to DBCTL, before attempting to connect again.

For disconnection, it offers the ability to specify whether an orderly or immediate disconnection from DBCTL is required.

The menu program is intended for use by CICS operators or network controllers, that is, users with special privileges.

BMS maps are used for both the menu and the inquiry programs. It should be noted that the bottom half of the menu screen includes all the items which appear on the inquiry screen, and the values are displayed on entry to the menu program, if they are known. The DRA startup table suffix is not included on the inquiry screen because the DRA startup table contains the application group name which is used for security checking.

After a connection request has been issued, it is possible to issue a disconnection request (orderly or immediate) from the menu program while the connection process is still in progress. After an orderly disconnection request has been issued, it is also possible to issue an immediate disconnection request while the orderly disconnection process is in progress. This has the effect of upgrading the orderly disconnection to an immediate disconnection.

**Connection program (DFHDBCON):** This program invokes the adapter requesting connection to DBCTL.

This program can be invoked either from the menu program or from the CICS PLT. It issues an ATTACH request of the CICS control program that later carries out resynchronization of in-doubt units of work with DBCTL. The control program then issues a WAIT request.

The connection program continues by loading, activating (using the EXEC CICS ENABLE command), and then calling the adapter (using a DFHRMCAL request). A set of parameters is passed to the adapter which includes:

- The CICS applid
- The DRA startup parameter table suffix (optional)
- The DBCTL ID (optional)
- A set of exit addresses.

As a result of the DFHRMCAL request issued from the connection program, the adapter loads the DRA startup/router module from the CICS STEPLIB library and passes control to it, supplying it with various parameters including the CICS applid, DRA startup parameter table suffix, and DBCTL ID. The DRA startup/router module loads the DRA startup table. It then initiates the processes required to establish the DRA and then returns control to the adapter which, in turn, returns control to the connection program which then terminates. Until this point is reached, any DBCTL requests issued from CICS tasks are rejected by the CICS RMI stub (the DBCTL call processor).
The DRA startup/router module is responsible for establishing the DRA environment, using the parameters specified in the DRA startup table in the CICS STEPLIB library, overridden by any parameters passed to it.

The DRA establishes contact with the DBCTL address space and then invokes the control exit to initiate the resynchronization process.

**Disconnection program (DFHDBDSC):** This program invokes the adapter requesting disconnection from DBCTL.

The disconnection program is used to terminate the DRA environment. Two types of disconnection are available:

**Orderly disconnection**

All existing CICS tasks using DBCTL are allowed to run to completion.

**Immediate disconnection**

Existing DL/I requests are allowed to complete but no further DL/I requests are accepted.

In both cases a DBCTL U113 abend is avoided. (DBCTL can issue a U113 abend if CICS terminates while there is an active DL/I thread running on its behalf in DBCTL. The thread remains active for the duration of the PSB schedule, but DBCTL would issue a U113 abend if the thread is doing something for the CICS task.)

The disconnection program calls the adapter, using DFHRMCAL, supplying a parameter to indicate the type of termination required.

In the case of immediate disconnection, the adapter issues a DRA TERM call and returns to the disconnection program only when all existing DL/I threads have completed. In the case of orderly disconnection, the adapter assumes responsibility for managing CICS tasks, that is, it continues to accept requests for current tasks using DBCTL until they terminate, but does not allow new CICS tasks to use DBCTL. When the adapter detects that the count of permitted tasks has reached zero, it issues a DRA TERM call.

The disconnection program finally posts the control program to notify it of the fact that the CICS-DBCTL interface has been terminated. The control program then terminates after starting the disable program. The disable program issues a DISABLE command for the adapter, and performs cleanup.

It should be noted that the terminal used to invoke the disconnection program is released after the input to the menu screen has been validated, enabling the terminal operator to use other programs. Any further messages from the disconnection process are generated centrally.

**Inquiry program (DFHDBIQ):** This program enables the user to inquire on the status of the interface. It is intended for a wider audience than the menu program; for example, application programmers.

**Control program (DFHDBCT)**

The control program is invoked in the following circumstances:

- When the control exit is invoked by the adapter on behalf of the DRA
- When a CEMT FORCEPURGE command is issued for a CICS task executing in DBCTL
When the disconnection program has received a response from the adapter as a result of a CICS-DBCTL interface termination request.

Its function in all cases is to issue messages. It then issues a WAIT after every invocation. Also, it has some special functions in three cases:

1. When contact has been made with DBCTL and resynchronization of in-doubts is required.
   In this case, the control program issues the command:
   ```
   EXEC CICS RESYNC ENTRYPNAME(adapter)  
   IDLIST(DBCTL's in-doubts) ...
   ```
   This causes CICS to create tasks for each in-doubt unit of work. Each task performs resynchronization and then informs the adapter via the CICS syncpoint manager as to whether the task has committed or backed out. The adapter then notifies the DRA on a task basis.

   The following is a list of the possible calls to the adapter from the CICS syncpoint manager:
   • Prepare to commit
   • Commit unconditionally
   • Backout
   • Unit of recovery is lost to CICS cold start
   • DBCTL should not be in-doubt about this unit of recovery.

   Notes:
   1. These items can be issued as a result of a RESYNC request.
   2. These items can be issued as a result of a RESYNC request only.

2. When /CHECKPOINT FREEZE has been requested.
   In this case, the control program invokes the disconnection program requesting an orderly disconnection from DBCTL. Generally, an orderly disconnection from DBCTL allows CICS tasks already using DBCTL to continue until task termination. However, when a /CHECKPOINT FREEZE has been requested, DBCTL prevents any PSB schedules from taking place. Thus, in this case, some tasks might be terminated before end of task is reached with a 'DBCTL not available' return code, if they issue a subsequent PSB schedule request.

3. When the disconnection program invokes the control program.
   In this case, the control program starts the disable program.

**DRA control exit (DFHDBCTX)**
The control exit is invoked in the DRA environment in the following circumstances:
• When contact has been established with the DBCTL address space, in order to initiate resynchronization.

   The control exit is invoked in the DRA environment whenever contact has been established with DBCTL, whether invoked by the user or due to the DRA automatically reestablishing contact after a DBCTL failure. The control exit receives an input parameter list that includes the DBCTL ID, DBCTL’s list of in-doubt units of work, and the DBCTL RSE name. The control exit posts the control program, which actually performs the resynchronization.
• When the MVS subsystem interface (SSI) rejects the IDENTIFY request to DBCTL, thereby causing the IDENTIFY to fail.
Database control (DBCTL)

This could occur if the DRA was trying to issue an IDENTIFY request to a DBCTL subsystem that was not running. In this case the control exit sets a response code of '0'. The first time in a connection attempt that the DRA receives a '0' response after an MVS SSI failure, the DRA outputs message DFS690A inviting the operator to reply WAIT or CANCEL. On subsequent failures when a response code of '0' is returned, the DRA waits for the length of time specified in the DRA startup table before attempting the IDENTIFY request again.

- When DBCTL rejects the IDENTIFY request to DBCTL; for example, incorrect application group name (AGN) supplied.
  In this case, the control exit asks the DRA to terminate.

- When the operator replies CANCEL to the DFS690A message during DRA initialization, because contact cannot be established with DBCTL.
  In this case, the control exit notifies the DRA to terminate immediately.

- When DBCTL abnormally terminates.
  In this case, the control exit invokes the control program and then it asks the DRA to issue an IDENTIFY request to DBCTL.

- When the DRA abnormally terminates.
  In this case, it is not possible to access DBCTL from the same CICS session without initializing the CICS-DBCTL interface using the menu program.

- When a /CHECKPOINT FREEZE request has been issued to DBCTL.
  Note that /CHECKPOINT FREEZE is the command used to close down a DBCTL subsystem. In this case the control exit invokes the control program which, in turn, invokes the disconnection program requesting an orderly disconnection from DBCTL. The control exit notifies the DRA to wait for a termination request.

DBCTL user-replaceable program (DFHDBUEX)

The DBCTL user-replaceable program, DFHDBUEX, is invoked whenever CICS successfully connects or disconnects from DBCTL. It provides the opportunity for the customer to supply code to enable and disable CICS-DBCTL transactions at these times.

The program runs as a CICS application and can thus issue EXEC CICS requests. The program is invoked with a CICS COMMAREA containing the following parameters:

- Request type: CONNECT | DISCONNECT
- Reason for disconnection: MENU DISCONNECTION | /CHECKPOINT FREEZE | DRA FAILURE | DBCTL FAILURE
- DRA startup table suffix
- DBCTL ID.

See the CICS Customization Guide for information about the DFHDBUEX program.

Disable program (DFHDBDI)

The disable program, DFHDBDI, is invoked when CICS disconnects from DBCTL. It performs cleanup, which includes disabling the adapter.

The DBCTL call processor program (DFHDLIDP)

Among the functions of the DBCTL call processor program, DFHDLIDP, are:

Issuing DFHRMCAL requests to the adapter: DL/1 requests issued from application programs that have been routed to this module are passed on to the
adapter. The DBCTL call processor constructs a register 1 parameter list that includes the DL/I parameter list and a thread token. It then issues a DFHHRMCAL request.

It is the responsibility of this module to generate the thread token required by the DRA.

**Maintaining return code compatibility:** If any calls are made to the RMI before the first part of the connection process has completed, that is, before the DFHDBCON program has received a “successful” response code from the DRA via the adapter, error return codes are set in the task control area (TCA) to indicate that DBCTL is unavailable. These codes are put in the user interface block (UIB) by the DL/I call router program, DFHDLI.

Similarly, the DBCTL call processor informs application programs when DBCTL is no longer available; for example, after a DBCTL abend.

Another function of the call processor is to set up the TCA fields, TCADLRC and TCADLTR, with response and reason codes respectively for the call. This ensures that the application program continues to receive responses indicating normal response, NOTOPEN, and INVREQ conditions, with the appropriate response and reason codes in the corresponding UIB fields, UIBFCTR and UIBDLTR, after NOTOPEN and INVREQ conditions have been raised.

**Initiating PC abends:** If an ‘unsuccessful’ return code is passed back to CICS as a result of a DBCTL request, indicating that the CICS thread must be abended, the DBCTL call processor issues a PC ABEND, which invokes syncpoint processing to back out changes made to recoverable resources. Various abend codes can be issued. Note that, in the case of a deadlock abend (abend code ADCD) it may be possible to restart the program.

Exception trace entries are output in the case of transaction abends.

**Writing CICS messages:** For any thread abend in DBCTL, a CICS message is written indicating the abend code passed back to CICS in the field PAPLRETC. Similarly, for any scheduling failures, where the application program receives the UIBR CODE field (UIBFCTR and UIBDLTR fields combined) set to X’0805’, the scheduling failure subcode is contained in a CICS message.

### The interface layer

**Adapter (DFHDBAT)**

Control is passed to the adapter via the CICS RMI. It is the responsibility of the adapter to construct the DRA INIT, DRA TERM, and DRA THREAD parameter lists from the RMI parameter list passed to it. It must also transform the DRA parameter list passed back after a DL/I call to the format expected by CICS.

Part of the DRA parameter list requires two tokens to be generated by CICS:
1. A thread token
2. A recovery token.

The thread token is generated by the DBCTL call processor, and enables a CICS unit of work to be related to a DBCTL unit of work. It is used by the asynchronous RESUME exit to identify the CICS thread to be resumed after a DL/I call.
The 16-byte recovery token is constructed by concatenating an 8-byte unique CICS subsystem name (the CICS applid) with the 8-byte CICS RMI recovery token (also known as the unit of work ID).

A further responsibility of the adapter is to manage CICS tasks when an orderly termination of the CICS-DRA interface has been requested by means of the CICS termination program. In this case, it continues to accept DL/I requests from CICS tasks currently using DBCTL, but does not allow new CICS tasks to use DBCTL. When the adapter detects that the count of current tasks has reached zero, it issues a DRA TERM call to shut down the interface.

Table 33 summarizes the types of invocations of the adapter code from CICS, and how the adapter reacts to the individual invocation.

Table 34 summarizes the types of invocations of the adapter code from the DRA, and how the adapter reacts to each individual invocation.

Table 35 on page 289 summarizes the cases when the adapter invokes the adapter exits.

Table 33. CICS-adapter request summary

<table>
<thead>
<tr>
<th>Invocation</th>
<th>Invoker</th>
<th>Adapter action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initialize</td>
<td>Connection program</td>
<td>Issues DRA INIT</td>
</tr>
<tr>
<td>Terminate-Orderly</td>
<td>Disconnection program</td>
<td>Issues DRA TERM after waiting for CICS-DBCTL tasks to quiesce</td>
</tr>
<tr>
<td>Terminate-Fast</td>
<td>Disconnection program</td>
<td>Issues DRA TERM</td>
</tr>
<tr>
<td>PSB Schedule</td>
<td>DBCTL call processor</td>
<td>Issues THREAD SCHED</td>
</tr>
<tr>
<td>DL/I request</td>
<td>DBCTL call processor</td>
<td>Issues THREAD DLI</td>
</tr>
<tr>
<td>Prepare</td>
<td>CICS syncpoint manager</td>
<td>Issues THREAD PREP</td>
</tr>
<tr>
<td>Commit</td>
<td>CICS syncpoint manager</td>
<td>Issues THREAD COMTERM</td>
</tr>
<tr>
<td>Abort</td>
<td>CICS syncpoint manager</td>
<td>Issues THREAD ABTERM</td>
</tr>
<tr>
<td>Lost To CICS cold start</td>
<td>CICS syncpoint manager</td>
<td>Issues COLD request</td>
</tr>
<tr>
<td>DBCTL should not be in doubt</td>
<td>CICS syncpoint manager</td>
<td>Issues UNKNOWN request</td>
</tr>
<tr>
<td>Task is terminating</td>
<td>CICS task manager</td>
<td>Issues TERMTHRD</td>
</tr>
<tr>
<td>Force Purge Task</td>
<td>Control program</td>
<td>Issues PURGE THREAD</td>
</tr>
<tr>
<td>Orderly CICS Term</td>
<td>CICS termination</td>
<td>Issues DRA TERM after waiting for CICS-DBCTL tasks to quiesce</td>
</tr>
<tr>
<td>Immediate CICS Term</td>
<td>CICS termination</td>
<td>Issues DRA TERM</td>
</tr>
<tr>
<td>CICS is abending</td>
<td>CICS termination</td>
<td>Issues DRA TERM</td>
</tr>
<tr>
<td>CICS has been canceled</td>
<td>CICS termination</td>
<td>Returns to CICS</td>
</tr>
</tbody>
</table>

Table 34. DRA-adapter request summary

<table>
<thead>
<tr>
<th>Invocation from the DRA</th>
<th>Adapter action</th>
</tr>
</thead>
<tbody>
<tr>
<td>CICS-DBCTL connection is complete</td>
<td>Invoke the control exit</td>
</tr>
<tr>
<td>MVS SSI has rejected the IDENTIFY request to DBCTL</td>
<td>Invoke the control exit</td>
</tr>
</tbody>
</table>
Table 34. DRA-adapter request summary (continued)

<table>
<thead>
<tr>
<th>Invocation from the DRA</th>
<th>Adapter action</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBCTL has rejected the IDENTIFY request</td>
<td>Invoke the control exit</td>
</tr>
<tr>
<td>Operator has replied CANCEL to message DFS690A</td>
<td>Invoke the control exit</td>
</tr>
<tr>
<td>DBCTL has terminated abnormally</td>
<td>Invoke the control exit</td>
</tr>
<tr>
<td>DRA has terminated abnormally</td>
<td>Invoke the control exit</td>
</tr>
<tr>
<td>/CHECKPOINT FREEZE has been issued</td>
<td>Invoke the control exit</td>
</tr>
<tr>
<td>PSB schedule, DL/I, syncpoint, thread termination, thread purge, or interface termination request is to be suspended</td>
<td>Invoke the suspend exit</td>
</tr>
<tr>
<td>PSB schedule, DL/I, syncpoint, thread termination, thread purge, or interface termination request is to be resumed</td>
<td>Invoke the resume exit</td>
</tr>
</tbody>
</table>

Table 35. Adapter exit summary

<table>
<thead>
<tr>
<th>Circumstances</th>
<th>Adapter action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Successful completion of THREAD SCHED request</td>
<td>Invoke the monitoring exit</td>
</tr>
<tr>
<td>Completion of THREAD COMTERM or THREAD ABTERM request</td>
<td>Invoke the monitoring exit</td>
</tr>
<tr>
<td>DRA thread failure</td>
<td>Invoke the status exit</td>
</tr>
<tr>
<td>Resynchronization request issued from CICS recovery manager</td>
<td>Invoke the token exit</td>
</tr>
<tr>
<td>CICS orderly or immediate term</td>
<td>Invoke the token exit</td>
</tr>
<tr>
<td>CICS ABEND</td>
<td>Invoke the token exit</td>
</tr>
<tr>
<td>Completion of DRA TERM issued as a result of a termination request from disconnection program</td>
<td>Invoke the statistics exit</td>
</tr>
<tr>
<td>Completion of DRA TERM issued as a result of a CICS orderly termination request</td>
<td>Invoke the statistics exit</td>
</tr>
</tbody>
</table>

Suspend exit (DFHDBSPX)
The suspend exit is invoked by the adapter on behalf of the DRA so that a CICS thread can be suspended during the processing of a DL/I call. The suspend exit outputs a trace entry immediately before issuing a WAIT, and a trace entry immediately after it is posted by the resume exit.

The suspend exit is also invoked by the adapter when a disconnection request from the menu is being processed.

Resume exit (DFHDBREX)
The resume exit is invoked asynchronously by the adapter on behalf of the DRA, and it is executed in the DRA environment. It handles both normal resume and abnormal resume after an abend of the thread. The resume exit issues an MVS POST.

When a thread fails, the resume exit is invoked and an ‘unsuccessful’ return code is passed back to the DBCTL call processor, indicating that CICS must issue an abend for that thread (task).
Database control (DBCTL)

**Adapter exits**
The following sections describe the adapter exits.

The **adapter statistics exit (DFHDBSTX)**: The statistics exit is invoked by the adapter when the CICS-DBCTL interface has been terminated by the CICS operator using the menu program to request disconnection from DBCTL. The exit is also invoked by the adapter when CICS is terminated in an orderly way.

The function of the exit is to invoke the CICS statistics domain supplying the data that has been returned from the DRA relating to the individual CICS-DBCTL session.

For a `/CHECKPOINT FREEZE` command, the exit is not invoked, but the statistics domain is called by DFHCDBTC.

The **adapter token exit (DFHDBTOX)**: The token exit is invoked by the adapter when a task is encountered which has not been allocated a thread token, that is, it has not been through the DBCTL call processor module. This occurs for resynchronization tasks and for the CICS termination invocation.

The **adapter monitoring exit (DFHDBMOX)**: The monitoring exit is invoked by the adapter when monitoring data has been returned by DBCTL as a result of a PSB schedule request, and a CICS `SYNCPOINT` or `DLI TERM` request. The exit passes the data on to the CICS monitoring domain to update the tasks monitoring information.

The **adapter status exit (DFHDBSSX)**: The status exit is invoked by the adapter in the event of a DRA thread failure, so that resources owned by the failing thread can be transferred to CICS, which then releases the transferred resources during syncpoint processing.

**DBCTL system definition**

DBCTL system definition is described in the IMS *System Definition Reference*.

**DBCTL PSB scheduling**

When a CICS task requests the scheduling of a DL/I PSB by means of an EXEC DLI SCHEDULE request or DL/I PCB call, and the request is for a DBCTL PSB, control is passed to DFHDLIDP.

**Database calls**

For DBCTL, DFHDLIDP invokes the CICS RMI to pass control to DBCTL.

**DBCTL PSB termination**

DBCTL PSB termination is performed during the syncpoint when the resource manager interface (RMI) communicates with DBCTL.

**System termination**

Support is provided to close down the CICS-DBCTL interface during CICS termination. This should avoid the possibility of causing DBCTL to terminate with a U113 abend because of CICS terminating while DL/I threads are running on its behalf in DBCTL.

To provide the support, there is an extension to the RMI to invoke active adapters at CICS termination.
If CICS termination hangs because the CICS-DBCTL interface does not close down, the operator should type in a `/DISPLAY ACTIVE` command on the DBCTL console and identify the threads corresponding to the CICS system being terminated. This is possible because the threads’ recovery tokens, which are displayed, start with the CICS applid. The operator should then issue `/STOP THREAD` requests for each thread.

**Control blocks**

The following diagram shows the major control blocks used to support the CICS-DBCTL interface:

![Diagram of control blocks](image)

The DL/I interface parameter list (DLP) is described in [“DL/I Interface parameter list (DLP)” on page 345](#).

The DBCTL global block (DGB) is acquired, from storage above the 16MB line, when the CICS-DBCTL interface is first initialized. It lasts for the remainder of the CICS execution.

The DBCTL scheduling block (DSB) is acquired, from storage above the 16MB line, when a task issues a PSB schedule request to DBCTL; that is, the PSB used does not appear in the remote PDIR. The DSB is freed at task termination.

See the [CICS Data Areas](#) manual for a detailed description of these control blocks.

**Modules**

<table>
<thead>
<tr>
<th>Module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHDBAT</td>
<td>Adapter</td>
</tr>
<tr>
<td>DFHDBCON</td>
<td>Initialization program</td>
</tr>
<tr>
<td>DFHDBCT</td>
<td>Control program</td>
</tr>
<tr>
<td>DFHDBCTX</td>
<td>Control exit</td>
</tr>
<tr>
<td>DFHDBDI</td>
<td>Disable program</td>
</tr>
<tr>
<td>DFHDBDSC</td>
<td>Termination program</td>
</tr>
<tr>
<td>DFHDBIE</td>
<td>Inquiry screens</td>
</tr>
<tr>
<td>DFHDBIQ</td>
<td>Inquiry program</td>
</tr>
<tr>
<td>DFHDBME</td>
<td>Menu program</td>
</tr>
<tr>
<td>DFHDBMOX</td>
<td>Monitoring exit</td>
</tr>
<tr>
<td>DFHDBNE</td>
<td>Menu screens</td>
</tr>
</tbody>
</table>

---

Database control (DBCTL)

Chapter 20. Database control (DBCTL)  291
### Database control (DBCTL)

<table>
<thead>
<tr>
<th>Module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHDBREX</td>
<td>Resume exit</td>
</tr>
<tr>
<td>DFHDBSPX</td>
<td>Suspend exit</td>
</tr>
<tr>
<td>DFHDBSSX</td>
<td>Status exit</td>
</tr>
<tr>
<td>DFHDBSTX</td>
<td>Statistics exit</td>
</tr>
<tr>
<td>DFHDBTOX</td>
<td>Token exit</td>
</tr>
<tr>
<td>DFHDBUEX</td>
<td>DBCTL user exit</td>
</tr>
<tr>
<td>DFHDLI</td>
<td>DL/I router program</td>
</tr>
<tr>
<td>DFHDLIDP</td>
<td>DBCTL call processor</td>
</tr>
</tbody>
</table>

### Exits

The following global user exit points are provided for DBCTL:
- In DFHDBCRI: XXDFB and XXDTO
- In DFHDBCT: XXDFA.

For further information about these exit points, see the [CICS Customization Guide](#) and the [CICS IMS Database Control Guide](#).
Chapter 21. Data interchange program

The data interchange program (DFHDIP) supports the batch controller functions of the IBM 3790 Communication System and the IBM 3770 Data Communication System. Support is provided for the transmit, print, message, user, and dump data sets of the 3790 system.

Design overview

The data interchange program is designed as a function manager for Systems Network Architecture (SNA) devices. It is invoked via DFHEDI for command-level requests, or internally by the basic mapping support (BMS) routines using the DFHDI macro. DFHDIP performs the following actions:

1. Determines whether a new output destination has been specified (it retains information about the previous destinations in the data interchange control block) and, if so, builds appropriate FMHs to select the new destination, and outputs these FMHs to the SNA device via terminal control.

2. Invokes the appropriate subroutine to perform the desired function:
   - ADD: Builds ADD FMH, transmits it and the user data
   - REPLACE: Builds REPLACE FMH, transmits it and the user data
   - ERASE: Builds ERASE FMH and RECID FMH and transmits them
   - NOTE: Builds NOTE FMH, transmits it, and returns the reply to the user
   - QUERY: Builds QUERY FMH, transmits it, and outputs END FMH
   - SEND: Outputs user data
   - WAIT: Waits for completion of the I/O
   - END: Builds END FMH and transmits it
   - ABORT: Builds ABORT FMH and transmits it
   - ATTACH: Removes FMH from initial input
   - DETACH: Frees the storage used by DFHDIP
   - RECEIVE: Reads a complete record from the logical device.

3. Sets the appropriate return code.

Figure 41 on page 294 shows the data interchange program interfaces.
Data interchange program

Notes:
1. The application program invokes DFHEDI (via DFHEIP) which then communicates with DFHDIP by setting fields in the TCA.
2. DFHDIP receives control.
3. If no storage has been obtained for the data interchange block (DIB), storage control is invoked. The storage is chained to the TCTTE. Significant status information, such as the currently selected destination, is remembered in the data interchange block, which is freed at the end of task processing.
4. A trace entry is made.
5. If logging is present (protected task and message integrity) and if a destination change or function change occurs on output, temporary-storage control is invoked to write the DIB to recoverable temporary storage.
6. Terminal control is invoked to output any built FMH and also to output the user data. (DFHTC TYPE=WRITE is issued.) For input requests, DFHTC TYPE=READ requests are issued to obtain a non-null input record.
7. Any errors obtained from the device are decoded and placed in the TCA return code slot. If no errors were detected, a return code of '0' (zero) is returned.

Modules
DFHEDI, DFHDIP

Exits
No global user exit points are provided for this function.

Trace
The following point ID is provided for the data interchange program:
• AP 00D7, for which the trace level is DI 1.

For more information about the trace points, see the CICS Trace Entries. For more information about using traces in problem determination, see the CICS Problem Determination Guide.
Chapter 22. Directory manager domain (DD)

The directory manager domain (also sometimes known simply as “directory manager”) manages directories of named tokens.

Directory manager domain’s specific gates

Table 36 summarizes the directory manager domain’s specific gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, and the functions provided by the gates.

Table 36. Directory manager domain’s specific gates

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDDI</td>
<td>DD 0201</td>
<td>CREATE_DIRECTORY</td>
</tr>
<tr>
<td></td>
<td>DD 0202</td>
<td>ADD_ENTRY</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DELETE_ENTRY</td>
</tr>
<tr>
<td></td>
<td></td>
<td>REPLACE_DATA</td>
</tr>
<tr>
<td>DDLO</td>
<td>DD 0301</td>
<td>LOCATE</td>
</tr>
<tr>
<td></td>
<td>DD 0302</td>
<td></td>
</tr>
<tr>
<td>DDBR</td>
<td>DD 0401</td>
<td>START_BROWSE</td>
</tr>
<tr>
<td></td>
<td>DD 0402</td>
<td>GET_NEXT_ENTRY</td>
</tr>
<tr>
<td></td>
<td></td>
<td>END_BROWSE</td>
</tr>
</tbody>
</table>

DDDI gate, CREATE_DIRECTORY function

The CREATE_DIRECTORY function of the DDDI gate is used to create a new directory with entry names of a given length.

Input parameters

DIRECTORY_NAME

is the four_character name of the directory to be created.

NAME_LENGTH

is the length of entry names in the directory. This value must be a multiple of four, and less than 256.

Output parameters

DIRECTORY_TOKEN

is the directory token

RESPONSE

is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON]

is returned when RESPONSE is INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>DUPLICATE_DIRECTORY, INVALID_NAME_LEN</td>
</tr>
</tbody>
</table>
Directory manager domain (DD)

**DDDII gate, ADD_ENTRY function**

The ADD_ENTRY function of the DDDII gate is used to add an entry to a directory.

**Input parameters**

**DIRECTORY_TOKEN**

is the token for the directory.

**ENTRY_NAME**

is the address of the entry name. The length is fixed for the directory.

**DATA_TOKEN**

is the data to be associated with the entry name in the directory.

**SUSPEND**

indicates whether Storage Manager GETMAIN requests should be conditional or unconditional. Takes one of the values:

YES|NO

**Output parameters**

**DUPLICATE_DATA_TOKEN**

is the data currently associated with the entry name if it already exists in the directory.

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**[REASON]**

is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>DUPLICATE, INSUFFICIENT_STORAGE</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_DIRECTORY</td>
</tr>
</tbody>
</table>

**DDDII gate, DELETE_ENTRY function**

The DELETE_ENTRY function of the DDDII gate is used to delete an entry from a directory.

**Input parameters**

**DIRECTORY_TOKEN**

is the token for the directory.

**ENTRY_NAME**

is the address of the entry name. The length is fixed for the directory.

**Output parameters**

**DATA_TOKEN**

is the data associated with the entry name when it was deleted.

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**[REASON]**

is returned when RESPONSE is EXCEPTION or INVALID. Possible values
DDDI gate, REPLACE_DATA function

The REPLACE_DATA function of the DDDI gate is used to replace the data associated with an existing entry name in a directory.

Input parameters

DIRECTORY_TOKEN
is the token for the directory.

ENTRY_NAME
is the address of the entry name. The length is fixed for the directory.

NEW_DATA_TOKEN
is the new data to be associated with the entry name.

PRIOR_DATA_TOKEN
is an optional parameter that indicates the data expected to be associated with the entry name just prior to it being replaced.

Output parameters

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NOT_FOUND, DATA_CHANGED</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_DIRECTORY</td>
</tr>
</tbody>
</table>

DDLO gate, LOCATE function

The LOCATE function of the DDLO gate is used to locate the data associated with an existing entry name in a directory.

Input parameters

DIRECTORY_TOKEN
is the token for the directory.

ENTRY_NAME
is the address of the entry name. The length is fixed for the directory.

Output parameters

DATA_TOKEN
is the data associated with the entry name.

RESPONSE
is the domain’s response to the call. It can have any of these values:
Directory manager domain (DD)

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]

is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NOT_FOUND</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_DIRECTORY</td>
</tr>
</tbody>
</table>

DDBR gate, START_BROWSE function

The START_BROWSE function of the DDBR gate is used to start an alphabetical browse through all of the entries in a directory.

**Input parameters**

DIRECTORY_TOKEN

is the token for the directory.

AT_NAME

is the address of an entry name at which the browse is to start. The first name found will be the first which is greater than or equal to this in alphabetical order.

TASK RELATED

is an optional parameter which indicates whether the browse will end at task end. It can be one of these values:

YES|NO

if not specified this parameter defaults to YES.

**Output parameters**

BROWSE_TOKEN

is the token for this browse.

RESPONSE

is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]

is returned when RESPONSE is INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>INVALID_DIRECTORY</td>
</tr>
</tbody>
</table>

DDBR gate, GET_NEXT_ENTRY function

The GET_NEXT_ENTRY function of the DDBR gate is used to get the next entry name in alphabetical order in a directory.

**Input parameters**

DIRECTORY_TOKEN

is the token for the directory.

BROWSE_TOKEN

is the token for the browse.
ENTRY_NAME
is a buffer in which the entry name will be returned.

Output parameters
DATA_TOKEN
is the token associated with the entry name.

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>BROWSE_END</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_DIRECTORY, INVALID_BROWSE, INVALID_NAME</td>
</tr>
</tbody>
</table>

DDBR gate, END_BROWSE function

The END_BROWSE function of the DDBR gate is used to end a browse on a directory.

Input parameters
DIRECTORY_TOKEN
is the token for the directory.

BROWSE_TOKEN
is the token for the browse.

Output parameters
RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>INVALID_DIRECTORY, INVALID_BROWSE</td>
</tr>
</tbody>
</table>

Directory manager domain’s generic gates

Table 37 summarizes the directory manager domain’s generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 37. Directory manager domain’s generic gates

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDDM</td>
<td>DD 0101</td>
<td>PRE_INITIALISE</td>
<td>DMDM</td>
</tr>
<tr>
<td></td>
<td>DD 0102</td>
<td>INITIALISE_DOMAIN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>QUIESCE_DOMAIN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TERMINATE_DOMAIN</td>
<td></td>
</tr>
</tbody>
</table>
Directory manager domain (DD)

For descriptions of these functions and their input and output parameters, refer to the sections dealing with the corresponding generic formats:

In preinitialization the directory manager adds its general subpool and global lock.

In initialization, quiesce, and termination processing, the directory manager domain performs only internal routines.

Exits

No global user exit points are provided in this domain.

Trace

The point IDs for the directory manager domain are of the form DD xxxx; the corresponding trace levels are DD 1, DD 2, and Exc.

For more information about the trace points, see the CICS Trace Entries. For more information about using traces in problem determination, see the CICS Problem Determination Guide.
Chapter 23. Dispatcher domain (DS)

The dispatcher domain is concerned with the attaching, running, and detaching of tasks, and the posting of TCBs with the following modes (names: concurrent (CO), ONC/RPC-owning (RP), quasi-reentrant (QR), resource-owning (RO), file-owning (FO), secondary LU usage (SZ), open key 8 (L8), JVM key 8 (J8), hot-pooling key 8 (H8), sockets (SO), sockets listener (SL), or secure sockets key 8 (S8).

Dispatcher domain’s specific gates

Table 38 summarizes the dispatcher domain’s specific gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and whether or not the functions are available through the exit programming interface (XPI).

Table 38. Dispatcher domain’s specific gates

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>XPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSAT</td>
<td>DS 0002</td>
<td>ATTACH</td>
<td>NO</td>
</tr>
<tr>
<td>DS 0003</td>
<td>CANCEL_TASK</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CHANGE_MODE</td>
<td>YES</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CHANGE_PRIORITY</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CLEAR_MATCH</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DELETE_SUBSPACE_TCBS</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FREE_SUBSPACE_TCBS</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NOTIFY_DELETE_TCB</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SET_PRIORITY</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TCB_POOL_MANAGEMENT</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>DSBR</td>
<td>DS 0010</td>
<td>END_BROWSE</td>
<td>NO</td>
</tr>
<tr>
<td>DS 0011</td>
<td>GET_NEXT</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td></td>
<td>INQUIRE_TASK</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td></td>
<td>INQUIRE_TCB</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SET_TASK</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SET_TCB</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td></td>
<td>START_BROWSE</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>DSIT</td>
<td>DS 0008</td>
<td>ACTIVATE_MODE</td>
<td>NO</td>
</tr>
<tr>
<td>DS 0009</td>
<td>ADD_TCB</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DELETE_ALL_OPEN_TCBS</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DELETE_OPEN_TCB</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DELETE_TCB</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FREE_TCB</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td></td>
<td>INQUIRE_DISPATCHER</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PROCESS_DEAD_TCBS</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SET_DISPATCHER</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>DSSR</td>
<td>DS 0004</td>
<td>ADD_SUSPEND</td>
<td>YES</td>
</tr>
<tr>
<td>DS 0005</td>
<td>DELETE_SUSPEND</td>
<td>YES</td>
<td></td>
</tr>
<tr>
<td></td>
<td>INQUIRE_SUSPEND_TOKEN</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SUSPEND</td>
<td>YES</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RESUME</td>
<td>YES</td>
<td></td>
</tr>
<tr>
<td></td>
<td>WAIT_MVS</td>
<td>YES</td>
<td></td>
</tr>
<tr>
<td></td>
<td>WAIT_OLDW</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td></td>
<td>WAIT_OLDC</td>
<td>NO</td>
<td></td>
</tr>
</tbody>
</table>
DSAT gate, ATTACH function

The ATTACH function of the DSAT gate is used to attach a new task.

- The transaction manager uses the function to attach system or nonsystem tasks that have PCT entries.
- Other parts of CICS use the function to attach system tasks that do not have PCT entries.

This function is used to attach a new task, and add it to the appropriate Dispatcher queue.

When the task is first dispatched, the calling domain receives the TASK_REPLY call at its DSAT gate (see "DSAT format, TASK_REPLY function" on page 325).

Input parameters

**PRIORITY**

affects a task’s dispatching precedence. It can have a value in the range 0 (low priority) through 255 (high priority).

**USER_TOKEN**

is the token by which the task to be attached is known to the caller.

**[TIMEOUT]**

is the deadlock time-out interval, in milliseconds.

**TYPE**

is the type of task. It can have either of these values:

- SYSTEM
- NON_SYSTEM

**[MODE]**

specifies the mode in which the task is to run. It can have any of these values:

- CO (concurrent)
- FO (file-owning)
- HB (hot-pooling)
- J8 (JVM key 8)
- L8 (open key 8)
- QR (quasi-reentrant)
- RD (resource-owning)
- RP (ONC/RPC-owning)
- SL (sockets listener)
- SO (sockets)
- SZ (secondary LU usage)
- S8 (secure sockets key 8)

**[TASK_REPLY_GATE_INDEX]**

is used when a gate other than the attaching domain’s default gate is to receive a resultant TASK_REPLY.

**[SPECIAL_TYPE(SMSY)]**

identifies the special task SMSY.

**[TRANSACTION_TOKEN]**

identifies the transaction associated with the attached task.

Output parameters

**TASK_TOKEN**

is the token by which the attached task is known to the dispatcher.

**RESPONSE**

is the dispatcher’s response to the call. It can have any one of these values:

- OK
- EXCEPTION
- DISASTER
- INVALID
- KERNERROR
- PURGED
DSAT gate, CHANGE_MODE function

The CHANGE_MODE function of DSAT gate is used to move a task from one CICS-managed TCB to another, or to select the mode in which the task is to run.

Input parameters

MODE

is the mode to be used by the task. It can have any of these values:

- CO (concurrent)
- FO (file-owning)
- HB (hot-pooling)
- JB (JVM key 8)
- LB (open key 8)
- QR (quasi-reentrant)
- RO (resource-owning)
- RP (ONC/RPC-owning)
- SL (sockets listener)
- SO (sockets)
- SZ (secondary LU usage)
- SB (secure sockets key 8)

[CONDITIONAL]

states whether the CHANGE_MODE should be conditional on the current load on the CPU. It can have either of these values:

- YES
- NO

MODENAME

2-character mode name with values as for MODE

MODENAME_TOKEN

token representing modename. More efficient than using MODENAME. The token is returned by ACTIVATE_MODE and by CHANGE_MODE (see OLD_MODENAME_TOKEN below)

TCB_TOKEN

token representing the TCB instance to which to switch. The token is returned by CHANGE_MODE (see OLD_TCB_TOKEN below)

[PRIMARY MATCH]

an 8-byte token to be used to search for a matching free TCB instance to which to switch.

[SECONDARY MATCH]

an 8-byte token to be used to search for a matching free TCB instance to which to switch.

[MAXSTRATEGY]

the strategy to be followed if a TCB instance that satisfies the PRIMARY_MATCH and SECONDARY_MATCH values is not found. The only value allowed is:

- EXACT_THEN_NEW_THEN_BEST
Dispatcher domain (DS)

Output parameters

OLD_MODE

is the mode used by the task when the CHANGE_MODE request was issued. It can have any of these values:

- CO (concurrent)
- FO (file-owning)
- HB (hot-pooling)
- J8 (JVM key 8)
- L8 (open key 8)
- QR (quasi-reentrant)
- RO (resource-owning)
- RP (ONC/RPC-owning)
- SL (sockets listener)
- SO (sockets)
- SZ (secondary LU usage)
- S8 (secure sockets key 8)

OLD_MODENAME

is the mode used by the task when the CHANGE_MODE request was issued. It can have the same values as OLD_MODE. OLD_MODENAME is preferred to OLD_MODE.

OLD_MODENAME_TOKEN

is a token representing the mode used by the task when the CHANGE_MODE request was issued.

OLD_TCB_TOKEN

is a token representing the TCB used by the task when the CHANGE_MODE request was issued.

MATCH_RESULT

indicates the level of success of the matching process. It can have any of these values:

- EXACT_MATCH
- NO_MATCH
- PRIM_NOT_SEC_MATCH
- NOT_APPLIC

NEW_TCB_TOKEN

token representing the TCB instance returned by the matching process.

RESPONSE

is the dispatcher’s response to the call. It can have any one of these values:

- OK
- DISASTER
- EXCEPTION
- INVALID
- KERNERROR
- PURGED

REASON

is returned when RESPONSE is DISASTER, EXCEPTION, INVALID, or PURGED. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>LOCK_FAILED, ACTIVATE_MODE_FAILED, ADD_TCB_FAILED, SUSPEND_FAILED</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>MODE_NOT_ACTIVE, NO_TCBS_ACTIVE, INSUFFICIENT_STORAGE, TCB_FAILED, TOO_FEW_TCBS</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_MODENAME, INVALID_MODENAME_TOKEN, INVALID_TCB_TOKEN</td>
</tr>
<tr>
<td>PURGED</td>
<td>TIMED_OUT, TASK_CANCELLED</td>
</tr>
</tbody>
</table>
**DSAT gate, CLEAR_MATCH function**

The CLEAR_MATCH function of the DSAT gate causes all match tokens associated with the calling TCB to be discarded.

**Input parameters**

None

**Output parameters**

RESPONSE

is the dispatcher’s response to the call. It can have only one value:

0K

**DSAT gate, NOTIFY_DELETE_TCB function**

The NOTIFY_DELETE function of the DSAT gate notifies the interested domain (AP) that a DELETE_TCB request is in progress.

**Input parameters**

TCB_TOKEN

The DS token representing the TCB instance for which notification is required when deleted.

**Output parameters**

RESPONSE

is the dispatcher’s response to the call. It can have only one value:

0K

**DSAT gate, CHANGE_PRIORITY function**

The CHANGE_PRIORITY function of DSAT gate has two effects:
1. It changes the dispatch priority of the issuing task.
2. It causes control to be given up to another task.

**Input parameters**

[PRIORITY]

is the new priority. It can have a value in the range 0 (low priority) through 255 (high priority).

**Output parameters**

[OLD_PRIORITY]

is the task’s former priority. It can have a value in the range 0 (low priority) through 255 (high priority).

RESPONSE

is the dispatcher’s response to the call. It can have any one of these values:

0K|DISASTER|INVALID|KERNERROR

**DSAT gate, SET_PRIORITY function**

The SET_PRIORITY function of DSAT gate changes the priority of the issuing task, or the task specified by the TASK_TOKEN parameter.
Dispatcher domain (DS)

**Input parameters**

**PRIORITY**

is the new priority. It can have a value in the range 0 (low priority) through 255 (high priority).

**[TASK_TOKEN]**

identifies the task whose priority is to be changed.

**[SPECIAL_TYPE(IMMEDIATE_SHUTDOWN_TASK)]**

identifies the special task “IMMEDIATE_SHUTDOWN_TASK”:

**Output parameters**

**[OLD_PRIORITY]**

is the task’s former priority. It can have a value in the range 0 (low priority) through 255 (high priority).

**[RESPONSE]**

is the dispatcher’s response to the call. It can have any one of these values:

OK|DISASTER|EXCEPTION|INVALID|KERNERROR

**[REASON]**

is returned when RESPONSE is EXCEPTION. It has the value:

INVALID_TASK_TOKEN

**DSAT gate, CANCEL_TASK function**

The CANCEL_TASK function of DSAT gate causes a specified task to be canceled.

**Input parameters**

**TASK_TOKEN**

is the token representing the task to be canceled.

**CANCEL_TYPE**

can have either of these values:

FORCE_CANCEL|NORMAL_CANCEL

**Output parameters**

**RESPONSE**

is the dispatcher’s response to the call. It can have any one of these values:

OK|DISASTER|EXCEPTION|INVALID|KERNERROR

**[REASON]**

is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>INVALID_TASK_TOKEN, INVALID_STATE, NOT_PURGEABLE, CANCEL_INHIBITED</td>
</tr>
<tr>
<td></td>
<td>INVALID_STATE_PURGE</td>
</tr>
</tbody>
</table>

**DSAT gate, FREE_SUBSPACE_TCBS function**

The FREE_SUBSPACE_TCBS function of DSAT gate releases any open subspace TCBs owned by the task, and makes them available for use by another task executing with the same subspace, or deletes the TCBs if the task is ‘unclean’.

**Input parameters**

None
Output parameters

**OPEN_TCBS_USED_AND_KEPT**
is a bit string indicating which TCB modes were used by the task, of and are now available to other tasks

**OPEN_TCBS_USED_AND_LOST**
is a bit string indicating which TCB modes were used by the task, of and have now been deleted because the task was ‘unclean’

**RESPONSE**
is the dispatcher’s response to the call. It can have any one of these values: OK|DISASTER|INVALID|KERNERROR

**[REASON]**
is returned when RESPONSE is DISASTER or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>LOCK_FAILED, NCEL_INHIBITED</td>
</tr>
<tr>
<td>INVALID</td>
<td>NOT_SUBSPACE_ELIGIBLE, NCEL_INHIBITED</td>
</tr>
</tbody>
</table>

**DSAT gate, DELETE_SUBSPACE_TCBS function**
The DELETE_SUBSPACE_TCBS function of DSAT gate deletes any open TCBs associated with the given subspace.

**Input parameters**

**SUBSPACE_TOKEN**
indicates the subspace whose associated open TCBs are to be deleted

**Output parameters**

**RESPONSE**
is the dispatcher’s response to the call. It can have any one of these values: OK|DISASTER|EXCEPTION|KERNERROR

**[REASON]**
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>LOCK_FAILED NCEL_INHIBITED</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>TOO_FEW_TCBS NCEL_INHIBITED</td>
</tr>
</tbody>
</table>

**DSAT gate, TCB_POOL_MANAGEMENT function**
The TCB_POOL_MANAGEMENT function of DSAT gate deletes unallocated TCBs which are excess to current requirements.

**Input parameters**
None

**Output parameters**

**RESPONSE**
is the dispatcher’s response to the call. It can have any one of these values: OK|DISASTER|KERNERROR
Dispatcher domain (DS)

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>LOCK_FAILED NCEL_INHIBITED</td>
</tr>
</tbody>
</table>

**DSBR gate, START_BROWSE function**
The START_BROWSE function of DSBR gate starts a browse session with the dispatcher.

**Input parameters**
None.

**Output parameters**
- **BROWSE_TOKEN**
  is the token representing this browse session.
- **RESPONSE**
  is the dispatcher’s response to the call. It can have any one of these values:
  OK|DISASTER|PURGED

**DSBR gate, END_BROWSE function**
The END_BROWSE function of DSBR gate ends a browse session with the dispatcher.

**Input parameters**
- **BROWSE_TOKEN**
  is the token identifying the browse session to be ended.

**Output parameters**
- **RESPONSE**
  is the dispatcher’s response to the call. It can have any one of these values:
  OK|DISASTER|INVALID|KERNERROR

[REASON]
is returned when RESPONSE is INVALID. It has this value:
INVALID_BROWSE_TOKEN

**DSBR gate, GET_NEXT function**
The GET_NEXT function of DSBR gate returns information about the next task.

**Input parameters**
- **BROWSE_TOKEN**
  is the token identifying the browse session.

**Output parameters**
- **TASK_TOKEN**
  is the token by which the task is known to the dispatcher.
- **DOMAIN_INDEX**
  is the 2-character index identifying the domain that made the ATTACH call for the task.
OPEN_MODES
is a 32-bit string which indicates which modes of open TCBs were used by
this task.

PRIORITY
is the task’s dispatch priority. It can have a value in the range 0 (low
priority) through 255 (high priority).

TYPE
is the type of task. It can have either of these values:
SYSTEM|NON_SYSTEM

STATE
is the state of the task. It can have any one of these values:
READY|RUNNING|SUSPENDED

[RESOURCE_NAME]
is the name of the resource that the task is waiting for, if the task is
suspended.

[RESOURCE_TYPE]
is the type of resource that the task is waiting for, if the task is suspended.

[RESOURCE_TIME]
is the interval of time that has passed since the task last issued a suspend
or wait.

USER_TOKEN
is the token by which the task is known to the caller that made the
ATTACH request for the task.

SUSPEND_TOKEN
is the token by which the dispatcher recognizes a task to be suspended or
resumed.

MODE
is the mode in which the task is to run. It can have any one of these
values:
CO (concurrent)
FO (file-owning)
HB (hot-pooling)
J8 (JVM key 8)
L8 (open key 8)
QR (quasi-reentrant)
RO (resource-owning)
RP (ONC/RPC-owning)
SL (sockets listener)
SO (sockets)
SZ (secondary LU usage)
S8 (secure sockets key 8)

RESPONSE
is the dispatcher’s response to the call. It can have any one of these values:
OK|DISASTER|EXCEPTION|INVALID|KERNERROR

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>END or INVALID_BROWSE_TOKEN</td>
</tr>
</tbody>
</table>
DSBR gate, INQUIRE_TASK function

The INQUIRE_TASK function of DSBR gate returns information about a specified task.

Input parameters

INPUT_TASK_TOKEN
is the token for the task to be inquired on.

Output parameters

TASK_TOKEN
is the token by which the task is known to the dispatcher.

DOMAIN_INDEX
is the 2-character index identifying the domain that made the ATTACH call for the task.

OPEN_MODES
is a 32-bit string which indicates which modes of open TCBs were used by this task.

PRIORITY
is the task’s dispatch priority. It can have a value in the range 0 (low priority) through 255 (high priority).

TYPE
is the type of task. It can have either of these values:
SYSTEM|NON_SYSTEM

STATE
is the state of the task. It can have any one of these values:
READY|RUNNING|SUSPENDED

[RESOURCE_NAME]
is the name of the resource that the task is waiting for, if the task is suspended.

[RESOURCE_TYPE]
is the type of resource that the task is waiting for, if the task is suspended.

[RESOURCE_TIME]
is the interval of time that has passed since the task last issued a suspend or wait.

USER_TOKEN
is the token by which the task is known to the caller that made the ATTACH request for the task.

SUSPEND_TOKEN
is the token by which the dispatcher recognizes a task to be suspended or resumed.

MODE
is the mode in which the task is to run. It can have any one of these values:
CO (concurrent)
FO (file-owning)
H8 (hot-pooling)
J8 (JVM key 8)
L8 (open key 8)
QR (quasi-reentrant)
RO (resource-owning)
RP (ONC/RPC-owning)
**Dispatcher domain (DS)**

SL (sockets listener)
SO (sockets)
SZ (secondary LU usage)
S8 (secure sockets key 8)

**RESPONSE**
is the dispatcher’s response to the call. It can have any one of these values:
OK|DISASTER|EXCEPTION|INVALID|KERNERROR

[REASON]
is returned when RESPONSE is INVALID. It has this value:
INVALID_TASK_TOKEN

**DSBR gate, SET_TASK function**
The SET_TASK function of DSBR gate marks the task as "unclean" so that open TCBs will be freed at task termination.

**Input parameters**
CLEANLINESS
specifies that the task is to marked "unclean". It can take only the value UNCLEAN

**DSBR gate, INQUIRE_TCB function**
The INQUIRE_TCB function of the DSBR gate returns the AP TCB-related token associated with the specified DS TCB_TOKEN. If the AP token has not yet been set by SET_TCB, then the function returns an AP_TCB_TOKEN value of zero.

**Input parameters**

[TCB_TOKEN]
token provided by DS representing the TCB instance for which the associated AP-related token is required. If this is omitted, the token of the running TCB is assumed.

**Output parameters**

AP_TCB_TOKEN
token provided by AP associated with the TCB instance defined by TCB_TOKEN.

RESPONSE
is the dispatcher’s response to the call. It can have one of these values:
OK|INVALID

[REASON]
is returned when RESPONSE is INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>INVALID_TCB_TOKEN</td>
</tr>
</tbody>
</table>

**DSBR gate, SET_TCB function**
The SET_TCB function of the DSBR gate sets the AP TCB-related token to be associated with the running TCB.

**Input parameters**

AP_TCB_TOKEN
token provided by AP to be associated with the running TCB.
Dispatcher domain (DS)

Output parameters

RESPONSE

is the dispatcher’s response to the call. It can have only one value:

OK

DSIT gate, INQUIRE_DISPATCHER function

The INQUIRE_DISPATCHER function of DSIT gate returns information about the current state of the dispatcher.

Input parameters

None.

Output parameters

[NUMBER_OF_SUBTASKS]

is the number of subtasks for concurrent mode.

[SCAN_DELAY_INTERVAL]

is the delay before terminal control is dispatched after a terminal is posted by the access method.

[MAXIMUM_WAIT_INTERVAL]

is the maximum delay before terminal control is dispatched.

[PRIORITY_MULTIPLIER]

determines how the priority of new tasks is to be penalized in ‘storage getting short’ and ‘storage critical’ situations.

[QR_BATCHING_VALUE]

is the number of POSTs for BATCH=YES waits in quasi-reentrant mode.

[MAXOPENTCBS]

is the current maximum number of Open TCBs allowed.

[ACTOPENTCBS]

is the number of Open TCBs being used by current tasks.

[RP_TCB_ATTACHED]

indicates whether or not the RP TCB is attached. It can have either of these values:

YES|NO

[SZ_TCB_ATTACHED]

indicates whether or not the SZ TCB is attached. It can have either of these values:

YES|NO

RESPONSE

is the dispatcher’s response to the call. It can have either of these values:

OK|DISASTER

DSIT gate, SET_DISPATCHER function

The SET_DISPATCHER function of DSIT gate sets the state of the dispatcher.

Input parameters

[NUMBER_OF_SUBTASKS]

is the number of subtasks for concurrent mode.
[SCAN_DELAY_INTERVAL]
is the delay before terminal control is dispatched after a terminal is posted by the access method.

[MAXIMUM_WAIT_INTERVAL]
is the maximum delay before terminal control is dispatched.

[PRIORITY_MULTIPLIER]
determines how quickly a task’s priority increases as it waits to be dispatched. The faster it increases the less likely a low priority task is to be held up for long periods by higher priority tasks in a busy system.

[QR_BATCHING_VALUE]
is the number of POSTs for BATCH=YES waits in quasi reentrant mode.

[MAXOPENTCBS]
is the maximum number of Open TCBs CICS will allow to exist.

Output parameters
RESPONSE
is the dispatcher’s response to the call. It can be any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>MAXWAIT_LESSTHAN_SCANDELAY, MAXOPENTCBS_OUT_OF_RANGE</td>
</tr>
</tbody>
</table>

DSIT gate, ACTIVATE_MODE function
The ACTIVATE_MODE function creates a mode to which TCBs can be added (by ADD_TCB) so that tasks can CHANGE_MODE to the TCBs.

Input parameters

[MODE]
is the mode to be activated. MODE and MODENAME are mutually exclusive but one must be specified. Mode is a single byte whose values correspond to the pre 1.3 modes QR, RO, FO, CO, SZ and RP.

[MODENAME]
is the two character string that becomes the block name of the sub dispatcher block created by ACTIVATE_MODE.

[IDENTITY]
is the name of the mode to be activated. It is a two byte character string.

[EXEC_CAPABLE]
indicates whether TCBs in this mode are to be set up to support the use of EXEC CICS commands by code running on them.

[LE_ENVIRONMENT]
indicates whether LE is to run in native MVS mode or in CICS mode on TCBs in this mode.

[TCB_KEY]
indicates the key to be specified on ATTACHes of TCBs in this mode.
Dispatcher domain (DS)

**[INHERIT_SUBSPACE]**
indicates whether TCBs in this mode will be able to run application code in a subspace.

**[ESSENTIAL_TCB]**
indicates whether CICS is to be brought down if a TCB in this mode suffers a non recoverable abend.

**[PRTY_RELATIVE_TO_QR]**
allows TCBs in this mode to have a different priority to that of the QR TCB.

**[MULTIPLE_TCBS]**
indicates whether this mode allows more than one TCB.

**[OPEN]**
indicates whether TCBs in this mode are to be managed by the Dispatcher domain as “Open TCBs”.

**[NOTIFY_DELETE]**
indicates which domain, if any, to notify when a DELETE_TCB is issued. It can be one of these values:

- AP
- NONE

Output parameters

**[MODENAME_TOKEN]**
is a token that identifies this modename.

**RESPONSE**
is the dispatcher’s response to the call. It can be any one of these values:

- OK
- EXCEPTION
- DISASTER
- INVALID
- KERNERROR

**[REASON]**
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>MODE_ALREADY_ACTIVE, INSUFFICIENT_STORAGE, MODENAME_ALREADY_ACTIVE, RESERVED_MODENAME, MODE_LIMIT_REACHED</td>
</tr>
</tbody>
</table>

**DSIT gate, ADD_TCB function**
The ADD_TCB function adds a TCB to a particular mode.

Input parameters

**[MODENAME]**
specifies the name of the mode the TCB is to be added to. MODENAME and MODENAME_TOKEN are mutually exclusive but one of them must be coded.

**[MODENAME_TOKEN]**
identifies mode the TCB is to be added by using the token returned by the ACTIVATE_MODE.

**[IDENTITY]**
is an eight character string to placed in the Dispatcher’s block that represents the TCB.
Output parameters

[TCB_TOKEN]
is a token that uniquely identifies this TCB.

RESPONSE
is the dispatcher’s response to the call. It can be any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>INSUFFICIENT_STORAGE, RESERVED_MODENAME</td>
</tr>
</tbody>
</table>

DSIT gate, DELETE_TCB function

The DELETE_TCB function is used by the caller to tell the Dispatcher that the TCB is to be shutdown and that the associated control blocks can be freed. If an attempt is made to shut down an essential TCB, an EXCEPTION response is returned with a reason of NOT_SUPPORTED.

NB no quiescing of tasks on the TCB is performed.

Input parameters

[TCB_TOKEN]
is a token provided by DS that uniquely identifies the TCB.

Output parameters

RESPONSE
is the dispatcher’s response to the call. It can be any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NOT_SUPPORTED</td>
</tr>
</tbody>
</table>

DSIT gate, DELETE_OPEN_TCB function

DELETE_OPEN_TCB schedules the termination of an open TCB. If the TCB is currently in use, the termination will occur when the owning task terminates.

Input parameters

[TCB_TOKEN]
is a token provided by DS that uniquely identifies the TCB.

Output parameters

RESPONSE
is the dispatcher’s response to the call. It can be any one of these values:
OK|DISASTER|INVALID|KERNERROR

[REASON]
is returned when RESPONSE is INVALID. Possible values are:
DSIT gate, **DELETE_ALL_OPEN_TCBS function**

DELETE_ALL_OPEN_TCBS schedules the termination of all open TCBs with a given modename. For TCBs that are currently in use, the termination will occur when the owning task terminates. The function does not prevent new TCBs of the given mode from being created.

**Input parameters**

**MODENAME**

is the name of the mode of the TCBs to be deleted.

**Output parameters**

**RESPONSE**

is the dispatcher’s response to the call. It can be any one of these values:

- OK
- DISASTER
- INVALID
- KERNERROR

**[REASON]**

is returned when RESPONSE is INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>INVALID_TCB_TOKEN</td>
</tr>
</tbody>
</table>

DSIT gate, **FREE_TCB function**

The FREE_TCB function is issued by the Kernel and tells the Dispatcher that a given TCB has terminated and been DETACHed.

**Input parameters**

**[TCB_TOKEN]**

is a token that uniquely identifies the TCB.

**Output parameters**

**RESPONSE**

is the dispatcher’s response to the call. It can be any one of these values:

- OK
- EXCEPTION
- DISASTER
- INVALID
- KERNERROR

**[REASON]**

is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>TASK_NOT_TERMINATED</td>
</tr>
</tbody>
</table>

DSIT gate, **PROCESS_DEAD_TCBS function**

The PROCESS_DEAD_TCBS function is issued by the SM system task each time it runs to tell the Dispatcher to process any TCBs it finds on its dead TCB chain. Such TCBs will be in an MVS WAIT issued by their ESTAE exit after suffering a non recoverable abend.

**Input parameters**

NONE
Output parameters
RESPONSE
is the dispatcher’s response to the call. It can be any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR

DSSR gate, ADD_SUSPEND function
The ADD_SUSPEND function of DSSR gate returns a suspend token which is used to identify a task to be suspended or resumed.

Input parameters
[RESOURCE_NAME]
is the name of the resource that the task is suspended on.

[RESOURCE_TYPE]
is the type of resource that the task is suspended on.

Output parameters
SUSPEND_TOKEN
is the token that is used to identify the task to be suspended or resumed.

RESPONSE
is the dispatcher’s response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR

[REASON]
is returned when RESPONSE is DISASTER. It has this value:
INSUFFICIENT_STORAGE

DSSR gate, DELETE_SUSPEND function
The DELETE_SUSPEND function of DSSR gate discards a suspend token.

Input parameters
SUSPEND_TOKEN
is the suspend token to be deleted.

Output parameters
RESPONSE
is the dispatcher’s response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR

[REASON]
is returned when RESPONSE is INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>INVALID_SUSPEND_TOKEN, SUSPEND_TOKEN_IN_USE</td>
</tr>
</tbody>
</table>

DSSR gate, INQUIRE_SUSPEND_TOKEN function
The INQUIRE_SUSPEND_TOKEN function of DSSR gate can be used to obtain the default suspend token for a task.

Input parameters
None.
Dispatcher domain (DS)

Output parameters

SUSPEND_TOKEN
is the default suspend token for a task.

RESPONSE
is the dispatcher’s response to the call. It can have either of these values:
OK|DISASTER

DSSR gate, SUSPEND function

The SUSPEND function of DSSR gate causes a running task to be suspended.

Input parameters

Note: [INTERVAL] and [DEADLOCK_ACTION] are mutually exclusive parameters.

SUSPEND_TOKEN
is the token identifying the task to be suspended.

PURGEABLE
is the purgeable status of the task. It can have either of these values:
YES|NO

[INTERVAL]
is an interval (in units as specified by TIME_UNIT) after which the task is given back control if it has not been resumed by a DSSR RESUME call.

[DEADLOCK_ACTION]
describes whether the suspended task should be purged if deadlock is detected, and if so, how it should be purged. It can have any one of these values:
DELAYED|IMMEDIATE|INHIBIT

[RESOURCE_NAME]
is the name of the resource that the task is suspended on.

[RESOURCE_TYPE]
is the type of resource that the task is suspended on.

[TIME_UNIT]
identifies the time units specified on the INTERVAL and DELAY parameters where present. It can have either of these values:
SECOND|MILLI_SECOND

[DELAY]
is an interval (in units as specified by TIME_UNIT) during which the task is not dispatched if CICS has other work to do.

[RETRY]
indicates whether or not the dispatcher is to retry the suspend operation, if the running task is not suspended by a preceding suspend operation. It can have either of these values:
YES|NO

[WLM_WAIT_TYPE]
indicates the reason for task’s wait state to the MVS workload manager. It can have any of these values:
Output parameters

[COMPLETION_CODE]
is a completion code supplied by the resumed task.

RESPONSE
is the dispatcher’s response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|
KERNERROR|PURGED

[REASON]
is returned when RESPONSE is INVALID or PURGED. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>INVALID_SUSPEND_TOKEN, ALREADY_SUSPENDED, CLEAN_UP_PENDING</td>
</tr>
<tr>
<td>PURGED</td>
<td>TASK_CANCELLED, TIMED_OUT</td>
</tr>
</tbody>
</table>

DSSR gate, RESUME function

The RESUME function of DSSR gate causes a suspended task to be resumed.

Input parameters

SUSPEND_TOKEN
is the token identifying the task to be resumed.

[COMPLETION_CODE]
is a completion code to be passed from the resumed task to the suspended task.

Output parameters

RESPONSE
is the dispatcher’s response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|
KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>TASK_CANCELLED, TIMED_OUT</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_SUSPEND_TOKEN, ALREADY_RESUMED</td>
</tr>
</tbody>
</table>

DSSR gate, WAIT_MVS function

The WAIT_MVS function of DSSR gate causes a task to wait on an ECB, or list of ECBS, to be posted via the MVS POST service.
Dispatcher domain (DS)

Input parameters

Note: ECB_ADDRESS and ECB_LIST_ADDRESS are mutually exclusive parameters; [INTERVAL] and [DEADLOCK_ACTION] are also mutually exclusive.

ECB_ADDRESS
is the address of the ECB for the task.

ECB_LIST_ADDRESS
is the address of a list of ECBs for the task.

PURGEABLE
is the purgeable status of the task. It can have either of these values:
YES|NO

[INTERVAL]
is an interval (in units as specified by TIME_UNIT) after which the task is given back control if it has not been resumed by a DSSR RESUME call.

[DEADLOCK_ACTION]
describes whether the suspended task should be purged if deadlock is detected, and if so, how it should be purged. It can have any one of these values:
DELAYED|IMMEDIATE|INHIBIT

[RESOURCE_NAME]
is the name of the resource that the task is suspended on.

[RESOURCE_TYPE]
is the type of resource that the task is suspended on.

[BATCH]
states whether requests are to be batched. It can have either of these values:
YES|NO

[TIME_UNIT]
identifies the time units specified on the INTERVAL and DELAY parameters where present. It can have either of these values:
SECOND|MILLI_SECOND

[DELAY]
is an interval (in units as specified by TIME_UNIT) during which the task is not dispatched if CICS has other work to do.

[RETRY]
indicates whether or not the dispatcher is to retry the suspend operation, if the running task is not suspended by a preceding suspend operation. It can have either of these values:
YES|NO

[WLM_WAIT_TYPE]
indicates the reason for task’s wait state to the MVS workload manager. It can have any of these values:
LOCK|IO|CMORESP|DISTRIB|
SESS_LOCALMVS|SESS_NETWORK|
SESS_SYSPLEX|TIMER|OTHER_PRODUCT|
MISC|IDLE
Output parameters

RESPONSE
is the dispatcher’s response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|
KERNERROR|PURGED

[REASON]
is returned when RESPONSE is INVALID or PURGED. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>ALREADY_WAITING, INVALID_ECB_ADDR</td>
</tr>
<tr>
<td>PURGED</td>
<td>TASK_CANCELLED, TIMED_OUT</td>
</tr>
</tbody>
</table>

DSSR gate, WAIT_OLDW function

The WAIT_OLDW function of DSSR gate causes a task to wait on an ECB, or list of
ECBs, that may be posted via the MVS POST service or by setting the POST bit
(X'40' in the first byte). This is supported only in QR mode.

Input parameters

Note: ECB_ADDRESS and ECB_LIST_ADDRESS are mutually exclusive
parameters; [INTERVAL] and [DEADLOCK_ACTION] are also mutually exclusive.

ECB_ADDRESS
is the address of the ECB for the task.

ECB_LIST_ADDRESS
is the address of a list of ECBs for the task.

PURGEABLE
is the purgeable status of the task. It can have either of these values:
YES|NO

[INTERVAL]
is an interval (in units as specified by TIME_UNIT) after which the
task is given back control if it has not been resumed by a DSSR
RESUME call.

[DEADLOCK_ACTION]
describes whether the suspended task should be purged if deadlock
is detected, and if so, how it should be purged. It can have any one
of these values:
DELAYED|IMMEDIATE|INHIBIT

[RESOURCE_NAME]
is the name of the resource that the task is suspended on.

[RESOURCE_TYPE]
is the type of resource that the task is suspended on.

[SPECIAL_TYPE(CSTP)]
identifies the special task CSTP.

[TIME_UNIT]
identifies the time units specified on the INTERVAL and DELAY
parameters where present. It can have either of these values:
SECOND|MILLI_SECOND
Dispatcher domain (DS)

[DELAY]

is an interval (in units as specified by TIME_UNIT) during which the task is not dispatched if CICS has other work to do.

[RETRY]

indicates whether or not the dispatcher is to retry the suspend operation, if the running task is not suspended by a preceding suspend operation. It can have either of these values:

YES | NO

[WLM_WAIT_TYPE]

indicates the reason for task’s wait state to the MVS workload manager. It can have any of these values:

LOCK | IO | CONV | CMORESP | DISTRIB | SESS_LOCALMVS | SESS_NETWORK | SESS_SYSPLEX | TIMER | OTHER_PRODUCT | MISC | IDLE

Output parameters

RESPONSE

is the dispatcher’s response to the call. It can have any one of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON]

is returned when RESPONSE is INVALID or PURGED. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>PURGED</td>
<td>TASK_CANCELLED, TIMED_OUT</td>
</tr>
<tr>
<td>INVALID</td>
<td>ALREADY_WAITING, INVALID_ECB_ADDR, INVALID_MODE</td>
</tr>
</tbody>
</table>

DSSR gate, WAIT_OLDC function

The WAIT_OLDC function of DSSR gate causes a task to wait on an ECB that must be posted by setting the ‘X’40’ bit rather than via the MVS POST service. This is supported only in QR mode.

Input parameters

Note: [INTERVAL] and [DEADLOCK_ACTION] are mutually exclusive parameters.

ECB_ADDRESS

is the address of the ECB for the task.

PURGEABLE

is the purgeable status of the task. It can have either of these values:

YES | NO

[INTERVAL]

is an interval (in units as specified by TIME_UNIT) after which the task is given back control if it has not been resumed by a DSSR RESUME call.

[DEADLOCK_ACTION]

describes whether the suspended task should be purged if deadlock is detected, and if so, how it should be purged. It can have any one of these values:
[RESOURCE_NAME] is the name of the resource that the task is suspended on.

[RESOURCE_TYPE] is the type of resource that the task is suspended on.

[TIME_UNIT] identifies the time units specified on the INTERVAL and DELAY parameters where present. It can have either of these values: SECOND|MILLI_SECOND

[DELAY] is an interval (in units as specified by TIME_UNIT) during which the task is not dispatched if CICS has other work to do.

[RETRY] indicates whether or not the dispatcher is to retry the suspend operation, if the running task is not suspended by a preceding suspend operation. It can have either of these values: YES|NO

[WLM_WAIT_TYPE] indicates the reason for task’s wait state to the MVS workload manager. It can have any of these values:
LOCK|IO|CONV|CMDRESP|DISTRIB|
SESS_LOCALMVS|SESS_NETWORK|
SESS_SYSPLEX|TIMER|OTHER_PRODUCT|
MISC|IDLE

Output parameters

RESPONSE is the dispatcher’s response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|
KERNERROR|PURGED

[REASON] is returned when RESPONSE is INVALID or PURGED. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>ALREADY_WAITING, INVALID_ECB_ADDR, INVALID_MODE</td>
</tr>
<tr>
<td>PURGED</td>
<td>TASK_CANCELLED, TIMED_OUT</td>
</tr>
</tbody>
</table>

Dispatcher domain’s generic gates

Table 39 summarizes the dispatcher domain’s generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 39. Dispatcher domain’s generic gates

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMDM</td>
<td>DS 0006</td>
<td>PRE_INITIALISE</td>
<td>DMDM</td>
</tr>
<tr>
<td></td>
<td>DS 0007</td>
<td>INITIALISE_DOMAIN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>QUIESCE_DOMAIN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TERMINATE_DOMAIN</td>
<td></td>
</tr>
</tbody>
</table>
Dispatcher domain (DS)

Table 39. Dispatcher domain’s generic gates (continued)

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>KEDS</td>
<td>DS 0012</td>
<td>TCB_REPLY</td>
<td>KEDS</td>
</tr>
<tr>
<td></td>
<td>DS 0013</td>
<td>TASK_REPLY</td>
<td></td>
</tr>
<tr>
<td>SMNT</td>
<td>DS 0145</td>
<td>STORAGE_NOTIFY</td>
<td>SMNT</td>
</tr>
<tr>
<td></td>
<td>DS 0113</td>
<td></td>
<td></td>
</tr>
<tr>
<td>STST</td>
<td>DS 0020</td>
<td>COLLECT_STATISTICS</td>
<td>STST</td>
</tr>
<tr>
<td></td>
<td>DS 0021</td>
<td>COLLECT_RESOURCE_STATS</td>
<td></td>
</tr>
<tr>
<td>APUE</td>
<td>DS 0121</td>
<td>SET_EXIT_STATUS</td>
<td>APUE</td>
</tr>
<tr>
<td></td>
<td>DS 0122</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For descriptions of these functions and their input and output parameters, you need to refer to the sections dealing with the corresponding generic formats:

- Functions and parameters

  Format APUE—“Application domain’s generic formats” on page 100
  Format DMDM—“Domain manager domain’s generic formats” on page 377
  Format KEDS—“Kernel domain’s generic formats” on page 752
  Format STST—“Statistics domain’s generic format” on page 1087
  Format SMNT—“Storage manager domain’s generic format” on page 1116

In preinitialization processing, the dispatcher domain sets the initial dispatching options:
- The priority aging value (PRTYAGE)
- Whether or not tasks are to be run in concurrent mode (SUBTSKS)
- The terminal scan delay interval (ICVTSD)
- The region exit time (ICV).

For a cold start, the information comes from the system initialization parameters (given in parentheses); for any other type of start, the information comes from the local catalog, but is then modified by any relevant system initialization parameters.

Dispatcher domain’s generic formats

Table 40 describes the generic formats owned by the dispatcher domain and shows the functions performed on the calls.

Table 40. Generic formats owned by dispatcher domain

<table>
<thead>
<tr>
<th>Format</th>
<th>Calling module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSAT</td>
<td>DFHDSKE</td>
<td>TASK_REPLY</td>
</tr>
<tr>
<td></td>
<td>DFHDSDS4</td>
<td>PURGE_INHIBIT_QUERY</td>
</tr>
</tbody>
</table>

In the descriptions of the formats that follow, the “input” parameters are input not to the dispatcher, but to the domain being called by the dispatcher. Similarly, the “output” parameters are output by the domain that was called by the dispatcher, in response to the call.
DSAT format, TASK_REPLY function

The TASK_REPLY function of DSAT format is used to notify the domain that attached a task that the task has had its first dispatch.

**Input parameters**

**USER_TOKEN**
- is the token by which the task that has been dispatched is known to the called domain.

**TASK_TOKEN**
- is the token by which the task that has been dispatched is known to the dispatcher.

**SUSPEND_TOKEN**
- is the suspend token that the task can be suspended against by default.

**Output parameters**

**RESPONSE**
- is the called domain’s response to the call. It can have any one of these values:
  - OK
  - DISASTER
  - INVALID
  - KERNERROR

DSAT format, PURGE_INHIBIT_QUERY function

The PURGE_INHIBIT_QUERY function of DSAT format is used by the dispatcher to see if a task selected for purge can be purged. Its main purpose is to find out from the AP domain whether the task is currently purgeable by the system.

**Input parameters**

**USER_TOKEN**
- is the token by which the task that has been dispatched is known to the called domain.

**TASK_TOKEN**
- is the token by which the task that has been dispatched is known to the dispatcher.

**Output parameters**

**PURGE_INHIBITED_RESPONSE**
- states whether the task can be purged. It can have either of these values:
  - YES
  - NO

**RESPONSE**
- always has the value OK.
Dispatcher domain (DS)

Modules

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHDSAT</td>
<td>Receives calls to the dispatcher DSAT gate. This gate carries out such work as:</td>
</tr>
<tr>
<td></td>
<td>ATTACH—Create new task</td>
</tr>
<tr>
<td></td>
<td>CHANGE_MODE—Change mode of running task</td>
</tr>
<tr>
<td></td>
<td>CHANGE_PRIORITY—Change priority of running task and release control</td>
</tr>
<tr>
<td></td>
<td>SET_PRIORITY—Change priority of running task or other task and keep running</td>
</tr>
<tr>
<td></td>
<td>CANCEL_TASK—Cancel specified task.</td>
</tr>
<tr>
<td>DFHDSBR</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>START_BROWSE</td>
</tr>
<tr>
<td></td>
<td>GET_NEXT</td>
</tr>
<tr>
<td></td>
<td>END_BROWSE</td>
</tr>
<tr>
<td></td>
<td>INQUIRE_TASK</td>
</tr>
<tr>
<td>DFHDSDM</td>
<td>Handles the following dispatcher requests:</td>
</tr>
<tr>
<td></td>
<td>DMDM PRE_INITIALISE</td>
</tr>
<tr>
<td></td>
<td>DMDM INITIALISE_DOMAIN</td>
</tr>
<tr>
<td></td>
<td>DMDM QUIESCE_DOMAIN</td>
</tr>
<tr>
<td></td>
<td>DMDM TERMINATE_DOMAIN</td>
</tr>
<tr>
<td>DFHDSIT</td>
<td>Handles the following dispatcher requests:</td>
</tr>
<tr>
<td></td>
<td>INQUIRE_DISPATCHER</td>
</tr>
<tr>
<td></td>
<td>SET_DISPATCHER</td>
</tr>
<tr>
<td>DFHDSKE</td>
<td>Handles kernel DS requirements, and handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>KEDS TCB_REPLY</td>
</tr>
<tr>
<td></td>
<td>KEDS TASK_REPLY</td>
</tr>
<tr>
<td>DFHDSMM</td>
<td>Receives the STORAGE_NOTIFY call from the storage manager domain.</td>
</tr>
<tr>
<td>DFHDSRR</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>ADD_SUSPEND</td>
</tr>
<tr>
<td></td>
<td>DELETE_SUSPEND</td>
</tr>
<tr>
<td></td>
<td>INQUIRE_SUSPEND_TOKEN</td>
</tr>
<tr>
<td></td>
<td>SUSPEND</td>
</tr>
<tr>
<td></td>
<td>RESUME</td>
</tr>
<tr>
<td></td>
<td>WAIT_MVS</td>
</tr>
<tr>
<td></td>
<td>WAIT_OLDW</td>
</tr>
<tr>
<td></td>
<td>WAIT_OLDDC</td>
</tr>
<tr>
<td>DFHDSST</td>
<td>Receives statistics calls from the ST domain</td>
</tr>
<tr>
<td>DFHDSUE</td>
<td>Receives the user exit gate call from the AP domain</td>
</tr>
</tbody>
</table>

Exits

There are two global user exit points in the dispatcher domain: XDSAWT and XDSBWCT. For further information about these, see the CICS Customization Guide.

Trace

The point IDs for the dispatcher domain are of the form DS xxxx; the corresponding trace levels are DS 1, DS 2, and Exc.

For more information about the trace points, see the CICS Trace Entries. For more information about using traces in problem determination, see the CICS Problem Determination Guide.
Chapter 24. Distributed program link

Distributed program link enables a program (the client program) in one CICS region to issue an EXEC CICS LINK command to link to a program (the server program) running in another CICS region (the resource region). The link can be through intermediate CICS regions.

The communication in distributed program link processing is, from the CICS side, synchronous, which means that it occurs during a single invocation of the client program, and that requests and replies between two programs can be directly correlated.

The CICS Intercommunication Guide includes information about distributed program link processing. Guidance information about designing and developing distributed applications is given in the CICS Distributed Transaction Programming Guide.

Figure 42 on page 328 gives an overview of distributed program link operation.
The DFHEIP module is described in "Chapter 35. EXEC interface" on page 511. This routes all program control requests to DFHEPC. DFHEPC passes all remote LINK requests to the program manager domain (PGLE_LINK_EXEC request). For local programs, program manager links to the program and, on return, it returns to DFHEPC. For remote programs, program manager returns to DFHEPC with an exception response, with a reason code indicating "remote program", and DFHEPC passes the request to the intersystems program, DFHISP. The operation of DFHISP for distributed program link is the same as for function shipping, but only the DFHXFP transformations are used. (See "Chapter 42. Function shipping" on page 673.) The operation of DFHPEP is described in "Chapter 64. Program control" on page 873; the interface to DFHPGLE LINK_EXEC is described in "Chapter 66. Program manager domain (PG)" on page 881.

Figure 42. Overview of program link
CICS handles session failures and systems failures for distributed program link processing by returning a TERMERR condition to the program that issued the LINK request.

If the server program terminates abnormally and does not handle the abend itself, DFHMIRS returns the abend code to the program that issued the LINK request. This code is the last abend code to occur in the server program, which may have handled other abends before terminating.

A client program using distributed program link can specify that a SYNCPOINT is to be taken in the resource region on successful completion of the server program. That is, any resources updated by the server program (or any associated program) are treated as if they are a separate unit of work.

Modules

The following modules are involved in the distributed program link:

- **DFHEIP**
  - EXEC interface (see “Chapter 35. EXEC interface” on page 511)
- **DFHEPC**
  - DFHEIP program control interface (see “Chapter 64. Program control” on page 873)
- **DFHISP**
  - ISC converse (see “Chapter 42. Function shipping” on page 673)
- **DFHMIRS**
  - Mirror transaction (see “Chapter 42. Function shipping” on page 673)
- **DFHPGLE**
  - PG domain - link exec function (see “Chapter 66. Program manager domain (PG)” on page 881)
- **DFHXFP**
  - Online data transformation program (see page 1565)

Exits

There are two global user exit points in DFHEPC: XPCREQ and XPCREQC.

Trace

No trace points are provided for this function.
Chapter 25. Distributed transaction processing

Distributed transaction processing enables a CICS transaction to communicate with a transaction running in another system. The transactions are designed and coded explicitly to communicate with each other, and thereby to use the intersystem link with maximum efficiency.

The communication in distributed transaction processing is, from the CICS side, synchronous, which means that it occurs during a single invocation of the CICS transaction and that requests and replies between two transactions can be directly correlated.

The CICS Intercommunication Guide tells you about multiregion operation and intersystem communication, and also includes some information about distributed transaction processing. Guidance information about designing and developing distributed applications is given in the CICS Distributed Transaction Programming Guide.

Design overview

CICS handles session failures and systems failures for distributed transaction processing in the same way as for CICS function shipping. See the relevant sections in "Chapter 42. Function shipping" on page 673 for further information.

Distributed transaction processing with MRO and LU6.1

Figure 43 gives an overview of the modules involved with distributed transaction processing for MRO and LU6.1 ISC.

```
  ┌───────────┐
  │ DFHEIP    │
  └─────┬─────┘
        MS'z6LLLL
          ┌───────────┐ ┌───────────┐
          │ BUILD_ATTACH │ DFHETC └──────┘ DFHZSTAP │
          │ EXTRACT      │              │
          │ POINT_TC     │              │
          └───────────┘              └───────────┘
                ┌─────┴─────┐ ┌─────┴─────┐
                │ DFHZARQ │ SEND │ DFHZISP │ FREE_TC
                │ SEND     │ WAIT │ ALLOCATE_TC │
                │ CONVERSE │      │         │
                │ RECEIVE (with journal) │
```

Figure 43. Distributed transaction processing for MRO and LU6.1

The DFHEIP module is described in "Chapter 35. EXEC interface" on page 511. This routes all terminal control requests to DFHETC. DFHETC handles BUILD_ATTACHMENT, EXTRACT, and POINT_TC requests itself, and routes all other requests to DFHZARQ except for FREE_TC and ALLOCATE_TC requests, which are routed to DFHZISP. If the request requires that the user conversation state be returned, DFHETC calls DFHZSTAP. All these modules are described in detail under "Modules" on page 333.
Distributed transaction processing

**Mapped and unmapped conversations (LU6.2)**

In **mapped** conversations, the data passed to and received from the LU6.2 application programming interface (API) is simply user data. Mapped conversations use the normal CICS API. Application programs and function shipping requests written for LU6.1 operate using mapped conversations when transferred to LU6.2.

*Figure 44* gives an overview of the modules involved with the processing of mapped conversations in LU6.2. ISC.

![Diagram of Distributed transaction processing for mapped conversations in LU6.2](image)

The DFHEIP module is described in "Chapter 35. EXEC interface" on page 511. This routes all terminal control requests to DFHETC. DFHETC routes all requests relating to an LU6.2 session to DFHETL except for ALLOCATE_TC requests, which are routed to DFHZISP.

In turn, DFHETL calls DFHZARL to process most requests; it calls DFHZISP to handle FREE_TC requests, and DFHZARM to handle the receipt of unrecognized or unsupported IDs. If the request requires that the user conversation state be returned, DFHETL calls DFHZSTAP.

DFHZARL’s processing depends on the type of request; for example, it calls DFHZISP to allocate a TCTTE, DFHZARR to receive data, and DFHZERH for outbound or inbound FMH7 processing. If the request needs to be transaction routed, DFHZARL calls DFHZXRL to route the request to the terminal-owning region (see "Chapter 102. Transaction routing" on page 1311).

With the exception of DFHZXRL, all these modules are described in detail under "Modules" on page 333.

**Unmapped** conversations (also known as **basic** conversations), are used principally for communication with device-level products that do not support mapped conversations, and which possibly do not have an API open to the user. In unmapped conversations, the data passed to and received from the LU6.2 API contains GDS headers.
Figure 45 gives an overview of the modules involved with the processing of unmapped conversations in LU6.2 ISC.

The DFHEIP module is described in "Chapter 35. EXEC interface" on page 511. This passes control to DFHEGL to process GDS commands. DFHEGL routes all GDS conversation-related commands directly to DFHZARL. Some validation of application-provided parameters is performed, and errors are reflected back to the application. If the request requires that the user conversation state be returned, DFHEGL calls DFHZSTAP.

DFHZARL’s processing depends on the type of request; for example, it calls DFHZISP to allocate a TCTTE, DFHZARR to receive data, and DFHZERH for outbound or inbound FMH7 processing. If the request needs to be transaction routed, DFHZARL calls DFHZXRL to route the request to the terminal-owning region (see "Chapter 102. Transaction routing" on page 1311).

With the exception of DFHZXRL, all these modules are described in detail in the next section.

**Modules**

**DFHEGL**

DFHEGL processes GDS commands. It is an EXEC interface processor module, and receives control directly from DFHEIP. The TCTTE for the session is located and checked for validity. All GDS conversation-related commands are mapped into a DFHLUC macro call and routed directly to DFHZARL. There is no mapping or unmapping of data, state indicators are not maintained, and there are no FMHs to process.

**DFHETC and DFHETL**

DFHEIP routes all terminal control requests to DFHETC (the EXEC interface processor for terminal control). DFHETC handles BUILD_ATTACH, EXTRACT, and POINT_TC requests itself. It routes all other requests relating to an MRO or LU6.1 session to DFHZARQ except for FREE_TC and ALLOCATE_TC requests, which are routed to DFHZISP. It routes all other requests relating to an LU6.2 session to DFHETL except for ALLOCATE_TC, which is routed to DFHZISP.
DFHETL performs the following actions:
1. Maps an application request into a form suitable for the DFHZCP and DFHZCC application request modules. This includes mapping application data into GDS records.
2. Detects errors and returns error codes to the application.
3. Unmaps data from GDS records.
4. Maintains state indicators.

For ISSUE CONFIRMATION, CONNECT PROCESS, EXTRACT PROCESS, ISSUE ERROR, ISSUE ABEND, and ISSUE SIGNAL commands, DFHETL:
1. Maps application requests into DFHLUC macro calls.
2. Updates state indicators in the TCTTE (for example, the TCTTE indicator that shows that a CONNECT PROCESS command has been issued).

For SEND and CONVERSE commands, DFHETL:
1. Obtains storage for the processing of outbound application data.
2. Creates attach FMHs, if appropriate.
3. Calls DFHZARL to transmit data.

For RECEIVE commands, DFHETL:
1. Obtains storage for the processing of inbound data.
2. Calls DFHZARL to receive inbound data.
3. Extracts inbound FMHs, as appropriate.
4. Unmaps inbound data.
5. Validates LLs and rejects them if not valid.
6. Manages the passing of data back to the application.
7. If the application issues a RECEIVE NOTRUNCATE request in order to receive only part of the chain, retains the residual data for subsequent RECEIVE requests. DFHETL receives one complete chain of data at a time from DFHZARL.

For WAIT commands, DFHETL calls DFHZARL.

For FREE commands, DFHETL:
1. Checks that the terminal is in the correct state to be freed.
2. Frees the storage used to hold RECEIVE data and the ETCB.
3. Calls DFHZISP to free the session.

**DFHZARL**

DFHZARL is always invoked via the DFHLUC macro. The DFHLUCDS DSECT maps a parameter list that is set up to pass information to and return information from DFHZARL. DFHZARL manages data in buffers, not in TIOAs. SEND commands cause data to be assembled by DFHZARL into a buffer until a WAIT, or other event, causes the data in the buffer to be transmitted.

DFHZARL invokes DFHZSDL to send data to VTAM, by placing requests on the activate chain. However, for optimization, DFHZARL can invoke DFHZSDL directly. Receive requests are handled by DFHZARR.

DFHZARL invokes DFHZUSR to manage the conversation state. The LU6.2 states for each session are stored in the TCTTE for that session.
If the request needs to be transaction routed, DFHZARL calls DFHZXRL to route the request to the terminal-owning region (see "Chapter 102. Transaction routing on page 1311").

Details of DFHZARL’s processing for the principal functions of the DFHLUC macro that is used to invoke DFHZARL are given below.

**INITIAL_CALL function**

This function is requested by DFHZSUP. DFHZARL acquires LU6.2 send and receive buffers. If the transaction is being started as a result of an ATTACH request received from a remote system, DFHZARL transfers any data received with the attach header from the TIOA into the receive buffer.

**ALLOCATE function**

DFHZARL performs the following actions:

1. If the request passed the address of a profile entry, puts this address in the TCA. If the request passed the name of a profile, calls transaction manager to locate the entry and then puts the address of the entry in the TCA.
2. If the request passed a netname rather than a specific sysid, calls DFHZLOC to locate the TCTTE for the netname and then puts the sysid into the DFHLUC parameter list (as if the caller had the specified sysid).
3. Copies the DFHLUC parameter list to LIFO storage.
4. Calls DFHZISP to allocate a TCTTE.
5. Addresses the TCTTE allocated.
6. Acquires LU6.2 send and receive buffers.
7. Sets the user state machine (DFHZUSRM), request = ALLOCATE_RESOURCE.
8. Returns results to the caller.

**SEND function**

DFHZARL performs the following actions:

1. Checks the user state machine (DFHZUSRM).
2. Checks the LL count and maintains a record of the outstanding LL count.
3. If the command is SEND LAST, INVITE, or CONFIRM, and the outstanding LL count is nonzero, issues an error message.
4. Sets the user state machine (DFHZUSRM).
5. Issues RECEIVE IMMEDIATE requests, as required, to pick up any negative responses sent by the partner program.

The caller must specify WAIT in the request to force the data to be sent immediately. SEND CONFIRM has an implicit WAIT, and control is not returned until a response has been received, when the state machine is set.

For a SEND request with WAIT, DFHZARL then:

1. Sets the user state machine (DFHZUSRM), request=WAIT.
2. Invokes DFHZSDL for transmission of the data in application area or send buffer.

For a SEND request without WAIT, DFHZARL then:

1. If there is sufficient space in the send buffer for all the data, transfers the data from the application area to the send buffer, and returns control to the caller.
2. Saves the INVITE and LAST indicators.
3. If the send buffer cannot hold all the data, invokes DFHZSDL for an implicit SEND.

If data or a CONFIRM command was sent (or both), DFHZARL then:
1. Checks for a signal received.
2. Checks for exception (negative) response received. If found, calls DFHZERH to handle the error. On return, sets the state machine.
3. Returns results to the caller.

When an implicit send is required, DFHZARL passes the data to DFHZSDL for transmission, passing the address of the data in the send buffer and in the application buffer. The total length of data passed to DFHZSDL is a multiple of the request unit size. On return to DFHZARL, the remaining data is transferred to the send buffer. The parameters passed to DFHZARL, such as INVITE and LAST, are not transmitted by DFHZSDL.

**RECEIVE function**
DFHZARL passes the DFHLUC parameter list, specifying the type of receive required, to DFHZARR for processing (see "DFHZARR" on page 338).

**ISSUE ERROR or ABEND function**
DFHZARL is called as a result of an ISSUE ERROR or ISSUE ABEND command, and performs the following actions:
1. Sets the user state machine
2. Calls DFHZERH.

**DFHZARM**

DFHETL may invoke DFHZARM to provide service functions. DFHZARQ passes control to DFHZARM instead of initiating DFHZSDS, DFHZRVS, and so on, if DFHZARQ finds that it is an LU6.2 session. This applies to the SEND, WAIT, RECEIVE, and SIGNAL commands. The same applies to DFHZISP for the FREE command.

DFHZARM translates the data stream to and from a format suitable for invoking DFHZARL. In particular:
- An LU6.2 attach FMH may have to be requested.
- Data must be passed in GDS record format (structured fields preceded by an LLID).

DFHZARM is invoked via the DFHLUCM macro, which has seven executable options:

```
DFHLUCM TYPE =
  - SEND
  - RECEIVE
  - WAIT
  - SIGNAL
  - FREE
  - INVALID_ID
```

DFHLUCM TYPE=STORAGE defines the storage in LIFO for passing primary input and output. The DSECT name is DFHLUMDS. TCTTE contains the secondary input and output. The principal functions are described in the following sections.
SEND function
DFHZARM performs the following actions:
1. Maps the data into GDS record format. The IDs used are:
   - X'12F1'
   - X'12F2'
   - X'12FF'.
2. Examines bits set in the TCTTE by DFHZARL to determine which DFC to apply.
3. Invokes DFHZARL (using a DFHLUC TYPE=SEND, LIST=... macro call) to pass the GDS records and DFC indicators.
4. Updates the state bits in TCTTE as necessary.
5. Interrogates the LU6.2 ATTACH_FMHBUILT bit in the TCTTE, which was set by DFHZSUP or DFHETL. This bit indicates whether this is first SEND. If an LU6.2 attach header has not already been built as a result of a CONNECT PROCESS command, DFHZARM issues CONNECT_PROCESS to DFHZARL, assuming syncllevel 2, before sending the data.

RECEIVE function
DFHZARM performs the following actions:
1. Calls DFHZARL using TYPE=BUFFER. Two calls are made. On the first call, the first 4 bytes (LLID) are retrieved into LIFO. These are examined and the LL is used to determine the TIOA size and to specify the length required in the second call.
2. On the second call, retrieves the remainder of the data directly into the TIOA. If the LL indicates concatenated data, a series of calls is made to retrieve all the data.

FREE function
The FREE function is used, for example, by DFHZISP to ensure that I/O has completed and CEB sent, using null data if necessary.

INVALID_ID function
The INVALID_ID function is used by DFHETL and DFHZARM itself. It handles the receipt of unrecognized or unsupported IDs. DFHZARM calls DFHZARL with ISSUE_ERROR (X'0889010x'), and sends a record with ID X'12F4' followed by the unrecognized ID. If the remote system responds, DFHZARM turns the flows around so that the local system can try again.

LU6.1 chains
An LU6.1 chain corresponds to one SEND command. LU6.2 chains are bigger, so:
- For outbound data, DFHZARM maps one SEND into one structured field (concatenated if necessary).
- For inbound data, DFHZARM retrieves one (possibly concatenated) field and calls it a chain, thus preserving compatibility.

DFHZARQ
DFHETC routes SEND, WAIT, CONVERSE, and some RECEIVE commands to DFHZARQ. RECEIVE commands are passed to DFHZARQ if input journaling is in effect. Otherwise, the call is routed to DFHZARL directly.

DFHZARQ passes control to DFHZARM instead of initiating DFHZSDS, DFHZRVS, and so on, if DFHZARQ finds that it is an LU6.2 session. This applies to the SEND, WAIT, RECEIVE, and SIGNAL commands.
Reasons for calling DFHZAQ are:

- To avoid duplication of existing code
- So that DFHZCP performs journaling of outbound data
- To perform an implicit CONNECT PROCESS if SEND or CONVERSE is the next session-related command after ALLOCATE
- To enable the SNA change direction (CD) and end bracket (EB) indicators to flow with the data.

DFHZARR

DFHZARR is called by DFHZARL to handle receive requests. Details of the processing follow.

**RECEIVE function**

This function must be able to handle receipt of the following:

- Application data
- FMH7s and ER1s (negative responses)
- PS_headers (Prepares, Request_commits)
- Indicators such as CD, CEB, and RQD2
- Signal.

Figure 46 gives an overview of the modules involved with the processing of receive requests. These modules are described in “Chapter 113. CICS executable modules” on page 1525.

DFHZARL passes the DFHLUC parameter list, specifying the type of receive required, to DFHZARR.

DFHZARR then performs the following actions:

1. Checks that request is valid; if not, returns error codes.
2. Initializes the application and LU6.2 receive buffers (by calls to DFHZARRA and the DFHZARR0 subroutine of DFHZARR respectively).
3. Calls DFHZARRC to determine what to process next.
4. Depending on DFHZARRC’s response, calls the relevant subroutine.

5. If “enough” (or all that can be) has not been received, loops back to step 3 on page 338; otherwise step 6.

6. Tests for (and returns) signal when it has been received.

The results of the receive are passed back to the caller in the DFHLUC parameter list.

To control this processing, DFHZARR uses the variables receive_type and what_next, as follows.

receive_type can have the following values:

- **RECEIVE_WAIT**
  Request was a receive and wait.

- **RECEIVE_IMMEDIATE**
  Request was a receive immediate.

- **LOOK_AHEAD**
  All the allowed user data has been received, but only one receive immediate call to the DFHZARR1 subroutine of DFHZARR is permitted to attempt to pick up indicators such as CD, CEB, or a PS_header.

- **NO_MORE_RECEIVES**
  No more calls to DFHZARR1 are permitted, but processing may continue with what has already been received.

- **NO_RECEIVE_LOOK_AHEAD**
  All the allowed user data has been received. An attempt must be made to pick up indicators such as CD, CEB, or a PS_header without a call to DFHZARR1. This value is only required for a receive immediate request.

- **RECEIVE_COMPLETE**
  Receive processing is finished.

The first two values are possible initial values of receive_type, and the other four are used as the receive progresses.

what_next is an output of DFHZARRC, and represents what is next to be processed. It can have the following values:

- **DATA_RECORD**
  Application data

- **FMH_RECORD**
  FMH7 in the buffer

- **PS_HEADER_RECORD**
  Prepare or Request_commit

- **PARTIAL_LL**
  First byte of a logical record only, therefore cannot tell whether it is a DATA_RECORD or PS_HEADER_RECORD

- **CD**
  Change Direction

- **CEB**
  Conditional End Bracket

- **RQD2**
  RQD2 without CD or CEB
Distributed transaction processing

RQD2_CD
RQD2 with CD

RQD2_CEB
RQD2 with CEB

ER1  Negative response

EMPTY_BUFFER
Nothing available to receive.

DFHZERH

DFHZERH is called by DFHZARL or DFHZARRF, when it is required to transmit error information or when error information has been received.

Outbound errors

For outbound errors, DFHZERH is invoked by DFHZARL following an ISSUE_ERROR, ISSUE_ABEND, or SYNC_ROLLBACK request.

An FMH7 must be transmitted, but can only be transmitted if the session is in the send state.

If the session is in the receive state, DFHZERH:
1. Sends a negative response
2. Purges the remaining data to end of chain.

In all cases, DFHZERH then:
1. Checks that the session is still in bracket
2. Clears the send buffers
3. Calls DFHZARL to send the FMH7.

Inbound errors

For inbound errors, DFHZERH is invoked by DFHZARL or DFHZARRF when a process-level exception response or an FMH7 has been received.

If an exception response is received while in the send state, DFHZERH purges the present output buffer and sends ‘LIC,CD,RQE1’ to put the conversation into receive state—so that the following FMH7 can be received.

If an FMH7 is received, DFHZERH examines the associated sense code and any GDS error log data, then returns to its caller.

DFHZISP

DFHZISP is called by DFHETC to perform ALLOCATE_TC requests. (ALLOCATE commands are passed to DFHZISP because DFHETC cannot check the session type until the session is allocated.)

DFHZISP is also called to perform FREE_TC requests.

DFHZSTAP

DFHZSTAP provides a means of determining the conversation state of an MRO or LU6.2 session from the application side. This function is required if the application issues an EXEC CICS EXTRACT ATTRIBUTES command with the STATE option, or a conversation-based command with the STATE option.
For MRO, modules that invoke MVS services via the DFHTC macro also update the conversation state information with a DFHZCNVM TYPE=PUT macro call. When an application requires the conversation state of a session, DFHETC calls DFHZSTAP using a DFHZSTAM TYPE=GETCURRSTATE macro, which returns a value representing the conversation state of the session.

For LU6.2, DFHZUSR is called to maintain the user conversation state machine. (See Chapter 107. VTAM LU6.2 on page 1373 for further details.) When an application requires the conversation state of a session, DFHETL (mapped) or DFHEGL (unmapped) calls DFHZSTAP using a DFHZSTAM TYPE=GETCURRSTATE macro. DFHZSTAP examines the DFHZUSR state machine and maps the information into a value representing the conversation state of the session.

**Exits**

No global user exit points are provided for this function.

**Trace**

The following point IDs are provided for distributed transaction processing:

- AP FDxx, for which the trace level is TC 1
- AP FExx (LU6.2 application receive requests), for which the trace levels are TC 2 and Exc.

For more information about the trace points, see the CICS Trace Entries. For more information about using traces in problem determination, see the CICS Problem Determination Guide.
Chapter 26. DL/I database support

Facilities for accessing DL/I databases and database control (DBCTL) support are available only with IMS/ESA.

Within a single CICS system, the following types of support can be available:

- DBCTL support present. For specific information about DBCTL, see "Chapter 20. Database control (DBCTL)" on page 281
- Remote DL/I and DBCTL support present (the PDIR system initialization parameter is specified). For specific information about remote DL/I, see "Chapter 70. Remote DL/I" on page 975

The rest of this section covers DL/I database support in general.

Design overview

The following types of DL/I requests can be made by a CICS system:

- EXEC DLI statements (converted into standard CALL DLI statements by DFHEDP)
- CALL DLI statements.

CICS support for DL/I is provided as follows:

1. A router component
   This component determines whether the call is using a remote or DBCTL PSB, and passes control to the appropriate call processor. This component is described in more detail later in this section.

2. A DL/I call processor
   This component is subdivided into:
   - A remote DL/I call processor
   - A DBCTL DL/I call processor.

   Each call processor deals with a specific interface that is described in the appropriate section of this book for the remote DL/I function and the DBCTL function.

Figure 47 on page 344 shows the relationships between the components of the CICS-DL/I interface.
DL/I database support

Notes:
1. When DL/I functions are requested by an application program or a CICS control module through execution of a CALL or CALLDLI macro, DFHDLI sets the required fields in the TCA. EXEC DLI statements are converted into standard CALL DLI statements by DFHEDP.
   If the request is for a remote database, DFHDLI passes control to DFHDLIRP. If the request is for a DBCTL database, DFHDLI passes control to DFHDLIDP.
   In addition to processing DL/I input/output requests, the DL/I interface, on request, schedules and terminates DL/I program specification blocks (PSBs).

The remainder of this section is concerned with the router component.

The router component (DFHDLI)

The router component receives a request in standard CALL DLI parameter lists. At schedule time, it determines whether the request is a remote or DBCTL request.

Amongst the functions of the router are the following:

Deciding where to process a request
At PSB schedule time, the router determines whether the DL/I requests issued from the application program should be routed to DBCTL or another CICS system (remote). The presence (or absence) of the PSB used in the PDIR determines where the call gets routed.

If no PDIR exists (that is, the PDIR=NO system initialization parameter is specified or is allowed to default), the request is routed to the DBCTL call processor.

If a PDIR has been specified, the router module scans the PDIR. All entries in the PDIR have a SYSIDNT option specified. If the PSB is not found in the PDIR, or if the PDIR entry specifies a SYSIDNT that is the SYSIDNT of the CICS system that is currently running, the request is routed to the DBCTL call processor. Otherwise, the request is routed to the remote call processor.

All DL/I requests are routed to the same DL/I call processor as the corresponding PSB schedule request in the same unit of work.
Initiating synchronization processing
The router provides special handling of the DL/I TERM call. When the router detects a TERM call, it forces a syncpoint, causing CICS to carry out syncpoint processing for the task.

Generating CICS trace records
The router module generates CICS trace records at DL/I call entry and DL/I call exit.

Control blocks
DL/I database support uses the control blocks DIB, DLP, and UIB, which are shown in Figure 48.

![Diagram of control blocks for DL/I database support]

DL/I interface block (DIB)
When an application program issues EXEC DLI requests, it uses the user DL/I interface block (DIB) instead of the user interface block (UIB). On return, DFHEDP extracts data from the UIB to place in the DIB. The storage for the user DIB is part of the application program. The definition of the user DIB is automatically inserted by the CICS translator for an EXEC DLI application program.

DL/I interface parameter list (DLP)
The DL/I interface parameter list (DLP) is a global DL/I interface control block that lasts for the duration of a CICS session, and contains information relating to
the type of DL/I support present in the CICS system. The DLP is created during CICS startup and is addressed by CSADLI in the CSA optional features list.

Figure 49 illustrates the DL/I parameter interface list (DLP) that is used.

User interface block (UIB)

The user interface block (UIB) is the control block used by the CALL and CALL DL/I interfaces to pass response codes and the PCB address list to application programs using CALL DL/I services. The UIB is acquired when a task issues its first PSB schedule request specifying that it requires a UIB. The UIB is freed at task termination. TCADLIBA points to the UIB.

See the CICS Data Areas manual for a detailed description of these control blocks.

Modules

Figure 50 on page 347 shows the module flow of DL/I requests to the DL/I call processors. Three main CICS-DL/I interface modules process DL/I requests from application programs. The first module, DFHDLI, determines what sort of DL/I request is being made and then passes control to one of two call processors. These are the DBCTL DL/I call processor, DFHDLIDP, and the remote call processor, DFHDLIRP.
The common CICS-DL/I interface modules consist of the following:

- **DFHDLI**—contains the code for routing requests to DFHDLIRP and DFHDLIDP
- **DFHDLIDP**—contains the code for DBCTL requests.
- **DFHDLIRP**—contains the code for remote DL/I requests

### Exits

The following global user exit points are provided in DFHDLI: XDLIPRE and XDLIPOST. For further information about these, see the [CICS Customization Guide](#) and the [CICS IMS Database Control Guide](#).

### Trace

The following point ID is provided for DL/I and DBCTL:

- AP 03xx, for which the trace levels are FC 1, FC 2, and Exc.

For more information about the trace points, see the [CICS Trace Entries](#). For more information about using traces in problem determination, see the [CICS Problem Determination Guide](#).
Chapter 27. Document Handler domain (DH)

The document handler domain manages CICS Documents.

Document Handler domain’s specific gates

Table 41 summarizes the document handler domain’s specific gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and whether or not the functions are available through the exit programming interface (XPI).

Table 41. Document Handler domain’s specific gates

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>XPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>DHDH</td>
<td>DH 0120</td>
<td>CREATE_DOCUMENT</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>DH 0121</td>
<td>INSERT_DATA</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>DH 0122</td>
<td>INSERT_BOOKMARK</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>DH 0123</td>
<td>REPLACE_DATA</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>DH 0124</td>
<td>DELETE_DOCUMENT</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>DH 0125</td>
<td>DELETE_DATA</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>DH 0126</td>
<td>DELETE_BOOKMARK</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>DH 0127</td>
<td>RETRIEVE_WITH_CTLINFO</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>DH 0128</td>
<td>RETRIEVE_WITHOUT_CTLINFO</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>DH 0129</td>
<td>INQUIRE_DOCUMENT</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>DH 012A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DHSI</td>
<td>DH 0200</td>
<td>SET_SYMBOL_VALUE_BY_API,</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>DH 0201</td>
<td>SET_SYMBOL_VALUE_BY_SSI,</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>DH 0202</td>
<td>ADD_SYMBOL_LIST</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>DH 0203</td>
<td>EXPORT_SYMBOL_LIST</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>DH 0204</td>
<td>IMPORT_SYMBOL_LIST</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>DH 0205</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DH 0206</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DH 0207</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DH 0208</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DH 0209</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DH 020A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DHTM</td>
<td>DH 0401</td>
<td>INITIALIZE_DOCTEMPLATES</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>DH 0402</td>
<td>ADD_REPLACE_DOCTEMPLATE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>DH 0403</td>
<td>DELETE_DOCTEMPLATE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>DH 0404</td>
<td>INQUIRE_DOCTEMPLATE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>DH 0405</td>
<td>INQUIRE_TEMPLATE_STATUS</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>DH 0406</td>
<td>START_BROWSE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>DH 0407</td>
<td>GET_NEXT</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>DH 0408</td>
<td>END_BROWSE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>DH 0409</td>
<td>READ_TEMPLATE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>DH 040A</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DH 0411</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DH 0412</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DH 0413</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DH 0414</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DH 0415</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DH 0416</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DH 0417</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DH 0418</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 41. Document Handler domain’s specific gates (continued)

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>XPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>DHRP</td>
<td>0C01</td>
<td>RECOVER_DEFINITIONS</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>0C02</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0C03</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0C04</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0C05</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0C06</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0C07</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0C08</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**DHDH gate, CREATE_DOCUMENT function**

The CREATE_DOCUMENT function of the DHDH gate is used to create a new CICS document.

**Input parameters**

- **[TEXT]**
  - is a buffer containing a block of text to be added to the document.

- **[BINARY]**
  - is a buffer containing a block of binary data to be added to the document.

- **[TEMPLATE_BUFFER]**
  - is a buffer containing a template to be added to the document.

- **[TEMPLATE_NAME]**
  - is the name of an RDO defined DOCTEMPLATE which is to be added to the document.

- **[SOURCE_DOCUMENT]**
  - is the document token of an existing document created by the same CICS task which is to be added to the document.

- **[RETRIEVED_DOCUMENT]**
  - is a buffer containing a document in a retrieved format which is to be added to the document.

- **[HOST_CODEPAGE]**
  - is the character encoding for the block of data being added to the document. This parameter is taken into account for the TEXT and TEMPLATE_BUFFER options and ignored for all other options.

- **[SYMBOL_LIST]**
  - is a buffer containing a list of symbols to be added to the symbol table of the document.

- **[TEMPLATE_IN_ERROR]**
  - is a buffer which is used by the Document Handler domain to return the name of a DOCTEMPLATE in which an error has been detected. This parameter is only meaningful when specified with the TEMPLATE_NAME option or the TEMPLATE_BUFFER option where the template in the TEMPLATE_BUFFER option contains an embedded template.

**Output parameters**

- **DOCUMENT_TOKEN**
  - is the token identifying the newly created document.
ERROR_OFFSET
is the offset into a template where a syntax error has been detected.

RETRIEVE_SIZE
is the maximum size in bytes that a retrieved copy of the document can be.

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>CODEPAGE_NOT_SPECIFIED INVALID_HOST_CODEPAGE</td>
</tr>
<tr>
<td></td>
<td>INVALID_TEMPLATE_SYNTAX TEMPLATE_NOT_FOUND</td>
</tr>
<tr>
<td></td>
<td>SOURCE_DOC_NOT_FOUND INVALID_RETRIEVE_FORMAT</td>
</tr>
<tr>
<td></td>
<td>SYMBOL_NAME_INVALID SYMBOL_VALUE_INVALID</td>
</tr>
<tr>
<td></td>
<td>EMBED_DEPTH_EXCEEDED INVALID_TEMPLATE_LENGTH</td>
</tr>
</tbody>
</table>

DHDH gate, INSERT_DATA function

The INSERT_DATA function of the DHDH gate is used to insert a block of data into an existing document.

Input parameters

DOCUMENT_TOKEN
is the token which identifies the document into which the data will be inserted.

[TEXT]
is a buffer containing a block of text to be added to the document.

[BINARY]
is a buffer containing a block of binary data to be added to the document.

[TEMPLATE_BUFFER]
is a buffer containing a template to be added to the document.

[TEMPLATE_NAME]
is the name of an RDO defined DOCTEMPLATE which is to be added to the document.

[SOURCEDOCUMENT]
is the name of a symbol defined in the symbol table. The value associated with the symbol will be added to the document.

[SOURCE_DOCUMENT]
is the document token of an existing document created by the same CICS task which is to be added to the document.

[RETRIEVED_DOCUMENT]
is a buffer containing a document in a retrieved format which is to be added to the document.

[HOST_CODEPAGE]
is the character encoding for the block of data being added to the document. This parameter is taken into account for the TEXT, SYMBOL and TEMPLATE_BUFFER options and ignored for all other options.
Document Handler domain (DH)

[INSERT_POINT] identifies the beginning or end as the position at which data should be inserted into a document. It can have either of these values:
START | END

[INSERT_AT] is the name of a bookmark which identifies the position at which the data should be inserted.

[TEMPLATE_IN_ERROR] is a buffer which is used by the Document Handler domain to return the name of a DOCTEMPLATE in which an error has been detected. This parameter is only meaningful when specified with the TEMPLATE_NAME option or the TEMPLATE_BUFFER option where the template in the TEMPLATE_BUFFER option contains an embedded template.

Output parameters

ERROR_OFFSET is the offset into a template where a syntax error has been detected.

RETRIEVE_SIZE is the maximum size in bytes that a retrieved copy of the document can be.

RESPONSE is the domain’s response to the call. It can have any of these values:
OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON] is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>CODEPAGE_NOT_SPECIFIED INVALID_HOST_CODEPAGE</td>
</tr>
<tr>
<td></td>
<td>EMBED_DEPTH_EXCEEDED INVALID_POINT_NOT_FOUND</td>
</tr>
<tr>
<td></td>
<td>INVALID_TEMPLATE_SYNTAX TEMPLATE_NOT_FOUND</td>
</tr>
<tr>
<td></td>
<td>SOURCE_DOC_NOT_FOUND INVALID_RETRIEVE_FORMAT</td>
</tr>
<tr>
<td></td>
<td>SYMBOL_NOT_FOUND INVALID_TEMPLATE_LENGTH</td>
</tr>
<tr>
<td>INVALID</td>
<td>DOCUMENT_NOT_FOUND</td>
</tr>
</tbody>
</table>

DHDH gate, INSERT_BOOKMARK function

The INSERT_BOOKMARK function of the DHDH gate is used to insert a bookmark into an existing document.

Input parameters

DOCUMENT_TOKEN is the token which identifies the document into which the bookmark will be inserted.

BOOKMARK_NAME is the 16 byte name of a bookmark to be added to the document.

[INSERT_POINT] identifies the beginning or end as the position at which the bookmark should be inserted into a document. It can have either of these values:
START | END
[INSERT_AT]
is the name of a bookmark which identifies the position at which the bookmark should be inserted.

**Output parameters**

**RETRIEVE_SIZE**
is the maximum size in bytes that a retrieved copy of the document can be.

**RESPONSE**
is the domain’s response to the call. It can have any of these values:
\[ \text{OK} | \text{EXCEPTION} | \text{DISASTER} | \text{INVALID} | \text{KERNERROR} | \text{PURGED} \]

**[REASON]**
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>INSERTPOINT_NOT_FOUND INVALID_BOOKMARK_NAME</td>
</tr>
<tr>
<td></td>
<td>DUPLICATE_BOOKMARK</td>
</tr>
<tr>
<td>INVALID</td>
<td>DOCUMENT_NOT_FOUND</td>
</tr>
</tbody>
</table>

**DHDH gate, REPLACE_DATA function**

The REPLACE_DATA function of the DHDH gate is used to replace the data between 2 bookmarks in an existing document.

**Input parameters**

**DOCUMENT_TOKEN**
is the token which identifies the document into which the data will be inserted.

**[TEXT]**
is a buffer containing a block of text to be added to the document.

**[BINARY]**
is a buffer containing a block of binary data to be added to the document.

**[TEMPLATE_BUFFER]**
is a buffer containing a template to be added to the document.

**[TEMPLATE_NAME]**
is the name of an RDO defined DOCTEMPLATE which is to be added to the document.

**[SYMBOL]**
is the name of a symbol defined in the symbol table. The value associated with the symbol will be added to the document.

**[SOURCE_DOCUMENT]**
is the document token of an existing document created by the same CICS task which is to be added to the document.

**[RETRIEVED_DOCUMENT]**
is a buffer containing a document in a retrieved format which is to be added to the document.

**[HOST_CODEPAGE]**
is the character encoding for the block of data being added to the document. This parameter is taken into account for the TEXT, SYMBOL and TEMPLATE_BUFFER options and ignored for all other options.
Document Handler domain (DH)

[FROM_POSITION]
identifies the beginning or end of the document as the start of the data which is to be replaced in the document. It can have either of these values:
START|END

[FROM_BOOKMARK]
is the name of a bookmark which identifies the start of the data which is to be replaced.

[TO_POSITION]
identifies the beginning or end of the document as the end of the data which is to be replaced in the document. It can have either of these values:
START|END

[TO_BOOKMARK]
is the name of a bookmark which identifies the end of the data which is to be replaced.

[TEMPLATE_IN_ERROR]
is a buffer which is used by the Document Handler domain to return the name of a DOCTEMPLATE in which an error has been detected. This parameter is only meaningful when specified with the TEMPLATE_NAME option or the TEMPLATE_BUFFER option where the template in the TEMPLATE_BUFFER option contains an embedded template.

Output parameters
ERROR_OFFSET
is the offset into a template where a syntax error has been detected.

RETRIEVE_SIZE
is the maximum size in bytes that a retrieved copy of the document can be.

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>CODEPAGE_NOT_SPECIFIED INVALID_HOST_CODEPAGE EMBED_DEPTH_EXCEEDED</td>
</tr>
<tr>
<td></td>
<td>INVALID_TEMPLATE_SYNTAX TEMPLATE_NOT_FOUND SOURCE_DOC_NOT_FOUND</td>
</tr>
<tr>
<td></td>
<td>INVALID RETRIEVE_FORMAT SYMBOL_NOT_FOUND FROM_BOOKMARK_NOT_FOUND</td>
</tr>
<tr>
<td></td>
<td>TO_BOOKMARK_NOT_FOUND INVALID TEMPLATE_LENGTH</td>
</tr>
<tr>
<td>INVALID</td>
<td>DOCUMENT NOT FOUND</td>
</tr>
</tbody>
</table>

DHDH gate, DELETE_DOCUMENT function

The DELETE_DOCUMENT function of the DHDH gate is used to delete a document.

Input parameters
DOCUMENT_TOKEN
is the token which identifies the document to be deleted.
Output parameters

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>DOCUMENT_NOT_FOUND</td>
</tr>
</tbody>
</table>

DHDH gate, DELETE_DATA function

The DELETE_DATA function of the DHDH gate is used to delete the data between 2 bookmarks in an existing document.

Input parameters

DOCUMENT_TOKEN
is the token which identifies the document from which the data will be deleted.

[FROM_POSITION]
identifies the beginning or end of the document as the start of the data which is to be deleted from the document. It can have either of these values:
START|END

[FROM_BOOKMARK]
is the name of a bookmark which identifies the start of the data which is to be deleted.

[TO_POSITION]
identifies the beginning or end of the document as the end of the data which is to be deleted from the document. It can have either of these values:
START|END

[TO_BOOKMARK]
is the name of a bookmark which identifies the end of the data which is to be deleted.

Output parameters

RETRIEVE_SIZE
is the maximum size in bytes that a retrieved copy of the document can be.

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>FROM_BOOKMARK_NOT_FOUND</td>
</tr>
<tr>
<td></td>
<td>TO_BOOKMARK_NOT_FOUND</td>
</tr>
<tr>
<td></td>
<td>INVALID_BOOKMARK_SEQUENCE</td>
</tr>
</tbody>
</table>
DHDH gate, DELETE_BOOKMARK function

The DELETE_BOOKMARK function of the DHDH gate is used to delete a bookmark in an existing document.

Input parameters

**DOCUMENT_TOKEN**  
is the token which identifies the document from which the bookmark will be deleted.

**BOOKMARK_NAME**  
is the name of the bookmark to be deleted from the document.

Output parameters

**RETRIEVE_SIZE**  
is the maximum size in bytes that a retrieved copy of the document can be.

**RESPONSE**  
is the domain’s response to the call. It can have any of these values:  
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]  
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>BOOKMARK_NOT_FOUND</td>
</tr>
<tr>
<td>INVALID</td>
<td>DOCUMENT_NOT_FOUND</td>
</tr>
</tbody>
</table>

DHDH gate, RETRIEVE_WITH_CTLINFO function

The RETRIEVE_WITH_CTLINFO function of the DHDH gate is used to retrieve a copy of an existing document. The retrieved copy will contain embedded control information.

Input parameters

**DOCUMENT_TOKEN**  
is the token which identifies the document to be retrieved.

**DOCUMENT_BUFFER**  
is a buffer into which the Document Handler domain will place the copy of the document.

Output parameters

**RESPONSE**  
is the domain’s response to the call. It can have any of these values:  
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]  
is returned when RESPONSE is INVALID. Possible values are:
DHDH gate, RETRIEVE_WITHOUT_CTLINFO function

The RETRIEVE_WITHOUT_CTLINFO function of the DHDH gate is used to retrieve a copy of an existing document. The retrieved copy will only contain the data in the document.

Input parameters

DOCUMENT_TOKEN
is the token which identifies the document to be retrieved.

DOCUMENT_BUFFER
is a buffer into which the Document Handler domain will place the copy of the document.

[CLIENT_CODEPAGE]
is the character encoding that the retrieved document should be converted to when it is placed in the buffer.

Output parameters

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>INVALID_HOST_CODEPAGE INVALID_CLIENT_CODEPAGE</td>
</tr>
<tr>
<td>INVALID</td>
<td>DOCUMENT_NOT_FOUND</td>
</tr>
</tbody>
</table>

DHDH gate, INQUIRE_DOCUMENT function

The INQUIRE_DOCUMENT function of the DHDH gate is used to obtain information about the document.

Input parameters

DOCUMENT_TOKEN
is the token which identifies the document to be queried.

Output parameters

[DOCUMENT_SIZE]
is the size of the data in a document.

[RETRIEVE_SIZE]
is the maximum size in bytes that a retrieved copy of the document can be.

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is INVALID. Possible values are:
**DHSL gate, SET_SYMBOL_VALUE_BY_API function**

The SET_SYMBOL_VALUE_BY_API function of the DHSL gate is used to set the value of a symbol in the symbol table. If the symbol does not exist in the table, it will be added. If the symbol does exist in the table, it will always be replaced.

**Input parameters**

- **DOCUMENT_TOKEN**
  is the token which identifies the document that owns the symbol table.

- **SYMBOL_NAME**
  is the name of the symbol in the symbol table.

- **VALUE**
  is the value to be associated with the symbol.

**Output parameters**

- **RESPONSE**
  is the domain’s response to the call. It can have any of these values:

  OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

- **[REASON]**
  is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>GETMAIN_ERROR SYMBOL_NAME_INVALID</td>
</tr>
<tr>
<td>INVALID</td>
<td>DOCUMENT_NOT_FOUND</td>
</tr>
</tbody>
</table>

**DHSL gate, SET_SYMBOL_VALUE_BY_SSI function**

The SET_SYMBOL_VALUE_BY_SSI function of the DHSL gate is used to set the value of a symbol in the symbol table. If the symbol does not exist in the table, it will be added. If the symbol does exist in the table, it will only be replaced if it was previously set using the SET_SYMBOL_VALUE_BY_SSI function.

**Input parameters**

- **DOCUMENT_TOKEN**
  is the token which identifies the document that owns the symbol table.

- **SYMBOL_NAME**
  is the name of the symbol in the symbol table.

- **VALUE**
  is the value to be associated with the symbol.

**Output parameters**

- **RESPONSE**
  is the domain’s response to the call. It can have any of these values:

  OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

- **[REASON]**
  is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>GETMAIN_ERROR SYMBOL_NAME_INVALID</td>
</tr>
<tr>
<td>INVALID</td>
<td>DOCUMENT_NOT_FOUND</td>
</tr>
</tbody>
</table>
are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>GETMAIN_ERROR SYMBOL_NAME_INVALID</td>
</tr>
<tr>
<td>INVALID</td>
<td>DOCUMENT_NOT_FOUND</td>
</tr>
</tbody>
</table>

**DHSL gate, ADD_SYMBOL_LIST function**

The ADD_SYMBOL_LIST function of the DHSL gate is used to add a list of symbols to the symbol table at one time.

**Input parameters**

**DOCUMENT_TOKEN**

is the token which identifies the document that owns the symbol table.

**SYMBOL_LIST**

is a buffer containing a list of symbols to be added to the symbol table of the document.

**Output parameters**

**ERROR_OFFSET**

is the offset into the symbol list where a syntax error has been detected.

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**[REASON]**

is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>GETMAIN_ERROR SYMBOL_NAME_INVALID</td>
</tr>
<tr>
<td></td>
<td>SYMBOL_VALUE_INVALID</td>
</tr>
<tr>
<td>INVALID</td>
<td>DOCUMENT_NOT_FOUND</td>
</tr>
</tbody>
</table>

**DHSL gate, EXPORT_SYMBOL_LIST function**

The EXPORT_SYMBOL_LIST function of the DHSL gate is used to export all the symbols in the symbol table in a form that can be re-imported with IMPORT_SYMBOL_LIST.

**Input parameters**

**DOCUMENT_TOKEN**

is the token which identifies the document that owns the symbol table.

**SYMBOL_LIST_BUFFER**

is a buffer that is to contain the exported symbol list.

**Output parameters**

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**[REASON]**

is returned when RESPONSE is INVALID. Possible values are:
**Document Handler domain (DH)**

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>DOCUMENT_NOT_FOUND</td>
</tr>
</tbody>
</table>

**DHSGL gate, IMPORT_SYMBOL_LIST function**

The IMPORT_SYMBOL_LIST function of the DHSGL gate is used to import all the symbols in the symbol table that were exported with EXPORT_SYMBOL_LIST.

**Input parameters**

- **DOCUMENT_TOKEN**
  - is the token which identifies the document that owns the symbol table.

- **SYMBOL_LIST_BUFFER**
  - is a buffer that contains the symbol list to be added to the symbol table.
  - This list should have been created using and the EXPORT_SYMBOL_LIST function.

**Output parameters**

- **ERROR_OFFSET**
  - is the offset into the list where a syntax error has been detected.

- **RESPONSE**
  - is the domain’s response to the call. It can have any of these values:
    - OK
    - EXCEPTION
    - DISASTER
    - INVALID
    - KERNERROR
    - PURGED

- **[REASON]**
  - is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:
    - SYMBOL_NAME_INVALID
    - SYMBOL_VALUE_INVALID
    - DOCUMENT_NOT_FOUND

**DHTM gate, INITIALIZE_DOCTEMPLATES function**

The INITIALIZE_DOCTEMPLATES function of the DHTM gate is used to initialize the state required by the template manager.

**Output parameters**

- **RESPONSE**
  - is the domain’s response to the call. It can have any of these values:
    - OK
    - EXCEPTION
    - DISASTER
    - INVALID
    - KERNERROR
    - PURGED

- **[REASON]**
  - is returned when RESPONSE is DISASTER. Possible values are:
    - DIRECTORY_ERROR

**DHTM gate, ADD_REPLACE_DOCTEMPLATE function**

The ADD_REPLACE_DOCTEMPLATE function of the DHTM gate is used to install a document template into the currently executing CICS system.
Input parameters

**DOCTEMPLATE**

is the name of the DOCTEMPLATE resource that is to be added.

**TEMPLATE_NAME**

is the name by which the DOCTEMPLATE is known outside of RDO.

**RESOURCE_TYPE**

specifies the type of resource containing the DOCTEMPLATE. It can have one of the following values:

- PDS_MEMBER
- FILE
- PROGRAM
- TSQUEUE
- TDQUEUE
- EXITPGM

**RESOURCE_NAME**

is the name of the resource containing the DOCTEMPLATE.

**[DDNAME]**

is the DDNAME of the PDS containing the DOCTEMPLATE resource if the resource resides on a PDS.

Output parameters

**[DATASET]**

is the dataset name of the PDS containing the DOCTEMPLATE resource if the resource resides on a PDS.

**[DOCTEMPLATE_IN_USE]**

is the name of the DOCTEMPLATE definition that uses the same TEMPLATE_NAME as the resource being defined.

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

- OK
- EXCEPTION
- DISASTER
- INVALID
- KERNERROR
- PURGED

**[REASON]**

is returned when RESPONSE is INVALID, DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>INVALIDRESOURCE_TYPE</td>
</tr>
<tr>
<td>DISASTER</td>
<td>DIRECTORY_ERROR</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>GETMAIN_FAILED NAME_IN_USE NOT_FOUND DDNAME_NOT_FOUND MEMBER_NOT_FOUND</td>
</tr>
</tbody>
</table>

DHTM gate, READ_TEMPLATE function

The READ_TEMPLATE function of the DHTM gate is used to read a named template into a buffer provided by the caller.

Input parameters

**TEMPLATE_NAME**

is the name of a previously installed document template.

**TEMPLATE_BUFFER**

is the buffer into which the template is to be read.

Output parameters

**[DOCTEMPLATE]**

is the name of the DOCTEMPLATE resource as it is known to RDO.
Document Handler domain (DH)

**RESPONSE**
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>DIRECTORY_ERROR INVALIDRESOURCE_TYPE</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>NOT_FOUND NOT_USABLE TRUNCATED</td>
</tr>
</tbody>
</table>

DHTM gate, **INQUIRE_DOCTEMPLATE** function
The INQUIRE_DOCTEMPLATE function of the DHTM gate returns information about a previously installed document template.

**Input parameters**
DOCTEMPLATE
is the name of the DOCTEMPLATE as known to RDO.

**Output parameters**
TEMPLATE_NAME
is the full name of the template as known outside RDO.

RESOURCE_TYPE
is the CICS or non-CICS resource type associated with the template. It can have one of the following values:
PDS_MEMBER|FILE|PROGRAM|TSQUEUE|TDQUEUE|EXITPGM

RESOURCE_NAME
is the name of the CICS or non-CICS resource.

DATASET
is the dataset name of the template PDS if the RESOURCE_TYPE indicates a PDS.

DDNAME
is the DDNAME of the template PDS if the RESOURCE_TYPE indicates a PDS.

**RESPONSE**
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>DIRECTORY_ERROR</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>NOT_FOUND</td>
</tr>
</tbody>
</table>

DHTM gate, **INQUIRE_TEMPLATE_STATUS** function
The INQUIRE_TEMPLATE_STATUS function of the DHTM gate is used to inquire the install status of one or more templates.
Input parameters

TEMPLATE_NAME_LIST
is a list of template names whose install status is sought.

TEMPLATE_STATUS_LIST
is a list of install status indicators for the templates named in the
TEMPLATE_NAME_LIST

Output parameters

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

DHTM gate, DELETE_DOCTYPE function

The DELETE_DOCTYPE function of the DHTM gate deletes a previously
installed DOCTYPE.

Input parameters

DOCTYPE
is the name of the DOCTYPE as known to RDO.

Output parameters

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>DIRECTORY_ERROR</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>NOT_FOUND</td>
</tr>
</tbody>
</table>

DHTM gate, START_BROWSE function

The START_BROWSE function of the DHTM gate is used to initiate a browse of
installed DOCTYPE definitions.

Output parameters

BROWSE_TOKEN
is a token identifying this DOCTYPE browse.

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

DHTM gate, GET_NEXT function

The GET_NEXT function of the DHTM gate returns information about the next
installed DOCTYPE in the browse.
Input parameters

BROWSE_TOKEN

is the token identifying this browse of the DOCTEMPLATE definitions.

Output parameters

DOCTEMPLATE

is the name of the DOCTEMPLATE as known to RDO.

TEMPLATE_NAME

is the full name of the template as known outside RDO.

RESOURCE_TYPE

is the CICS or non-CICS resource type associated with the template. It can have one of the following values:

PDS_MEMBER|FILE|PROGRAM|TSQUEUE|TDQUEUE|EXITPGM

RESOURCE_NAME

is the name of the CICS or non-CICS resource.

DATASET

is the dataset name of the template PDS if the RESOURCE_TYPE indicates a PDS.

DDNAME

is the DDNAME of the template PDS if the RESOURCE_TYPE indicates a PDS.

RESPONSE

is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]

is returned when RESPONSE is INVALID or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>INVALID_BROWSE_TOKEN</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>BROWSE_END</td>
</tr>
</tbody>
</table>

DHTM gate, END_BROWSE function

The END_BROWSE function of the DHTM gate is used to terminate a browse of installed DOCTEMPLATE definitions.

Input parameters

BROWSE_TOKEN

is the token identifying this browse of the DOCTEMPLATE definitions.

Output parameters

RESPONSE

is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]

is returned when RESPONSE is INVALID. Possible values are:
RESPONSE | Possible REASON values
---|---
INVALID | INVALID_BROWSE_TOKEN

**DHRP gate, RECOVER_DEFINITIONS function**

The RECOVER_DEFINITIONS function of the DHRP gate is used to purge/recover DOCTEMPLATE definitions from the global catalog depending upon the CICS start type.

**Output parameters**

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

0K | DISASTER

**[REASON]**

is returned when RESPONSE is DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>INVALID_BROWSE_TOKEN CATALOG_BROWSE_FAILURE CATALOG_PURGE_FAILURE LOGIC_ERROR WAIT_PHASE_FAILURE ABEND</td>
</tr>
</tbody>
</table>

**Document Handler domain’s generic gates**

Table 42 summarizes the document handler domain’s generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMDM</td>
<td>DH 0101</td>
<td>INITIALIZE_DOMAIN</td>
<td>DMDM</td>
</tr>
<tr>
<td></td>
<td>DH 0102</td>
<td>QUIESCE_DOMAIN</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DH 0103</td>
<td>TERMINATE_DOMAIN</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DH 0104</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DH 0105</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DH 0106</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DH 0107</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DH 0108</td>
<td></td>
<td></td>
</tr>
<tr>
<td>APUE</td>
<td>DH 0D01</td>
<td>SET_EXIT_STATUS</td>
<td>APUE</td>
</tr>
<tr>
<td></td>
<td>DH 0D02</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DH 0D03</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DH 0D04</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DH 0D05</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DH 0D06</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DH 0D07</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DH 0D08</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RMRO</td>
<td>DH 0301</td>
<td>PERFORM_PREPARE</td>
<td>RMRO</td>
</tr>
<tr>
<td></td>
<td>DH 0302</td>
<td>PERFORM_COMMIT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DH 0303</td>
<td>PERFORM_SHUNT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DH 0304</td>
<td>PERFORM_UNSHUNT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DH 0305</td>
<td>START_BACKOUT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DH 0308</td>
<td>END_BACKOUT</td>
<td></td>
</tr>
</tbody>
</table>
Table 42. Document Handler domain’s generic gates (continued)

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>RMDE</td>
<td>DH 0301</td>
<td>START_DELIVERY</td>
<td>RMDE</td>
</tr>
<tr>
<td></td>
<td>DH 0302</td>
<td>DELIVER_RECOVERY</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DH 0303</td>
<td>END_DELIVERY</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DH 0304</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DH 0306</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DH 0308</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RMKP</td>
<td>DH 0301</td>
<td>TAKE_KEYPOINT</td>
<td>RMKP</td>
</tr>
<tr>
<td></td>
<td>DH 0302</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DH 0303</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DH 0304</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DH 0307</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DH 0308</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For descriptions of these functions and their input and output parameters, refer to the §§ dealing with the corresponding generic formats:

**Functions and parameters**

| Format DMDM—“Domain manager domain’s generic formats” on page 377 |
| Format APUE—“Application domain’s generic formats” on page 100 |
| Format RMRO—“Recovery Manager domain’s call back formats” on page 953 |
| Format RMDE—“Recovery Manager domain’s call back formats” on page 953 |
| Format RMKP—“Recovery Manager domain’s call back formats” on page 953 |

**Modules**

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHDHDM</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>INITIALIZE_DOMAIN</td>
</tr>
<tr>
<td></td>
<td>QUIESCE_DOMAIN</td>
</tr>
<tr>
<td></td>
<td>TERMINATE_DOMAIN</td>
</tr>
<tr>
<td>DFHDHDH</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>CREATE_DOCUMENT</td>
</tr>
<tr>
<td></td>
<td>INSERT_DATA</td>
</tr>
<tr>
<td></td>
<td>INSERT_BOOKMARK</td>
</tr>
<tr>
<td></td>
<td>REPLACE_DATA</td>
</tr>
<tr>
<td></td>
<td>DELETE_DOCUMENT</td>
</tr>
<tr>
<td></td>
<td>DELETE_DATA</td>
</tr>
<tr>
<td></td>
<td>DELETE_BOOKMARK</td>
</tr>
<tr>
<td></td>
<td>RETRIEVE_WITH_CTLINFO</td>
</tr>
<tr>
<td></td>
<td>RETRIEVEWITHOUT_CTLINFO</td>
</tr>
<tr>
<td></td>
<td>INQUIRE_DOCUMENT</td>
</tr>
<tr>
<td>Module</td>
<td>Function</td>
</tr>
<tr>
<td>---------</td>
<td>--------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DFHDHSL</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>SET_SYMBOL_VALUE_BY_API,</td>
</tr>
<tr>
<td></td>
<td>SET_SYMBOL_VALUE_BY_SSI,</td>
</tr>
<tr>
<td></td>
<td>ADD_SYMBOL_LIST</td>
</tr>
<tr>
<td></td>
<td>EXPORT_SYMBOL_LIST</td>
</tr>
<tr>
<td></td>
<td>IMPORT_SYMBOL_LIST</td>
</tr>
<tr>
<td></td>
<td>DFHDHTM Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>INITIALIZE_DOCTEMPLATES</td>
</tr>
<tr>
<td></td>
<td>ADD_REPLACE_DOCTEMPLATE</td>
</tr>
<tr>
<td></td>
<td>DELETE_DOCTEMPLATE</td>
</tr>
<tr>
<td></td>
<td>INQUIRE_DOCTEMPLATE</td>
</tr>
<tr>
<td></td>
<td>INQUIRE_TEMPLATE_STATUS</td>
</tr>
<tr>
<td></td>
<td>START_BROWSE</td>
</tr>
<tr>
<td></td>
<td>GET_NEXT</td>
</tr>
<tr>
<td></td>
<td>END_BROWSE</td>
</tr>
<tr>
<td></td>
<td>READ_TEMPLATE</td>
</tr>
<tr>
<td></td>
<td>DFHDHRM Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>PERFORM_PREPARE</td>
</tr>
<tr>
<td></td>
<td>PERFORM_COMMIT</td>
</tr>
<tr>
<td></td>
<td>PERFORM_SHUNT</td>
</tr>
<tr>
<td></td>
<td>PERFORM_UNSHUNT</td>
</tr>
<tr>
<td></td>
<td>START_BACKOUT</td>
</tr>
<tr>
<td></td>
<td>END_BACKOUT</td>
</tr>
<tr>
<td></td>
<td>START_DELIVERY</td>
</tr>
<tr>
<td></td>
<td>DELIVER_RECOVERY</td>
</tr>
<tr>
<td></td>
<td>END_DELIVERY</td>
</tr>
<tr>
<td></td>
<td>TAKE_KEYPOINT</td>
</tr>
<tr>
<td></td>
<td>DFHDHUUE Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>SET_EXIT_STATUS</td>
</tr>
<tr>
<td></td>
<td>DFHDHPB Processes data supplied on the BINARY parameter of</td>
</tr>
<tr>
<td></td>
<td>CREATE_DOCUMENT, INSERT_DATA and REPLACE_DATA calls of DFHDHDDH.</td>
</tr>
<tr>
<td></td>
<td>DFHDHPD Processes data supplied on the SOURCE_DOCUMENT parameter of</td>
</tr>
<tr>
<td></td>
<td>CREATE_DOCUMENT, INSERT_DATA and REPLACE_DATA calls of DFHDHDDH.</td>
</tr>
<tr>
<td></td>
<td>DFHDHPM Processes data supplied on the TEMPLATE_NAME parameter of</td>
</tr>
<tr>
<td></td>
<td>CREATE_DOCUMENT, INSERT_DATA and REPLACE_DATA calls of DFHDHDDH.</td>
</tr>
<tr>
<td></td>
<td>DFHDHPS Processes data supplied on the SYMBOL parameter of INSERT_DATA</td>
</tr>
<tr>
<td></td>
<td>and REPLACE_DATA calls of DFHDHDDH.</td>
</tr>
<tr>
<td></td>
<td>DFHDHPT Processes data supplied on the TEXT parameter of</td>
</tr>
<tr>
<td></td>
<td>CREATE_DOCUMENT, INSERT_DATA and REPLACE_DATA calls of DFHDHDDH.</td>
</tr>
<tr>
<td></td>
<td>DFHDHPU Processes data supplied on the TEMPLATE_BUFFER parameter of</td>
</tr>
<tr>
<td></td>
<td>CREATE_DOCUMENT, INSERT_DATA and REPLACE_DATA calls of DFHDHDDH.</td>
</tr>
<tr>
<td></td>
<td>DFHDHPX Processes data supplied on the RETRIEVED_DOCUMENT parameter of</td>
</tr>
<tr>
<td></td>
<td>CREATE_DOCUMENT, INSERT_DATA and REPLACE_DATA calls of DFHDHDDH.</td>
</tr>
<tr>
<td></td>
<td>DFHDHDR Reads templates held as member’s of partitioned datasets.</td>
</tr>
<tr>
<td></td>
<td>DFHDHEI Reads templates held on CICS resources.</td>
</tr>
<tr>
<td></td>
<td>DFHDHPR Reads PDS members containing templates.</td>
</tr>
<tr>
<td></td>
<td>DFHDHDUF DH domain offline dump formatting routine</td>
</tr>
</tbody>
</table>
Document Handler domain (DH)

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHDHTRI</td>
<td>Interprets DH domain trace entries</td>
</tr>
</tbody>
</table>

Exits

No global user exit points are provided in this domain.

Trace

The point IDs for the document handler domain are of the form DH xxxx; the corresponding trace levels are DH 1, DH 2, and Exc.

For more information about the trace points, see the CICS Trace Entries. For more information about using traces in problem determination, see the CICS Problem Determination Guide.
Chapter 28. Domain manager domain (DM)

The domain manager domain (also sometimes known simply as “domain manager”) is responsible for maintaining, through the use of catalog services, permanent information about individual domains.

Each domain has certain permanent characteristics. These are stored on the local catalog and include the name, token, and ID; these characteristics are unique for each domain. Each domain also has volatile characteristics (including the phase number and the status), which are not stored on the catalog.

The domain manager attaches initialization and termination tasks for other domains. It maintains phase information of the other domains to allow controlled introduction and withdrawal of domain services during initialization and termination. For each domain, a phase number denotes the set of services that are available from the domain. An increased phase number would correspond to an increased set of available functions.

During initialization, the system phase is the minimum of the phase numbers of the active domains. During shutdown, the system phase is the maximum of the phase numbers of the active domains.

The domain manager also maintains and manages a queue of waiting domains (called “waiters”); these waiting domains are waiting for a specific domain to reach a certain phase or for the system phase to reach a certain level.

Domain manager domain’s specific gates

Table 43 summarizes the domain manager domain’s specific gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and whether or not the functions are available through the exit programming interface (XPI).

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>XPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMDM</td>
<td>DM 0001</td>
<td>ADD_DOMAIN</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>DM 0002</td>
<td>QUIESCE_SYSTEM</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SET_PHASE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WAIT_PHASE</td>
<td>NO</td>
</tr>
<tr>
<td>DMIQ</td>
<td>DM 0003</td>
<td>START_BROWSE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>DM 0004</td>
<td>GET_NEXT</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>END_BROWSE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INQ_DOMAIN_BY_NAME</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INQ_DOMAIN_BY_TOKEN</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INQ_DOMAIN_BY_ID</td>
<td>NO</td>
</tr>
<tr>
<td>DMEN</td>
<td>DM 0210</td>
<td>LISTEN</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>DM 0211</td>
<td>DELETE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NOTIFY_SMSVSAM_OPERATIONAL</td>
<td>NO</td>
</tr>
</tbody>
</table>
Domain manager domain (DM)

**DMEN gate, LISTEN function**

The LISTEN function of the DMEN gate is issued to register an interest in an event notification facility (ENF) event. The MVS event notification facility is a generalized communication facility which allows subsystems to broadcast notification of events.

If a domain wishes to be notified of particular ENF events, it must register the events that it wishes to be notified of with Domain Manager using the LISTEN interface.

When an ENF event occurs domain manager will invoke the named listen gate of all domains that registered for that event.

**Input parameters**

EVENT

is the event that the caller is registering an interest in, and can have any of these values:

- SMSVSAM_OPERATIONAL

LISTEN_GATE

is the gate number of the gate at which the caller wishes to be notified when the event occurs.

**Output parameters**

RESPONSE

is DFHDMEN’s response to the call. It can have any of these values:

- OK
- EXCEPTION
- DISASTER
- INVALID
- KERNERROR
- PURGED

[REASON]

is returned when response is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>UNKNOWN_EVENT, DUPLICATE_LISTEN</td>
</tr>
</tbody>
</table>

**DMEN gate, DELETE function**

The DELETE function of the DMEN gate is used to deregister an interest in an ENF event.

If a domain is registered with domain manager for notification of an ENF event and that domain no longer wishes to receive notification of that event then it can deregister its interest in the event using the DELETE interface.

**Input parameters**

EVENT

is the event which the caller wishes to deregister its interest in. It can have any of these values:

- SMSVSAM_OPERATIONAL

LISTEN_GATE

is the gate number of the gate which the caller specified as its listen gate when it registered an interest in this event.
Domain manager domain (DM)

Output parameters

RESPONSE
is DFHDMEN’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|

[REASON]
is returned when response is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>LISTEN_NOT_ACTIVE</td>
</tr>
</tbody>
</table>

DMEN gate, NOTIFY_SMSVSAM_OPERATIONAL function

Domains that have registered their interest in ENF events are invoked at their identified listen gates when the ENF event occurs. A unique DMEN notify function is provided for each event to allow event specific parameters to be specified in a meaningful way.

The NOTIFY_SMSVSAM_OPERATIONAL function of the DMEN gate is used to notify domains which have registered an interest in it of the occurrence of the SMSVSAM operational event.

Input parameters

NOTIFY_PLIST
is a parameter list specific to the ENF event being notified, which was supplied by the subsystem issuing the ENF signal.

Output parameters

RESPONSE
is DFHDMEN’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|

[REASON]
is returned when response is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>RESTART_RLS_FAILED</td>
</tr>
</tbody>
</table>

DMDM gate, ADD_DOMAIN function

The ADD_DOMAIN function of the DMDM gate adds a new domain to the DM table (on the CICS catalog) of all domains. Because the add is placed on the catalog, it survives system failure. A delete is required to remove the entry.

Input parameters

DOMAIN_NAME
is a unique string, 1 through 8 characters, which is the name of the domain.

PROGRAM_NAME
is a unique string, 1 through 8 characters, which is the name of the initialization module for the specified domain.
Domain manager domain (DM)

**DOMAIN_TOKEN**

is the unique index that corresponds to the new table entry for the domain.

**DOMAIN_ID**

is the unique character pair, usually an abbreviated form of the domain name.

**Output parameters**

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

- OK
- EXCEPTION
- INVALID
- KERNERROR
- DISASTER

**[REASON]**

is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>LOADER_ERROR, ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>DUPLICATE_DOMAIN_NAME, PROGRAM_NOT_FOUND, INSUFFICIENT_STORAGE, DUPLICATE_DOMAIN_TOKEN</td>
</tr>
</tbody>
</table>

**DMDM gate, QUIESCE_SYSTEM function**

The QUIESCE_SYSTEM function of the DMDM gate is used to call the domain manager to cause a normal shutdown of the system.

**Input parameters**

None.

**Output parameters**

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

- OK
- INVALID
- KERNERROR
- DISASTER

**[REASON]**

is returned when RESPONSE is DISASTER or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>INSUFFICIENT_STORAGE, ABEND, LOOP</td>
</tr>
<tr>
<td>INVALID</td>
<td>SYSTEM_INITIALIZING</td>
</tr>
</tbody>
</table>

**DMDM gate, SET_PHASE function**

When a domain issues SET_PHASE during initialization, it is declaring that it is now prepared to support a given set of services.

When a domain issues SET_PHASE during quiesce, it is asserting that it still needs the set of services identified by that phase number.

The system phase is the minimum of all active domains’ phases during initialization, and the maximum during quiesce.
Input parameters

**PHASE**
specifies the set of services that are to be available.

**STATUS**
is either ACTIVE or INACTIVE.

Output parameters

**RESPONSE**
is the domain’s response to the call. It can have any of these values: OK, INVALID, KERNERROR, DISASTER

**[REASON]**
is returned when RESPONSE is DISASTER or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>INVALID</td>
<td>SYSTEM_NOT_INITIALIZING, SYSTEM_NOTQUIESCING, INVALID_PHASE</td>
</tr>
</tbody>
</table>

**DMDM gate, WAIT_PHASE function**

The WAIT_PHASE function of the DMDDM gate is used to wait until the services required to carry on the work are available.

A WAIT_PHASE for a given phase is understood by CICS as a SET_PHASE for at least the phase specified in the phase parameter of WAIT_PHASE.

Input parameters

**PHASE**
specifies the set of services that are to be available.

**STATUS**
specifies the required status. It is either ACTIVE or INACTIVE.

**[DOMAIN_TOKEN]**
specifies the domain. If this is omitted, a wait on the system phase is actioned, rather than for a particular domain.

Output parameters

**RESPONSE**
is the domain’s response to the call. It can have any of these values: OK, EXCEPTION, INVALID, KERNERROR, DISASTER

**[REASON]**
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>DOMAIN_TOKEN_NOT_ACTIVE</td>
</tr>
<tr>
<td></td>
<td>(This can be returned only if the DOMAIN_TOKEN option is specified.)</td>
</tr>
<tr>
<td>INVALID</td>
<td>SYSTEM_NOT_INITIALIZING, SYSTEM_NOTQUIESCING, INVALID_PHASE</td>
</tr>
</tbody>
</table>
Domain manager domain (DM)

**DMIQ gate, START_BROWSE function**
The START_BROWSE function of the DMIQ gate is used to create a browse thread. The GET_NEXT function request issued after this command returns the first domain in the active domain list.

**Input parameters**
None.

**Output parameters**

<table>
<thead>
<tr>
<th>BROWSE_TOKEN</th>
<th>is the token identifying this browse session.</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESPONSE</td>
<td>is the domain’s response to the call. It can have any of these values: OK</td>
</tr>
<tr>
<td>[REASON]</td>
<td>is returned when RESPONSE is DISASTER. Possible values are: ABEND, LOOP</td>
</tr>
</tbody>
</table>

**DMIQ gate, GET_NEXT function**
The GET_NEXT function of the DMIQ gate is used to return the next available record or an END indication.

**Input parameters**

| BROWSE_TOKEN | is the token identifying this browse session. |

**Output parameters**

| DOMAIN_NAME | is a unique string, 1 through 8 characters, which is the name of the domain. |
| PROGRAM_NAME| is a unique string, 1 through 8 characters, which is the name of the initialization module for the specified domain. |
| DOMAIN_TOKEN| is the unique index that corresponds to the new table entry for the domain. |
| DOMAIN_ID   | is the unique character pair, usually an abbreviated form of the domain name. |
| DOMAIN_STATUS| is ACTIVE or INACTIVE. |
| DOMAIN_PHASE| is the current phase level for that domain. |
| RESPONSE    | is the domain’s response to the call. It can have any of these values: OK|EXCEPTION|INVALID|KERNERROR|DISASTER |
| [REASON]    | is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. |
Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>END_LIST</td>
</tr>
<tr>
<td>INVALID</td>
<td>BROWSE_TOKEN_NOT_FOUND</td>
</tr>
</tbody>
</table>

**DMIQ gate, END_BROWSE function**

The END_BROWSE function of the DMIQ gate is used to release the browse thread at any time.

**Input parameters**

**BROWSE_TOKEN**

is the token identifying this browse session.

**Output parameters**

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK | INVALID | KERNERROR | DISASTER

[REASON]

is returned when RESPONSE is DISASTER or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>INVALID</td>
<td>BROWSE_TOKEN_NOT_FOUND</td>
</tr>
</tbody>
</table>

**DMIQ gate, INQ_DOMAIN_BY_NAME function**

The INQ_DOMAIN_BY_NAME function of the DMIQ gate is used to get the domain’s token, ID, status, and phase for the specified domain name.

**Input parameters**

**DOMAIN_NAME**

is the unique name of an existing domain.

**Output parameters**

**DOMAIN_TOKEN**

is the unique index that corresponds to the table entry for the domain.

**DOMAIN_ID**

is the unique character pair, usually an abbreviated form of the domain name.

**DOMAIN_STATUS**

is ACTIVE or INACTIVE.

**DOMAIN_PHASE**

is the current phase level for that domain.

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK | INVALID | KERNERROR | DISASTER
Domain manager domain (DM)

[REASON]

is returned when RESPONSE is DISASTER or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>INVALID</td>
<td>DOMAIN_NAME_NOT_FOUND</td>
</tr>
</tbody>
</table>

DMIQ gate, INQ_DOMAIN_BY_TOKEN function

The INQ_DOMAIN_BY_TOKEN function of the DMIQ gate is used to get the domain’s name, ID, status, and phase for the specified domain token.

**Input parameters**

**DOMAIN_TOKEN**

is the unique index that corresponds to the table entry for the domain.

**Output parameters**

**DOMAIN_NAME**

is a unique string, 1 through 8 characters, which is the name of the domain.

**DOMAIN_ID**

is the unique character pair, usually an abbreviated form of the domain name.

**DOMAIN_STATUS**

is ACTIVE or INACTIVE.

**DOMAIN_PHASE**

is the current phase level for that domain.

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK|INVALID|KERNERROR|DISASTER

[REASON]

is returned when RESPONSE is DISASTER or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>INVALID</td>
<td>DOMAIN_TOKEN_NOT_FOUND</td>
</tr>
</tbody>
</table>

DMIQ gate, INQ_DOMAIN_BY_ID function

The INQ_DOMAIN_BY_ID function of the DMIQ gate is used to get the domain’s token, name, status, and phase for the specified domain ID.

**Input parameters**

**DOMAIN_ID**

is the unique character pair, usually an abbreviated form of the domain name.
Output parameters

**DOMAIN_TOKEN**
- is the unique index that corresponds to the table entry for the domain.

**DOMAIN_NAME**
- is a unique string, 1 through 8 characters, which is the name of the domain.

**DOMAIN_STATUS**
- is ACTIVE or INACTIVE.

**DOMAIN_PHASE**
- is the current phase level for that domain.

**RESPONSE**
- is the domain’s response to the call. It can have any of these values:
  - OK
  - INVALID
  - KERNERROR
  - DISASTER

**[REASON]**
- is returned when RESPONSE is DISASTER or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>INVALID</td>
<td>DOMAIN_ID_NOT_FOUND</td>
</tr>
</tbody>
</table>

Domain manager domain’s generic gates

Table 44 summarizes the domain manager domain’s generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSAT</td>
<td>None</td>
<td>TASK_REPLY</td>
<td>DSAT</td>
</tr>
</tbody>
</table>

For descriptions of the DSAT function and its input and output parameters, refer to the section dealing with the corresponding generic format:

Functions and parameters

Format DSAT—“Dispatcher domain’s generic formats” on page 324

Domain manager domain’s generic formats

Table 45 describes the generic formats owned by the domain manager domain and shows the functions performed on the calls.

<table>
<thead>
<tr>
<th>Format</th>
<th>Calling module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMDM</td>
<td>DFHKETCB</td>
<td>PRE_INITIALIZE</td>
</tr>
<tr>
<td></td>
<td>DFHKETCB</td>
<td>INITIALIZE_DOMAIN</td>
</tr>
<tr>
<td></td>
<td>DFHKETCB</td>
<td>QUIESCE_DOMAIN</td>
</tr>
<tr>
<td></td>
<td>DFHKETCB</td>
<td>TERMINATE_DOMAIN</td>
</tr>
</tbody>
</table>
In the descriptions of the formats that follow, the “input” parameters are input not to the domain manager domain, but to the domain being called by the domain manager. Similarly, the “output” parameters are output by the domain that was called by the domain manager, in response to the call.

**DMDM format, PRE_INITIALIZE function**

The DFHKETCB module issues a preinitialization call to each of the following domains: LC, PA, TR, ME, DU, LM, SM, DD, DS, XM, LD, and DM.

Apart from the LD, and DM domains, preinitialization takes place under the job-step TCB; for LD, and DM, it takes place under the resource-owning (RO) TCB.

In preinitialization processing, the domain manager domain reads information about domains from the local catalog, and passes it to the kernel. It then attaches the initialization tasks for all the other domains.

**Input parameters**
None.

**Output parameters**

- **RESPONSE**
  is the domain’s response to the call. It can have any of these values: OK|INVALID|KERNERROR|DISASTER

- **[REASON]**
  is returned when RESPONSE is DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>INSUFFICIENT_STORAGE, ABEND, LOOP</td>
</tr>
</tbody>
</table>

**DMDM format, INITIALIZE_DOMAIN function**

The domain manager domain issues an INITIALIZE_DOMAIN function call to a domain. In initialization processing, the domain manager domain performs only internal routines.

**Input parameters**
None.

**Output parameters**

- **RESPONSE**
  is the domain’s response to the call. It can have any of these values: OK|INVALID|KERNERROR|DISASTER

- **[REASON]**
  is returned when RESPONSE is DISASTER or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>INSUFFICIENT_STORAGE, ABEND, LOOP</td>
</tr>
<tr>
<td>INVALID</td>
<td>ALREADY_INITIALIZED</td>
</tr>
</tbody>
</table>
DMDM format, QUIESCE_DOMAIN function

The domain manager domain issues a QUIESCE_DOMAIN function call to a domain when the system is required to shut down normally. The domain manager domain initiates quiesce processing by attaching the quiesce task for each domain.

Input parameters
None.

Output parameters
RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|INVALID|KERNERROR|DISASTER

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>INSUFFICIENT_STORAGE, ABEND, LOOP</td>
</tr>
</tbody>
</table>

DMDM format, TERMINATE_DOMAIN function

The domain manager domain issues a TERMINATE_DOMAIN function call to a domain when the system is required to shut down quickly. This call is always issued under the job-step TCB.

The domain manager domain does no termination processing.

Input parameters
CLEAN_UP
indicates whether or not the TERMINATE_DOMAIN function request is being issued under a cleanup-only ESTAE exit. It can have either of these values:
YES|NO

YES implies restrictions for termination logic, specifically that an ATTACH request cannot be issued.

CANCEL
indicates whether or not the termination is happening because of an operator CANCEL command. It can have either of these values:
YES|NO

YES means that attached subtasks are no longer dispatchable.

TERMINATION_TYPE
indicates whether the termination is happening because of either a quiesce or an abnormal shutdown. It can have either of these values:
QUIESCE|IMMEDIATE

Output parameters
RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|INVALID|KERNERROR|DISASTER

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:
### Modules

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHDMDM</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>INITIALIZE_DOMAIN</td>
</tr>
<tr>
<td></td>
<td>PRE_INITIALIZE</td>
</tr>
<tr>
<td></td>
<td>QUIESCE_DOMAIN</td>
</tr>
<tr>
<td></td>
<td>QUIESCE_SYSTEM</td>
</tr>
<tr>
<td></td>
<td>TERMINATE_DOMAIN</td>
</tr>
<tr>
<td></td>
<td>SET_PHASE</td>
</tr>
<tr>
<td></td>
<td>WAIT_PHASE</td>
</tr>
<tr>
<td></td>
<td>ADD_DOMAIN</td>
</tr>
<tr>
<td>DFHDMDS</td>
<td>Handles the TASK_REPLY request</td>
</tr>
<tr>
<td>DFHDMDFS</td>
<td>Formats the DM domain control blocks in a CICS system dump</td>
</tr>
<tr>
<td>DFHDMEN</td>
<td>Handles LISTEN, DELETE, NOTIFY_SMSVSAM_OPERATIONAL</td>
</tr>
<tr>
<td>DFHDMENF</td>
<td>Broadcasts ENF events to interested domains</td>
</tr>
<tr>
<td>DFHDMIQ</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>START_BROWSE</td>
</tr>
<tr>
<td></td>
<td>GET_NEXT</td>
</tr>
<tr>
<td></td>
<td>END_BROWSE</td>
</tr>
<tr>
<td></td>
<td>INQUIRE_DOMAIN_BY_ID</td>
</tr>
<tr>
<td></td>
<td>INQUIRE_DOMAIN_BY_NAME</td>
</tr>
<tr>
<td></td>
<td>INQUIRE_DOMAIN_BY_TOKEN</td>
</tr>
<tr>
<td>DFHDMSC</td>
<td>Provides authorized services for the DM ENF support</td>
</tr>
<tr>
<td>DFHDMTRI</td>
<td>Interprets DM domain trace entries</td>
</tr>
<tr>
<td>DFHDMWQ</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>INITIALIZE</td>
</tr>
<tr>
<td></td>
<td>SET_UP_WAIT</td>
</tr>
<tr>
<td></td>
<td>RESUME_WAITERS</td>
</tr>
<tr>
<td></td>
<td>RESUME_DOMAIN_WAITERS</td>
</tr>
<tr>
<td></td>
<td>RESUME_PHASE_WAITERS</td>
</tr>
</tbody>
</table>

### Exits

No global user exit points are provided in this domain.

### Trace

The point IDs for the domain manager domain are of the form DM xxxx; the corresponding trace levels are DM 1, DM 2, and Exc.

For more information about the trace points, see the [CICS Trace Entries](#). For more information about using traces in problem determination, see the [CICS Problem Determination Guide](#).
Chapter 29. Dump domain (DU)

The dump domain is responsible for producing storage dumps and for handling the associated data sets and status in the CICS system. Two types of dump are produced:

Transaction dumps
These are written to the CICS-managed BSAM data sets DFHDMPA and DFHDMPB. They consist of the storage areas related to a particular transaction.

System dumps
CICS uses the MVS SDUMP facility to dump the entire CICS region to an MVS SYS1.DUMP data set.

The two dump tables (one for each dump type) are indexed by the dump code and contain details of the options required for each request.

Design overview

Figure 51 gives an overview of the dump domain architecture.

![Diagram of CICS dump domain structure]

Figure 51. CICS dump domain structure
### Dump domain’s specific gates

Table 46 summarizes the dump domain’s specific gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and whether or not the functions are available through the exit programming interface (XPI).

**Table 46. Dump domain’s specific gates**

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>XPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>DUDT</td>
<td>DU 0500</td>
<td>ADD_TRAN_DUMPCODE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>DU 0501</td>
<td>DELETE_TRAN_DUMPCODE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INQUIRE_TRAN_DUMPCODE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SET_TRAN_DUMPCODE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>STARTBR_TRAN_DUMPCODE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GETNEXT_TRAN_DUMPCODE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ENDBR_TRAN_DUMPCODE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADD_SYSTEM_DUMPCODE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DELETE_SYSTEM_DUMPCODE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INQUIRE_SYSTEM_DUMPCODE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SET_SYSTEM_DUMPCODE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>STARTBR_SYSTEM_DUMPCODE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GETNEXT_SYSTEM_DUMPCODE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ENDBR_SYSTEM_DUMPCODE</td>
<td>NO</td>
</tr>
<tr>
<td>DUDU</td>
<td>DU 0101</td>
<td>TRANSACTION_DUMP</td>
<td>YES</td>
</tr>
<tr>
<td></td>
<td>DU 0102</td>
<td>SYSTEM_DUMP</td>
<td>YES</td>
</tr>
<tr>
<td>DUSR</td>
<td>DU 0301</td>
<td>DUMPDS_OPEN</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>DU 0302</td>
<td>DUMPDS_CLOSE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DUMPDS_SWITCH</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INQUIRE_CURRENT_DUMPDS</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INQUIRE_DUMPDS_OPEN_STATUS</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INQUIRE_DUMPDS_AUTOSWITCH</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SET_DUMPDS_AUTOSWITCH</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SET_DUMPTABLE_DEFAULTS</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INQUIRE_INITIAL_DUMPDS</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SET_INITIAL_DUMPDS</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INQUIRE_SYSTEM_DUMP</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SET_SYSTEM_DUMP</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INQUIRE_RETRY_TIME</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SET_RETRY_TIME</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SET_CONNECT_TOKEN</td>
<td></td>
</tr>
</tbody>
</table>

**DUDT gate, ADD_TRAN_DUMPCODE function**

The ADD_TRAN_DUMPCODE function of the DUDT gate is invoked to add a new dump code to the transaction dump table.

**Input parameters**

**DUMPSCOPE**

- **LOCAL**
- **RELATED**

**LOCAL**

- indicates that the SDUMP request is not sent to MVS images in the sysplex which are running XCF/MRO connected CICS systems
RELATED indicates that, when an SDUMP is initiated for the dump code, the request is sent to all MVS images in the sysplex which are running one or more CICS systems connected via XCF/MRO to the CICS on which the SDUMP is initiated.

TRANSACTION_DUMPCODE is the transaction dump code.

TRANSACTION_DUMP states whether a transaction dump is required for this dump code. It can have either of these values:

YES | NO

SYSTEM_DUMP states whether a system dump is required for this dump code. It can have either of these values:

YES | NO

TERMINATE_CICS states whether CICS is to be terminated for this dump code. It can have either of these values:

YES | NO

MAXIMUM_DUMPS is the maximum number of times the dump code action can be taken.

Output parameters

RESPONSE is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>DUPLICATE_DUMPCODE, INVALID_DUMPCODE, CATALOG_FULL, INSUFFICIENT_STORAGE, IO_ERROR</td>
</tr>
</tbody>
</table>

Process flow

1. Issue LMLM LOCK for DUTABLE lock.
2. Issue DUDT ADD_TRAN_DUMPCODE call to DFHDUTM.
3. Issue LMLM UNLOCK for DUTABLE lock.

DFHDUTM process flow

1. Validate the dump code to be added. Transaction dump codes are 4 bytes and must not contain leading or embedded blanks. If the dump code is not valid, return to the caller indicating the exception.
2. Scan the transaction dump table to find the correct place to insert the dump code in collating sequence. If an entry already exists for that dump code, return to the caller indicating duplicate dump code. If the entry is about to use the last available entry in the dump table block, obtain a new block and initialize it with null values. Create a dump table entry in the next available entry, indicated by TDTFREEHEAD pointer in the anchor block, using the parameter
values passed by the caller. Set up the NEXT and PREV pointers of the new
entry and higher and lower entries to include the new entry in the correct
sequence in the table.

3. Write the dump code information to the global catalog.

**DUDT gate, DELETE_TRAN_DUMPCODE function**

The DELETE_TRAN_DUMPCODE function of the DUDT gate is invoked to delete
an existing dump code from the transaction dump table.

**Input parameters**

TRANSACTION_DUMPCODE

is the transaction dump code.

**Output parameters**

RESPONSE

is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]

is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>DUMPCODE_NOT_FOUND, IO_ERROR</td>
</tr>
</tbody>
</table>

**Process flow**

1. Issue LMLM LOCK for DUTABLE lock.
2. Issue DUDT DELETE_TRAN_DUMPCODE call to DFHDUTM.
3. Issue LMLM UNLOCK for DUTABLE lock.

**DFHDUTM process flow**

1. Locate the dump code in the transaction dump table. If it cannot be found, return to the caller indicating DUMPCODE_NOT_FOUND exception.
2. Adjust the NEXT and PREV of the higher and lower entries in the table to bypass this entry, and set its NEXT and PREV pointers to 0.
3. Delete the information for the dump code from the global catalog. If the attempt to delete from the catalog indicates that the record is not found, it is assumed that the dump code was present on the dump table as a result of a LOCATE_TRAN_DUMPCODE subroutine call that does not update the catalog.

**DUDT LOCATE_TRAN_DUMPCODE process flow**

1. Validate the dump code for which a dump has been requested (see
   ADD_TRAN_DUMPCODE).
2. Search the transaction dump table for the dump code. If it is found, set up the return DUDT parameters to indicate whether CICS is to be terminated, and whether a system or transaction dump is to be taken, using values taken from the dump table entry.

If the dump code does not exist on the dump table, an entry is added, using default values (see the CICS Problem Determination Guide) and the DUDT return parameters are set up dependent on these default values. (This default entry is not added to the global catalog.)
DUDT gate, INQUIRE_TRAN_DUMPCODE function

The INQUIRE_TRAN_DUMPCODE function of the DUDT gate is invoked to inquire on a dump code in the transaction dump table.

**Input parameters**

**TRANSACTION_DUMPCODE**

is the transaction dump code.

**Output parameters**

**[DUMPSCOPE]**

indicates whether an SDUMP request is to be sent to all MVS images in the sysplex which are running CICS systems connected via XCF/MRO to the system on which the command is issued. It can have either of the following values:

LOCAL | RELATED

**LOCAL**

indicates that the SDUMP request is not sent to MVS images in the sysplex which are running XCF/MRO connected CICS systems

**RELATED**

indicates that, when an SDUMP is initiated for the dump code, the request is sent to all MVS images in the sysplex which are running one or more CICS systems connected via XCF/MRO to the CICS on which the SDUMP is initiated.

**[TRANSACTION_DUMP]**

states whether a transaction dump is required for this dump code. It can have either of these values:

YES | NO

**[SYSTEM_DUMP]**

states whether a system dump is required for this dump code. It can have either of these values:

YES | NO

**[TERMINATE_CICS]**

states whether CICS is to be terminated for this dump code. It can have either of these values:

YES | NO

**[MAXIMUM_DUMPS]**

is the maximum number of times the dump code action can be taken.

**[COUNT]**

is the number of times the dump code action has been taken.

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

**[REASON]**

is returned when RESPONSE is EXCEPTION. It has this value:

DUMPCODE_NOT_FOUND

**Process flow**

1. Issue LMLM LOCK for DUTABLE lock.
2. Issue DUDT INQUIRE_TRAN_DUMPCODE call to DFHDUTM.
3. Issue LMLM UNLOCK for DUTABLE lock.

**DFHDUTM process flow**
1. Locate the dump code in the transaction dump code table. If it cannot be found, return to the caller indicating DUMPCODE_NOT_FOUND exception.
2. Return the dump code table entry information to the caller in the DUDT parameters.

**DUDT gate, SET_TRAN_DUMPCODE function**
The SET_TRAN_DUMPCODE function of the DUDT gate is invoked to set options for a dump code in the transaction dump table.

**Input parameters**

[DUMPSCOPE]
indicates whether an SDUMP request is to be sent to all MVS images in the sysplex which are running CICS systems connected via XCF/MRO to the system on which the command is issued. It can have either of the following values:
LOCAL | RELATED
LOCAL
indicates that the SDUMP request is not sent to MVS images in the sysplex which are running XCF/MRO connected CICS systems
RELATED
indicates that, when an SDUMP is initiated for the dump code, the request is sent to all MVS images in the sysplex which are running one or more CICS systems connected via XCF/MRO to the CICS on which the SDUMP is initiated.

TRANSACTION_DUMPCODE
is the transaction dump code.

[TRANSACTION_DUMP]
states whether a transaction dump is required for this dump code. It can have either of these values:
YES | NO

[SYSTEM_DUMP]
states whether a system dump is required for this dump code. It can have either of these values:
YES | NO

[TERMINATE_CICS]
states whether CICS is to be terminated for this dump code. It can have either of these values:
YES | NO

[MAXIMUM_DUMPS]
is the maximum number of times the dump code action can be taken.

[RESET_COUNT]
states whether COUNT is to be reset to zero. It can have either of these values:
YES | NO
Output parameters

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK\|EXCEPTION\|DISASTER\|INVALID\|KERNERROR\|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>DUMPCODE_NOT_FOUND, CATALOG_FULL, IO_ERROR</td>
</tr>
</tbody>
</table>

Process flow

1. Issue LMLM LOCK for DUTABLE lock.
2. Issue DUDT SET_TRAN_DUMPCODE call to DFHDUTM.
3. Issue LMLM UNLOCK for DUTABLE lock.

DFHDUTM process flow

1. Locate the dump code in the transaction dump code table. If it cannot be found, return to the caller indicating DUMPCODE_NOT_FOUND exception.
2. Change the values on the dump code table entry for any passed in the DUDT parameter list (some or all may be changed). If the RESET_COUNT parameter is present, set the count of the number of dumps taken for this dump code to zero.
3. Make the same changes to the dump code information about the global catalog. If the attempt to delete from the catalog indicates that the record is not found, it is assumed that the dump code was present on the dump table as a result of a LOCATE_TRAN_DUMPCODE subroutine call that does not update the catalog. See “DUDT LOCATE_TRAN_DUMPCODE process flow” on page 384 for a description of the process flow of this function.

DUDT gate, STARTBR_TRAN_DUMPCODE function

The STARTBR_TRAN_DUMPCODE function of the DUDT gate is invoked to start a browse session on the transaction dump table.

Input parameters
None.

Output parameters

BROWSE_TOKEN
is the token identifying the browse session.

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK\|EXCEPTION\|DISASTER\|INVALID\|KERNERROR\|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. It has this value:
INSUFFICIENT_STORAGE

Process flow

1. Issue LMLM LOCK for DUTABLE lock.
2. Issue DUDT STARTBR_TRAN_DUMPCODE call to DFHDUTM.
3. Issue LMLM UNLOCK for DUTABLE lock.
DFHDUTM process flow

1. Add a new browse token to the end of the browse token table. Set the value of the last dump code used to null in the browse token table entry.
2. Return the browse token to the caller.

DUDT gate, GETNEXT_TRAN_DUMPCode function

The GETNEXT_TRAN_DUMPCode function of the DUDT gate is invoked in a browse session to get the next entry in the transaction dump table.

Input parameters

BROWSE_TOKEN
is the token identifying the browse session.

Output parameters

[DUMPSCOPE]
indicates whether an SDUMP request is to be sent to all MVS images in the sysplex which are running CICS systems connected via XCF/MRO to the system on which the command is issued. It can have either of the following values:
LOCAL | RELATED

LOCAL
indicates that the SDUMP request is not sent to MVS images in the sysplex which are running XCF/MRO connected CICS systems

RELATED
indicates that, when an SDUMP is initiated for the dump code, the request is sent to all MVS images in the sysplex which are running one or more CICS systems connected via XCF/MRO to the CICS on which the SDUMP is initiated.

[TRANSACTION_DUMPCode]
is the transaction dump code.

[TRANSACTION_DUMP]
states whether a transaction dump is required for this dump code. It can have either of these values:
YES | NO

[System_DUMP]
states whether a system dump is required for this dump code. It can have either of these values:
YES | NO

[TERMINATE_CICS]
states whether CICS is to be terminated for this dump code. It can have either of these values:
YES | NO

[MAXIMUM_DUMPS]
is the maximum number of times the dump code action can be taken.

[COUNT]
is the number of times the dump code action has been taken.

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED
[REASON] is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>END_BROWSE</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_BROWSE_TOKEN</td>
</tr>
</tbody>
</table>

**Process flow**
1. Issue LMLM LOCK for DUTABLE lock.
2. Issue DUDT GETNEXT_TRAN_DUMPCODE call to DFHDUTM.
3. Issue LMLM UNLOCK for DUTABLE lock.

**DFHDUTM process flow**
1. Search the browse token table for the browse token passed in the DUDT parameters. If the browse token cannot be found, perform error handling (exception trace, message, and dump) and return to the caller.
2. Obtain the value of the last dump code read by this browse session from the browse token table entry, and scan the dump table for a higher dump code entry. If there are no more entries, return END_BROWSE exception to the call; otherwise return the details of the dump code table entry in the parameters and save the value of the dump code in the browse token entry.

**DUDT gate, ENDBR_TRAN_DUMPCODE function**
The ENDBR_TRAN_DUMPCODE function of the DUDT gate is invoked to end a browse session on the transaction dump table.

**Input parameters**
**BROWSE_TOKEN** is the token identifying the browse session.

**Output parameters**
**RESPONSE** is the domain’s response to the call. It can have any of these values:
- OK
- EXCEPTION
- DISASTER
- INVALID
- KERNERROR
- PURGED

[REASON] is returned when RESPONSE is INVALID. It has this value:
- INVALID_BROWSE_TOKEN

**Process flow**
1. Issue LMLM LOCK for DUTABLE lock.
2. Issue DUDT ENDBR_TRAN_DUMPCODE call to DFHDUTM.
3. Issue LMLM UNLOCK for DUTABLE lock.

**DFHDUTM process flow**
1. Search the browse token table for the browse token passed in the DUDT parameters. If the browse token cannot be found, perform error handling (exception trace, message, and dump) and return to the caller.
2. Set the browse token table entry to nulls and adjust the NEXT and PREV pointers to bypass the entry.
Dump domain (DU)

**DUDT gate, ADD_SYSTEM_DUMPCODE function**

The ADD_SYSTEM_DUMPCODE function of the DUDT gate is invoked to add a new dump code to the system dump table.

**Input parameters**

**DAEOPTION**
states whether a dump produced for this dumpcode is eligible for suppression by the MVS Dump Analysis and Elimination (DAE) component. It can have either of these values:

YES | NO

**DUMPSCOPE**
indicates whether an SDUMP request is to be sent to all MVS images in the sysplex which are running CICS systems connected via XCF/MRO to the system on which the command is issued. It can have either of the following values:

LOCAL | RELATED

**LOCAL** indicates that the SDUMP request is not sent to MVS images in the sysplex which are running XCF/MRO connected CICS systems

**RELATED** indicates that, when an SDUMP is initiated for the dump code, the request is sent to all MVS images in the sysplex which are running one or more CICS systems connected via XCF/MRO to the CICS on which the SDUMP is initiated.

**SYSTEM_DUMPCODE**

is the system dump code.

**SYSTEM_DUMP**
states whether a system dump is required for this dump code. It can have either of these values:

YES | NO

**TERMINATE_CICS**
states whether CICS is to be terminated for this dump code. It can have either of these values:

YES | NO

**MAXIMUM_DUMPS**
is the maximum number of times the dump code action can be taken.

**Output parameters**

**RESPONSE**
is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[**[REASON]**] is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>DUPLICATE_DUMPCODE, INVALID_DUMPCODE, CATALOG_FULL, INSUFFICIENT_STORAGE, IO_ERROR</td>
</tr>
</tbody>
</table>
Process flow
1. Acquire KE system dump lock.
2. Issue DUDT ADD_SYSTEM_DUMPCODE call to DFHDUTM.
3. Release KE system dump lock.

DFHDUTM process flow
1. Validate the dump code to be added. System dump codes are 4 bytes and must not contain leading or embedded blanks. If the dump code is not valid, return to the caller indicating the exception.
2. Scan the system dump table to find the correct place to insert the dump code in collating sequence. If an entry already exists for that dump code, return to the caller indicating duplicate dump code. If the entry is about to use the last available entry in the dump table block, obtain a new block and initialize it with null values. Create a dump table entry in the next available entry, indicated by TDTFREEHEAD pointer in the anchor block, using the parameter values passed by the caller. Set up the NEXT and PREV pointers of the new entry and higher and lower entries to include the new entry in the correct sequence in the table.
3. Write the dump code information to the global catalog.

DUDT gate, DELETE_SYSTEM_DUMPCODE function
The DELETE_SYSTEM_DUMPCODE function of the DUDT gate is invoked to delete an existing dump code from the system dump table.

Input parameters
SYSTEM_DUMPCODE
is the system dump code.

Output parameters
RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>DUMPCODE_NOT_FOUND, IO_ERROR</td>
</tr>
</tbody>
</table>

Process flow
1. Acquire KE system dump lock.
2. Issue DUDT DELETE_SYSTEM_DUMPCODE call to DFHDUTM.
3. Release KE system dump lock.

DFHDUTM process flow
1. Locate the dump code in the system dump table. If it cannot be found, return to the caller indicating DUMPCODE_NOT_FOUND exception.
2. Adjust the NEXT and PREV of the higher and lower entries in the table to bypass this entry, and set its NEXT and PREV pointers to 0.
3. Delete the information for the dump table from the global catalog. If the attempt to delete from the catalog indicates that the record is not found, it is
Dump domain (DU)

assumed that the dump code was present on the dump table as a result of a
LOCATE_SYSTEM_DUMPCODE subroutine call that does not update the
catalog.

DUDT LOCATE_SYSTEM_DUMPCODE process flow
1. Validate the dump code for which a dump has been requested (see
   ADD_SYSTEM_DUMPCODE).
2. Search the system dump table for the dump code. If it is found, set up the
   return DUDT parameters to indicate whether CICS is to be terminated, and
   whether a system dump is to be taken, using values taken from the dump table
   entry.
   If the dump code does not exist on the dump table, an entry is added, using
   default values (see the CICS Problem Determination Guide) and the DUDT return
   parameters are set up dependent on these default values. (This default entry is
   not added to the global catalog.)

DUDT gate, INQUIRE_SYSTEM_DUMPCODE function
The INQUIRE_SYSTEM_DUMPCODE function of the DUDT gate is invoked to
inquire on a dump code in the system dump table.

Input parameters
SYSTEM_DUMPCODE
is the system dump code.

Output parameters
DAEOPTION
states whether a dump produced for this dumpcode is eligible for
suppression by the MVS Dump Analysis and Elimination (DAE)
component. It can have either of these values:
YES|NO

DUMPSCOPE
indicates whether an SDUMP request is to be sent to all MVS images in the
sysplex which are running CICS systems connected via XCF/MRO to the
system on which the command is issued. It can have either of the
following values:
LOCAL|RELATED

LOCAL
indicates that the SDUMP request is not sent to MVS images in the
sysplex which are running XCF/MRO connected CICS systems

RELATED
indicates that, when an SDUMP is initiated for the dump code, the
request is sent to all MVS images in the sysplex which are running one
or more CICS systems connected via XCF/MRO to the CICS on which
the SDUMP is initiated.

[System_Dump]
states whether a system dump is required for this dump code. It can have
either of these values:
YES|NO

[Terminate_CICS]
states whether CICS is to be terminated for this dump code. It can have
either of these values:
[MAXIMUM_DUMPS] is the maximum number of times the dump code action can be taken.

[COUNT] is the number of times the dump code action has been taken.

RESPONSE is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. It has this value:
DUMPCODE_NOT_FOUND

**Process flow**
1. Acquire KE system dump lock.
2. Issue DUDT INQUIRE_SYSTEM_DUMPCODE call to DFHDUTM.
3. Release KE system dump lock.

**DFHDUTM process flow**
1. Locate the dump code in the system dump code table. If it cannot be found, return to the caller indicating DUMPCODE_NOT_FOUND exception.
2. Return the dump code table entry information to the caller in the DUDT parameters.

**DUDT gate, SET_SYSTEM_DUMPCODE function**
The SET_SYSTEM_DUMPCODE function of the DUDT gate is invoked to set options for a dump code in the system dump table.

**Input parameters**

[DAEOPTION] states whether a dump produced for this dumpcode is eligible for suppression by the MVS Dump Analysis and Elimination (DAE) component. It can have either of these values:
YES|NO

[DUMPSCOPE] indicates whether an SDUMP request is to be sent to all MVS images in the sysplex which are running CICS systems connected via XCF/MRO to the system on which the command is issued. It can have either of the following values:
LOCAL|RELATED

LOCAL indicates that the SDUMP request is not sent to MVS images in the sysplex which are running XCF/MRO connected CICS systems

RELATED indicates that, when an SDUMP is initiated for the dump code, the request is sent to all MVS images in the sysplex which are running one or more CICS systems connected via XCF/MRO to the CICS on which the SDUMP is initiated.

SYSTEM_DUMPCODE is the system dump code.
Dump domain (DU)

[SYSTEM_DUMP]
states whether a system dump is required for this dump code. It can have either of these values:
YES|NO

[TERMINATE_CICS]
states whether CICS is to be terminated for this dump code. It can have either of these values:
YES|NO

[MAXIMUM_DUMPS]
is the maximum number of times the dump code action can be taken.

[RESET_COUNT]
states whether COUNT is to be reset to zero. It can have either of these values:
YES|NO

Output parameters
RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>DUMPCODE_NOT_FOUND, CATALOG_FULL, IO_ERROR</td>
</tr>
</tbody>
</table>

Process flow
1. Acquire KE system dump lock.
2. Issue DUDT SET_SYSTEM_DUMPCODE call to DFHDUTM.
3. Release KE system dump lock.

DFHDUTM process flow
1. Locate the dump code in the system dump code table. If it cannot be found, return to the caller indicating DUMPCODE_NOT_FOUND exception.
2. Change the values on the dump code table entry for any passed in the DUDT parameter list (some or all may be changed). If the RESET_COUNT parameter is present, set the count of the number of dumps taken for this dump code to zero.
3. Make the same changes to the dump code information about the global catalog. If the attempt to delete from the catalog indicates that the record is not found, it is assumed that the dump code was present on the dump table as a result of a LOCATE_SYSTEM_DUMPCODE subroutine call that does not update the catalog. See "DUDT LOCATE_SYSTEM_DUMPCODE process flow" on page 392 for a description of the process flow of this function.

DUDT gate, STARTBR_SYSTEM_DUMPCODE function
The STARTBR_SYSTEM_DUMPCODE function of the DUDT gate is invoked to start a browse session on the system dump table.

Input parameters
None.
Output parameters

**BROWSE_TOKEN**

is the token identifying the browse session.

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

- OK  
- EXCEPTION  
- DISASTER  
- INVALID  
- KERNERROR  
- PURGED

**[REASON]**

is returned when RESPONSE is EXCEPTION. It has this value:

- INSUFFICIENT_STORAGE

Process flow

1. Acquire KE system dump lock.
2. Issue DUDT STARTBR_SYSTEM_DUMPCODE call to DFHDUTM.
3. Release KE system dump lock.

**DFHDUTM process flow**

1. Add a new browse token to the end of the browse token table. Set the value of the last dump code used to null in the browse token table entry.
2. Return the browse token to the caller.

**DUDT gate, GETNEXT_SYSTEM_DUMPCODE function**

The GETNEXT_SYSTEM_DUMPCODE function of the DUDT gate is invoked in a browse session to get the next entry in the system dump table.

Input parameters

**BROWSE_TOKEN**

is the token identifying the browse session.

Output parameters

**[DAEOPTION]**

states whether a dump produced for this dump code is eligible for suppression by the MVS Dump Analysis and Elimination (DAE) component. It can have either of these values:

- YES  
- NO

**[DUMPSCOPE]**

indicates whether an SDUMP request is to be sent to all MVS images in the sysplex which are running CICS systems connected via XCF/MRO to the system on which the command is issued. It can have either of the following values:

- LOCAL  
- RELATED

**LOCAL**

indicates that the SDUMP request is not sent to MVS images in the sysplex which are running XCF/MRO connected CICS systems

**RELATED**

indicates that, when an SDUMP is initiated for the dump code, the request is sent to all MVS images in the sysplex which are running one or more CICS systems connected via XCF/MRO to the CICS on which the SDUMP is initiated.

**[SYSTEM_DUMPCODE]**

is the system dump code.
[SYSTEM_DUMP] states whether a system dump is required for this dump code. It can have either of these values:
YES|NO

[TERMINATE_CICS] states whether CICS is to be terminated for this dump code. It can have either of these values:
YES|NO

[MAXIMUM_DUMPS] is the maximum number of times the dump code action can be taken.

[COUNT] is the number of times the dump code action has been taken.

RESPONSE is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>END_BROWSE</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_BROWSE_TOKEN</td>
</tr>
</tbody>
</table>

Process flow
1. Acquire KE system dump lock.
2. Issue DUDT GETNEXT_SYSTEM_DUMPCODE call to DFHDUTM.
3. Release KE system dump lock.

DFHDUTM process flow
1. Search the browse token table for the browse token passed in the DUDT parameters. If the browse token cannot be found, perform error handling (exception trace, message, and dump) and return to the caller.
2. Obtain the value of the last dump code read by this browse session from the browse token table entry, and scan the dump table for a higher dump code entry. If there are no more entries, return END_BROWSE exception to the call; otherwise return the details of the dump code table entry in the parameters and save the value of the dump code in the browse token table entry.

DUDT gate, ENDBR_SYSTEM_DUMPCODE function
The ENDBR_SYSTEM_DUMPCODE function of the DUDT gate is invoked to end a browse on the system dump table.

Input parameters
BROWSE_TOKEN is the token identifying the browse session.

Output parameters
RESPONSE is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED
[REASON] is returned when RESPONSE is INVALID. It has this value:

INVALID_BROWSE_TOKEN

Process flow
1. Acquire KE system dump lock.
2. Issue DUDT ENDBR_SYSTEM_DUMPCODE call to DFHDUTM.
3. Release KE system dump lock.

DFHDUTM process flow
1. Search the browse token table for the browse token passed in the DUDT parameters. If the browse token cannot be found, perform error handling (exception trace, message, and dump) and return to the caller.
2. Set the browse token table entry to nulls and adjust the NEXT and PREV pointers to bypass the entry.

DUDU gate, TRANSACTION_DUMP function
The TRANSACTION_DUMP function of the DUDU gate is invoked to take a transaction dump.

Input parameters

Note: The [SEGMENT] and [SEGMENT_LIST] parameters are mutually exclusive.

TRANSACTION_DUMPCODE is a 4-character identifier for this dump request, used to index the transaction dump table to determine the options to be used.

The following set of optional input parameters indicates which parts of storage are to be included in the transaction dump. Each parameter can have either of these values: YES|NO.

[CSA] – common system area
[CSA] – common system area
[TCA] – task control area
[PROGRAM] – program storage
[TRT] – internal trace table
[TERMINAL] – terminal-related storage areas
[TRANSACTION] – transaction-related storage areas
[SIT] – system initialization table
[PPT] – processing program table
[PCT] – program control table
[TCT] – terminal control table
[FCT] – file control table

[SEGMENT] specifies the address and length of a single block of storage to be dumped.
Dump domain (DU)

[SEGMENT_LIST]
specifies the address and length of a list of length-address pairs of storage blocks to be dumped. SEGMENT and SEGMENT_LIST may not be specified together.

[INDIRECT_CALL]
states whether the call is indirect, that is, whether the actual requester of the dump is not the immediate caller of the dump domain. It can have either of these values:

YES | NO

Output parameters

DUMPID
is a character string of the form “rrrr/cccc” giving a unique identification to this dump request. “rrrr” is the run number of this CICS instance. Leading zeros are removed. The run number is incremented every time CICS is initialized. “cccc” is the count of this dump request within this CICS run.

RESPONSE
is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. It can have any one of these values:

FESTAE_FAILED
The MVS FESTAE macro failed to set up a functional recovery routine during the processing of the system dump request.

IWMWQWRK_FAILED
An MVS IWMWQWRK macro call to Workload Manager returned a warning or error response during the processing of the system dump request.

INVALID_SVC_CALL
DFHDUSVC received a request for an invalid function.

INVALID_PROBDESC
The MVS PROBDESC parameters, which CICS creates and passes to MVS on an SDUMP call, contained invalid data.

OPEN_ERROR
Failed to open the CICS dump data set during an autoswitch.

NOT_OPEN
The dump data set is currently closed.

INVALID_DUMPCODE
The transaction dump code failed validation.

PARTIAL_TRANSACTION_DUMP
There was insufficient space in the current dump data set for this dump. Autoswitching had not been requested.

SUPPRESSED_BY_DUMPOPTION
A system dump requested through the dump table for this transaction dump code was suppressed because the DUMP=NO system initialization parameter had been specified.
**Dump domain (DU)**

**SUPPRESSED_BY_DUMPTABLE**
The dump table specified that no dump was required for this dump code.

**SUPPRESSED_BY_USEREXIT**
The XDUREQ user exit requested suppression of this dump.

**PARTIAL_SYSTEM_DUMP**
A system dump requested through the dump table for this transaction dump code was incomplete because of insufficient space on the SYS1.DUMP data set.

**SDUMP_FAILED**
A system dump requested through the dump table for this transaction dump code failed because of an MVS or I/O failure.

**SDUMP_BUSY**
A system dump requested through the dump table for this transaction dump code failed because another address space was in the process of taking an SDUMP.

**SDUMP_NOT_AUTHORIZED**
A system dump requested through the dump table for this transaction dump code failed because the CICS authorized function control block (AFCB) indicates that CICS use of SDUMP is not authorized.

**INSUFFICIENT_STORAGE**
A system dump requested through the dump table for this transaction dump code failed because CICS failed to acquire the necessary storage to build the SDUMP parameter list.

**NO_DATASET**
A system dump requested through the dump table for this transaction dump code failed because there were no SYS1.DUMP data sets available.

**Process flow**

1. Issue LMLM LOCK for DUTABLE lock.
2. Issue DUDT LOCATE_TRAN_DUMPCODE call to DFHDUTM. If the dump table is not available, CICS takes a system dump and terminates.
3. Issue LMLM UNLOCK for DUTABLE lock.
4. If XDUREQ exit active, issue APEX INVOKE_USER_EXIT.
5. If XDUREQ exit not active or it was active and the return code was zero:
   - If dump table indicates that a system dump is required for this transaction dump code and the DUMP=NO system initialization parameter was not specified, invoke CICS SVC to take system dump, retrying as necessary if SDUMP is busy.
   - If dump table indicates that a transaction dump is required, call DFHDUXD with a DUDD format parameter list to take a transaction dump.
6. If XDUREQC exit active, issue APEX INVOKE_USER_EXIT.
7. Issue LMLM LOCK for DUTABLE lock.
8. Issue DUDT COMMIT_TRAN_DUMPCODE call to DFHDUTM.
9. Issue LMLM UNLOCK for DUTABLE lock.
10. Issue KEDD PERFORM_SYSTEM_ACTION to terminate CICS if the dump table indicated that termination was required for this dump code.
Dump domain (DU)

**DUDD TAKE_DUMP process flow**

In DFHDUXD:

1. If dump data set is closed or is a dummy data set, and the XDUOUT exit is not active, return to caller.
2. Issue LMLM LOCK for dump data set lock.
3. Invoke transaction dump formatting routines (DFHxxXDF), with DUXF FORMAT function, in turn to dump required areas to the transaction dump data set. If, at any point, the DUXF FORMAT function returns a response of EXCEPTION and a reason of RESTART, an autoswitch has occurred and the DUXF FORMAT calls have to be issued again.
4. Issue LMLM UNLOCK for dump data set lock.
5. If DFHDUXD is terminating with a DISASTER response and XDUOUT is active, issue APEX INVOKE_USER_EXIT for XDUOUT, passing the abnormal termination indication.

**DUDT COMMIT_TRAN_DUMPCODE process flow**

The DUDT COMMIT_TRAN_DUMPCODE function updates statistics for the dump code, according to whether or not the dump domain took the requested dumps.

1. Locate the entry on the transaction dump table. Return to the caller, indicating exception if the entry is not found.
2. Increment the global system dump statistics in the DUA and the system dump statistics on the dump table entry, for either dump-taken or dump-suppressed depending on the input system-dump parameter.
3. Increment the global transaction dump statistics in the DUA and the transaction dump statistics for either dump-taken or dump-suppressed depending on the input transaction-dump parameter.

**DUDU gate, SYSTEM_DUMP function**

The SYSTEM_DUMP function of the DUDU gate is invoked to take a system dump.

**Input parameters**

**SYSTEM_DUMPCODE**

is an 8-character identifier for this dump request, used to index the system dump table to determine the options to be used.

**[MESSAGE_TEXT]**

specifies the address and length of the message text associated with this system dump.

**[TITLE]**

specifies the address and length of a title to be associated with this dump.

**[CALLER]**

specifies the address and length of a character string to appear as the caller of this dump.

**[SYMPTOM_RECORD]**

specifies the address and length of the symptom record associated with this dump.

**[SYMPTOM_STRING]**

specifies the address and length of the symptom string associated with this dump.
TERMINATE_CICS
states whether CICS is to be terminated after the dump if there is no entry in the dump table for this dump code; that is, it overrides the termination default of NO. It can have either of these values:
YES|NO

INDIRECT_CALL
states whether the call is indirect, that is, whether the actual requester of the dump is not the immediate caller of the dump domain. It can have either of these values:
YES|NO

Output parameters

DUMPID
is a character string of the form “rrrr/cccc” giving a unique identification to this dump request. “rrrr” is the run number of this CICS instance. Leading zeros are removed. The run number is incremented every time CICS is initialized. “cccc” is the count of this dump request within this CICS run.

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

REASON
is returned when RESPONSE is EXCEPTION. It can have any one of these values:
INVALID_DUMPCODE
The system dump code failed validation.

SUPPRESSED_BY_DUMPOPTION
A system dump requested through the dump table for this system dump code was suppressed because the DUMP=NO system initialization parameter had been specified.

SUPPRESSED_BY_DUMPTABLE
The dump table specified that no dump was required for this dump code.

SUPPRESSED_BY_USEREXIT
The XDUREQ user exit requested suppression of this dump.

PARTIAL_SYSTEM_DUMP
A system dump requested through the dump table for this system dump code was incomplete because of insufficient space on the SYS1.DUMP data set.

SDUMP_FAILED
A system dump requested through the dump table for this system dump code failed because of an MVS or I/O failure.

SDUMP_BUSY
A system dump requested through the dump table for this system dump code failed because another address space was in the process of taking an SDUMP.

SDUMP_NOT_AUTHORIZED
A system dump requested through the dump table for this system dump code failed because the CICS authorized function control block (AFCB) indicates that CICS use of SDUMP is not authorized.
Dump domain (DU)

**INSUFFICIENT_STORAGE**
A system dump requested through the dump table for this system dump code failed because CICS failed to acquire the necessary storage to build the SDUMP parameter list.

**NO_DATASET**
A system dump requested through the dump table for this system dump code failed because there were no SYS1.DUMP data sets available.

**Process flow**
1. Acquire KE system dump lock.
2. If the DUMP=YES system initialization parameter was specified:
   • Issue DUDT LOCATE_SYSTEM_DUMPCODE call to DFHDUTM.
   • If dump table indicates system dump required:
     – If XDUREQ exit active, issue APEX INVOKE_USER_EXIT.
     – If XDUREQ exit not active or it was active and the return code was zero, invoke CICS SVC to take system dump, retrying as necessary if SDUMP is busy.
     – If XDUREQC exit active, issue APEX INVOKE_USER_EXIT.
3. Issue DUDT COMMIT_SYSTEM_DUMPCODE call to DFHDUTM.
4. Release KE system dump lock.
5. Issue KEDD PERFORM_SYSTEM_ACTION to terminate CICS if the dump table indicated that termination was required for this dump code.

**DUDT COMMIT_SYSTEM_DUMPCODE process flow**
The COMMIT_SYSTEM_DUMPCODE function of the DUDT gate updates statistics for the dump code, according to whether or not the dump domain took the requested dumps.
• Locate the entry on the system dump table. Return to the caller, indicating exception if the entry is not found.
• Increment the global system dump statistics and the system dump statistics on the dump table entry, for either dump-taken or dump-suppressed depending on the input system-dump parameter.

**DUSR gate, CROSS_SYSTEM_DUMP_AVAIL function**
The CROSS_SYSTEM_DUMP_AVAIL function of the DUSR gate is used to inform the dump domain about the DUMP_AVAIL token which links CICS with the MVS workload manager.

**Input parameters**
CROSS_SYSTEM_DUMP_AVAIL is the CICS to MVS workload manager token.

**Output parameters**
RESPONSE is the domain’s response to the call. It can have any of these values:
OK|DISASTER|INVALID|KERNERROR|PURGED

**Process flow**
• Set the CICS to MVS workload manager connect token in the DUA.
DUSR gate, DUMPDS_OPEN function

The DUMPDS_OPEN function of the DUSR gate is invoked to open the CICS dump data set.

**Input parameters**

None.

**Output parameters**

- **RESPONSE**
  
  is the domain's response to the call. It can have any of these values:
  
  OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

- **[REASON]**
  
  is returned when RESPONSE is EXCEPTION. It has this value:
  
  OPEN_ERROR

  which indicates that the MVS OPEN of the dump data set failed.

**Process flow**

1. Issue LMLM LOCK for dump data set lock.
2. Call DUSU OPEN function.
3. Issue LMLM UNLOCK for dump data set lock.

DUSR gate, DUMPDS_CLOSE function

The DUMPDS_CLOSE function of the DUSR gate is invoked to close the CICS dump data set.

**Input parameters**

None.

**Output parameters**

- **RESPONSE**
  
  is the domain's response to the call. It can have any of these values:
  
  OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

**Process flow**

1. Issue LMLM LOCK for dump data set lock.
2. Call DUSU CLOSE function.
3. Issue LMLM UNLOCK for dump data set lock.

DUSR gate, DUMPDS_SWITCH function

The DUMPDS_SWITCH function of the DUSR gate is invoked to switch to the alternate CICS dump data set.

**Input parameters**

None.

**Output parameters**

- **RESPONSE**
  
  is the domain's response to the call. It can have any of these values:
  
  OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

- **[REASON]**
  
  is returned when RESPONSE is EXCEPTION. It has this value:
Dump domain (DU)

**OPEN_ERROR**

which indicates that the MVS OPEN of the dump data set failed.

**Process flow**
1. Issue LMLM LOCK for dump data set lock.
2. Call DUSU SWITCH function.
3. Issue LMLM UNLOCK for dump data set lock.

**DUSU SWITCH process flow**
1. Process as for DUSU CLOSE.
2. Switch current data set name in the DUA.
3. Process as for DUSU OPEN.

**DUSU OPEN process flow**
1. Return if the DUA indicates already open.
2. Call DUIO OPEN function.
3. Update status on catalog.

**DUSU CLOSE process flow**
1. If data set is open:
   - Call DUIO ALLOC_STG function to get storage for DYNALLOC parameter list.
   - Issue DYNALLOC to get data set name for current dump data set.
2. Call DUIO CLOSE function.
3. If XDUCLSE exit is active, call APEX INVOKE_USER_EXIT.
4. Set status in the DUA to closed.
5. Free DYNALLOC parameter list if necessary.

**DUIO OPEN process flow**
1. Return if the DUA indicates transaction dump data set is already open.
2. Issue MVS GETMAIN for DU Open Block if it is not yet allocated.
3. Issue MVS OPEN.
4. Set status to open in the DUA.
5. Write end-of-data record.

DUIO uses the DCB OPEN exit to complete the DCB with block size and LRECL, and to determine the size of the buffer to be used by CICS. The DCB abend exit and the SYNAD routine are also activated to detect any errors that may occur during OPEN.

**DUIO CLOSE process flow**
1. Return if already closed.
2. Issue MVS CLOSE.
3. Issue MVS FREEPOOL to release buffers.
4. If this close is not for a switch, free the DU open block.
5. Set status to closed in the DUA.

**DUIO ALLOC_STG process flow**
1. Issue MVS GETMAIN for requested storage.
2. Clear acquired area to hexadecimal zeros.
DUSR gate, **INQUIRE_CURRENT_DUMPDS function**

The INQUIRE_CURRENT_DUMPDS function of the DUSR gate returns the name of the current dump data set.

**Input parameters**
None.

**Output parameters**

**CURRENT_DUMPDS**

is the name of the current dump data set. It can have either of these values:

DFHDMPA | DFHDMPB

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

DUSR gate, **INQUIRE_DUMPDS_OPEN_STATUS function**

The INQUIRE_DUMPDS_OPEN_STATUS function of the DUSR gate returns an indication of whether the current dump data set is open or closed.

**Input parameters**
None.

**Output parameters**

**OPEN_STATUS**

is the open status of the current dump data set. It can have either of these values:

OPEN | CLOSED

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

DUSR gate, **INQUIRE_DUMPDS_AUTOSWITCH function**

The INQUIRE_DUMPDS_AUTOSWITCH function of the DUSR gate returns an indication of whether autoswitching is active or not.

**Input parameters**
None.

**Output parameters**

**AUTOSWITCH**

is the dump data set autoswitch status. It can have either of these values:

ON | OFF

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

DUSR gate, **SET_DUMPDS_AUTOSWITCH function**

The SET_DUMPDS_AUTOSWITCH function of the DUSR gate is used to set autoswitching on or off.
Dump domain (DU)

**Input parameters**

**AUTOSWITCH**

is the dump data set autoswitch status. It can have either of these values:

- ON
- OFF

**Output parameters**

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

- OK
- EXCEPTION
- DISASTER
- INVALID
- KERNERROR
- PURGED

**Process flow**

1. Set new autoswitch value in the DUA.
2. Call DUSU UPDATE_CATALOGUE function, to write the DU state record to local catalog, using the current status from the DUA.

**DUSR gate, INQUIRE_INITIAL_DUMPDS function**

The `INQUIRE_INITIAL_DUMPDS` function of the DUSR gate returns the setting of the initial dump data set.

**Input parameters**

None.

**Output parameters**

**INITIAL_DUMPDS**

is the initial dump data set. It can have any one of these values:

- DFHDMPA
  
  Open DFHDMPA first when CICS is next initialized.

- DFHDMPB
  
  Open DFHDMPB first when CICS is next initialized.

- AUTO
  
  When CICS is next initialized, open the extent that was not active when CICS last terminated.

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

- OK
- EXCEPTION
- DISASTER
- INVALID
- KERNERROR
- PURGED

**DUSR gate, SET_INITIAL_DUMPDS function**

The `SET_INITIAL_DUMPDS` function of the DUSR gate is used to change the setting of the initial dump data set.

**Input parameters**

**INITIAL_DUMPDS**

is the initial dump data set. It can have any one of these values:

- DFHDMPA
  
  Open DFHDMPA first when CICS is next initialized.

- DFHDMPB
  
  Open DFHDMPB first when CICS is next initialized.

- AUTO
  
  When CICS is next initialized, open the extent that was not active when CICS last terminated.
Output parameters

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

Process flow

1. Set new initial dump data set value in the DUA.
2. Call DUSU UPDATE_CATALOGUE function, to write the DU state record to local catalog, using the current status from the DUA.

**DUSR** gate, **SET_DUMPTABLE_DEFAULTS** function

The **SET_DUMPTABLE_DEFAULTS** function of the DUSR gate is invoked during system initialization to update the DUA with the DAE option specified in a SIT or as a SIT override.

Input parameters

**DAE_DEFAULT**

indicates whether temporary dump table entries added by CICS will indicate DAE (dump eligible for DAE suppression) of NODAE (dump will not be suppressed by DAE). It can have either of the values:

DAE | NODAE

**SYDUMAX_DEFAULT**

is taken from system initialization parameter (SIT=SYDUMAX), which specifies the maximum number of system dumps which can be taken per dump table entry.

**TRDUMAX_DEFAULT**

is taken from system initialization parameter (SIT=TRDUMAX), which specifies the maximum number of transaction dumps which can be taken per dump table entry.

Output parameters

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK | DISASTER | INVALID | KERNERROR | PURGED

Process flow

1. Set DAE_DEFAULT flag value in the DUA. 1 indicates DAE, 0 indicates NODAE.
2. Call DUSU UPDATE_CATALOGUE function, to write the DU state record to local catalog, using the current status from the DUA.

**DUSR** gate, **INQUIRE_SYSTEM_DUMP** function

The **INQUIRE_SYSTEM_DUMP** function of the DUSR gate returns the setting of the system dump suppression flag.

Input parameters

None.

Output parameters

**SYSTEM_DUMP**

is the system dump option, indicating whether or not SDUMPs are to be taken by this CICS system. It can have either of these values:
Dump domain (DU)

YES|NO

where NO means that SDUMPs are not taken by this CICS system.

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

DUSR gate, SET_SYSTEM_DUMP function
The SET_SYSTEM_DUMP function of the DUSR gate is used to change the setting of the system dump suppression flag.

Input parameters
SYSTEM_DUMP
is the system dump option, indicating whether or not SDUMPs are to be taken by this CICS system. It can have either of these values:
YES|NO

where NO means that SDUMPs are not taken by this CICS system.

Output parameters
RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

Process flow
• Set new system dump suppression flag value in the DUA.
• Call DUSU UPDATE_CATALOGUE function, to write the DU state record to local catalog, using the current status from the DUA.

DUSR gate, INQUIRE_RETRY_TIME function
The INQUIRE_RETRY_TIME function of the DUSR gate returns the value of the SDUMP retry time.

Input parameters
None.

Output parameters
RETRY_TIME
is the value in seconds of the time interval for which CICS should retry SDUMP requests that fail because another SDUMP is in progress within the MVS system. The SDUMP is retried at intervals of five seconds for the specified total time.

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

DUSR gate, SET_RETRY_TIME function
The SET_RETRY_TIME function of the DUSR gate is invoked to set the SDUMP retry time.
Input parameters

RETRY_TIME

is the value in seconds of the time interval for which CICS should retry SDUMP requests that fail because another SDUMP is in progress within the MVS system. The SDUMP is retried at intervals of five seconds for the specified total time.

Output parameters

RESPONSE

is the domain's response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

Process flow

1. Set new SDUMP retry time in the DUA.
2. Call DUSU UPDATE_CATALOGUE function, to write the DU state record to local catalog, using the current status from the DUA.

Miscellaneous process flows

DUIO format, WRITE function

1. If the first record in the block to be written is a dump header:
   • Issue MVS NOTE to get location of last record written (an end-of-data record).
   • Issue MVS POINT to position for overwrite of the end-of-data record.
2. Issue MVS WRITE.
3. Issue DSSR WAIT_MVS on the I/O ECB.
4. Issue MVS CHECK for I/O completion. This drives the DCB abend exit if an error or end-of-extent is encountered and results in an error or END_OF_EXTENT response from DUIO.

DUSU format, WRITE function

1. Call DUIO WRITE function if the dump data set is open and is not a dummy.
2. If an end-of-extent occurred:
   • If autoswitch is not active, close data set as for DUSU CLOSE above.
   • If autoswitch is active, turn autoswitch off and process as for DUSU SWITCH.

DUXF format, FORMAT function

This is the format of the parameter list passed to the transaction dump formatting routines (DFHxxXDF). There is a SUB_FUNCTION parameter which indicates the areas to be dumped. Each formatting routine is responsible for handling a subset of the subfunctions. The subfunctions and corresponding formatting modules are listed below in the order of the subfunction invocation from DFHDUXD.

<table>
<thead>
<tr>
<th>Module</th>
<th>Subfunction</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHXDxdf</td>
<td>DUXF_FORMAT_DUMP_HEADER</td>
</tr>
<tr>
<td>DFHXDxdf</td>
<td>DUXF_FORMAT_SHORT_SYMPTOM_STRIN</td>
</tr>
<tr>
<td>DFHXDxdf</td>
<td>DUXF_FORMAT_CICS_SERVICE_LEVEL</td>
</tr>
<tr>
<td>DFHXDxdf</td>
<td>DUXF_FORMAT_PSW_REGISTERS</td>
</tr>
<tr>
<td>DFHSAXDF</td>
<td>DUXF_FORMAT_TCA</td>
</tr>
<tr>
<td>DFHPCCxdf</td>
<td>DUXF_FORMAT_LIFO</td>
</tr>
<tr>
<td>DFHSAXDF</td>
<td>DUXF_FORMAT_COMM AREAS</td>
</tr>
</tbody>
</table>
Dump domain (DU)

<table>
<thead>
<tr>
<th>Module</th>
<th>Subfunction</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHSAXDF</td>
<td>DUXF_FORMAT_CSA</td>
</tr>
<tr>
<td>DFHTRXDF</td>
<td>DUXF_FORMAT_TRT</td>
</tr>
<tr>
<td>DFHXDxdf</td>
<td>DUXF_FORMAT_SEGMENT</td>
</tr>
<tr>
<td>DFHXDxdf</td>
<td>DUXF_FORMAT_SEGMENT_LIST</td>
</tr>
<tr>
<td>DFHSAXDF</td>
<td>DUXF_FORMAT_TRANSACTION_STORAGE</td>
</tr>
<tr>
<td>DFHSAXDF</td>
<td>DUXF_FORMAT_FCA</td>
</tr>
<tr>
<td>DFHTCXDF</td>
<td>DUXF_FORMAT_TCTT</td>
</tr>
<tr>
<td>DFHPCXDF</td>
<td>DUXF_FORMAT_PROGRAM</td>
</tr>
<tr>
<td>DFHSAXDF</td>
<td>DUXF_FORMAT_DCT</td>
</tr>
<tr>
<td>DFHFCXDF</td>
<td>DUXF_FORMAT_FCT</td>
</tr>
<tr>
<td>DFHTCXDF</td>
<td>DUXF_FORMAT_TCT</td>
</tr>
<tr>
<td>DFHXRXDF</td>
<td>DUXF_FORMAT_XRF</td>
</tr>
<tr>
<td>DFHPCXDF</td>
<td>DUXF_FORMAT_PCT</td>
</tr>
<tr>
<td>DFHPCXDF</td>
<td>DUXF_FORMAT_PPT</td>
</tr>
<tr>
<td>DFHSAXDF</td>
<td>DUXF_FORMAT_SIT</td>
</tr>
<tr>
<td>DFHDLXDF</td>
<td>DUXF_FORMAT_DLI</td>
</tr>
<tr>
<td>DFHPCXDF</td>
<td>DUXF_FORMAT_MODULE_INDEX</td>
</tr>
<tr>
<td>DFHXDxdf</td>
<td>DUXF_FORMAT_DUMP_TRAILER</td>
</tr>
</tbody>
</table>

**DUXW format, HEX function**

1. Construct record in buffer indicating that this data should be formatted as hexadecimal.
2. If buffer is full, call DUSU WRITE to output it.
3. If XDUOUT exit is active, call APEX INVOKE_USER_EXIT.

**DUXW format, NON_HEX function**

1. Construct record in buffer indicating that this data should be printed as-is; that is, it is already a character string.
2. If buffer is full, call DUSU WRITE to output it.
3. If XDUOUT exit is active, call APEX INVOKE_USER_EXIT.

**Dump domain’s generic gates**

Table 47 summarizes the dump domain’s generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMDM</td>
<td>DU 0001</td>
<td>PRE_INITIALISE</td>
<td>DMDM</td>
</tr>
<tr>
<td></td>
<td>DU 0002</td>
<td>INITIALISE_DOMAIN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>QUIESCE_DOMAIN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TERMINATE_DOMAIN</td>
<td></td>
</tr>
<tr>
<td>APUE</td>
<td>DU 0301</td>
<td>SET_EXIT_STATUS</td>
<td>APUE</td>
</tr>
<tr>
<td></td>
<td>DU 0302</td>
<td></td>
<td></td>
</tr>
<tr>
<td>STST</td>
<td>DU 0500</td>
<td>COLLECT_STATISTICS</td>
<td>STST</td>
</tr>
<tr>
<td></td>
<td>DU 0501</td>
<td>COLLECT_RESOURCE_STATS</td>
<td></td>
</tr>
</tbody>
</table>

For descriptions of these functions and their input and output parameters, refer to the sections dealing with the corresponding generic formats:
In preinitialization processing, the dump domain establishes the initial dumping status:

- System dumping is enabled or suppressed, as required.
- The next transaction dump data set to be used is flagged.
- The transaction dump data set autoswitch status is set on or off, as required.
- The dump retry interval is established.
- The system dump table is initialized to empty.

For a cold start, the information comes from the system initialization parameters; for any other type of start, the information comes from the local catalog, but is then modified by any relevant system initialization parameters.

In initialization processing, the dump domain loads the transaction dump table and the system dump table from the global catalog.

In quiesce processing, the dump domain performs only internal routines.

In termination processing, the dump domain closes the transaction dump data set.

**DMDM PRE_INITIALIZE function**

The PRE_INITIALIZE function of the DMDM gate performs the following functions:

1. Issue MVS GETMAIN for DU anchor block (DUA) and initialize it.
2. Read DU state record from the local catalog and set values in the DUA.
3. Initialize to empty the system dump table.
5. Acquire startup information from the parameter manager (PA) domain and set it in the DUA.
6. Inform the kernel that DU system dump is available by issuing KEDD ADD_GATE for the DFHDUDU gate.

**DMDM INITIALIZE_DOMAIN function**

The INITIALIZE_DOMAIN function of the DMDM gate performs the following functions:

1. Load the system dump table from the global catalog.
2. Load the transaction dump table from the global catalog.
3. Issue LMLM ADD_LOCK for the dump data set lock (DUDATSET).
4. Issue LMLM ADD_LOCK for the dump table lock (DUTABLE).
5. Issue LMLM UNLOCK for DUTABLE lock.
6. Issue KEDD ADD_GATE for the DU STST, DUDT, and APUE gates.
Dump domain (DU)

7. Initialize transaction dump, including loading DFHDUIO, and indicate that the
dump table is available to the DUDU TRANSACTION_DUMP function.
8. Update DU state record on catalog.
9. Issue LMLM UNLOCK for DUDATSET lock, thereby making the transaction
dump function available.

DMDM QUIESCE_DOMAIN function

The QUIESCE_DOMAIN function of the DMDM gate issues a DMDM
WAIT_PHASE function request to ensure all statistics are collected.

DMDM TERMINATE_DOMAIN function

The TERMINATE_DOMAIN function of the DMDM gate issues a DUSU CLOSE
request to close the transaction dump data set.

APUE SET_EXIT_STATUS function

The SET_EXIT_STATUS function of the APUE gate sets the exit status flag in the
DUA for the specified exit.

STST COLLECT_STATISTICS function

The COLLECT_STATISTICS function of the STST gate is called from the statistics
domain. The process flow is:
1. Issue LMLM LOCK for DUTABLE lock on the transaction dump table.
2. Acquire KE system dump lock.
3. Issue STST COLLECT_STATISTICS call to DFHDUTM.
4. Release DUTABLE lock and system dump lock.

DFHDUTM process flow

If the COLLECT_STATISTICS parameters requested DATA, the following statistics
records are written to the statistics domain:
1. If the RESOURCE_TYPE is not specified or is SYSDUMP, a DFHSDGPS global
system dump statistics record is created, using global system dump counts
(taken and suppressed) from the DUA. The KE system lock is released while a
STATS_PUT request is made to the statistics domain. The lock is obtained again
on successful completion of the STATS_PUT.
2. If the RESOURCE_TYPE is not specified or is TRANDUMP, a DFHTDGPS
global transaction dump statistics record is created, using global transaction
dump counts (taken and suppressed) from the DUA. The DUTABLE lock is
released while a RECORD_STATISTICS request is made to the statistics
domain. The lock is obtained again on successful completion of the
RECORD_STATISTICS.
3. If the RESOURCE_TYPE is not specified or is SYSDUMP, a DFHSDRPS
statistics detail record is written for every dump code found on the system
dump table. The records contain the statistics for that dump code held on the
dump table entry. The DFHSDRPS records are buffered and full buffers are
written out using a RECORD_STATISTICS call to the statistics domain.
4. If the RESOURCE_TYPE is not specified or is TRANDUMP, a DFHTDRPS
statistics detail record is written for every dump code found on the transaction
dump table. The records contain the statistics for that dump code held on the
dump table entry. The DFHTDRPS records are buffered and full buffers are
written out using a RECORD_STATISTICS call to the statistics domain.
The global system and transaction dump counts (taken and suppressed) in the DUA are also reset to zero. The last_reset_time is also updated in the DUA at this time.

**STST COLLECTRESOURCE_STATS function**

The COLLECTRESOURCE_STATS function of the STST gate is called from an EXEC CICS command. The process flow is:

1. Issue LMLM LOCK for DUTABLE lock on the transaction dump table.
2. Acquire KE system dump lock.
3. Issue STST COLLECTRESOURCE_STATS call to DFHDUTM.
4. Release DUTABLE lock and system dump lock.

**DFHDUTM process flow**

1. Validate RESOURCE_TYPE for either SYSDUMP or TRANDUMP. Perform error processing and return INVALID to the caller if it is neither of these.
2. If the RESOURCE_ID has not been passed, format a global statistics record, using counts of dumps taken and suppressed from the DUA, for either system or transaction dumps, depending on the RESOURCE_TYPE. Return this record to the caller in the RESOURCE_STATISTICS_DATA parameter.
3. If the RESOURCE_ID is present, it should contain a dump code. Search the relevant dump table (depending on RESOURCE_TYPE). Return ID_NOT_FOUND exception to the caller if the dump code cannot be found. If the dump code is found, format either a DFHTDRPS or a DFHSDRPS statistics record using the dumps taken and suppressed statistics on the dump table entry. This record is formatted in the next available space in the RESOURCE_STATISTICS_DATA buffer.

---

**Control blocks**

**Dump domain anchor block (DUA)**

There is one DU anchor block in the system. It is created when DU is initialized, and lasts for the lifetime of the system. It contains information relating to the status of the domain, and pointers to other control blocks.

**Dump domain open block.**

This contains the data areas associated with the dump data set DCB, namely the ECB, DCB itself, DECB, and the output buffer. It resides below the 16MB line. It is allocated when the data set is opened, and freed when either an explicit close is issued or the end of the current data set is reached and autoswitching is not active.

**System dump table (SDT)**

Storage for this table is obtained during dump domain preinitialization. The table is then initialized with null table entries. During dump domain initialization, the table is loaded with any values held on the global catalog for system dump codes that were explicitly added during previous CICS runs. Any system dumps taken before this point in initialization use default dump values (see the CICS Problem Determination Guide for information held for each dump code, and the default values).

Table entries are added during a CICS run either explicitly via CEMT or EXEC CICS commands, or implicitly, with default values, if a dump is requested for which an entry does not exist. These entries can be changed or deleted via CEMT or EXEC CICS commands. Explicitly added entries...
Dump domain (DU)

are written to the global catalog. Further blocks of storage are obtained if necessary as each block fills up. Storage for deleted entries is not reused, because activity on the table is low.

The DU domain anchor block contains pointers to the table, to the first and last active entries in the table, and to the next available entry. The table contains forward and backward pointers so that the table can be accessed in dump code sequence, and additional blocks are chained off the header of the previous block.

Transaction dump table (TDT)

Storage for this table is obtained during dump domain initialization and the table is then loaded with any values held on the global catalog which were explicitly added during previous CICS runs.

Table entries are added during a CICS run either explicitly via CEMT or EXEC CICS commands, or implicitly, with default values, if a dump is requested for which an entry does not exist. These entries can be changed or deleted via CEMT or EXEC CICS commands. Explicitly added entries are written to the global catalog. Further blocks of storage are obtained if necessary as each block fills up. Storage for deleted entries is not reused, because activity on the table is low.

The DU domain anchor (DUA) block contains pointers to the table, to the first and last active entries in the table, and to the next available entry. The table contains forward and backward pointers so that the table can be accessed in dump code sequence, and additional blocks are chained off the header of the previous block.

Browse token table (BTT)

This table holds browse tokens for both system and transaction dump tables. Each browse session started on either dump table is allocated a token that is held in this table, along with the dump code of the last dump table entry obtained by the browse session.

Storage for this table is obtained when the first dump table browse session of a CICS run is started. More storage is obtained when the table is full. Storage for deleted entries is not reused.

The structure of the table is the same as for the dump tables, as shown in Figure 52 on page 413.
Notes:
1. This example is for the transaction dump table, but it also applies to the SDT and the BTT.
2. The global catalog contained records for dump codes AAAA, BBBB, CCCC, DDDD, and FFFF.
3. Dump code BBBB has been deleted by an EXEC CICS command, so the NEXT and PREV pointers have been set to zero.
4. Dump code EEEE has been added during this CICS run and the pointers in entries for DDDD and FFFF adjusted to include EEEE in the correct sequence.
5. In this example, the first table block is not full, so TDT_NEXT in the block header is zero.

For a detailed description of these control blocks, see the CICS Data Areas manual.

Modules

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHAPTRV</td>
<td>System dump formatting program, ZC Install</td>
</tr>
<tr>
<td>DFHAPTRY</td>
<td>System dump formatting program, XM related</td>
</tr>
<tr>
<td>DFHAPTRX</td>
<td>System dump formatting program, ZC persistent sessions</td>
</tr>
<tr>
<td>DFHDUDDM</td>
<td>Processes requests to the DMDM gate of the dump domain</td>
</tr>
<tr>
<td>DFHDUDT</td>
<td>Processes requests to the DUDT gate of the dump domain</td>
</tr>
<tr>
<td>DFHDUDU</td>
<td>Processes requests to the DUDU gate of the dump domain</td>
</tr>
<tr>
<td>DFHDUIO</td>
<td>Processes domain subroutine requests of format DUIO</td>
</tr>
<tr>
<td>DFHDUSR</td>
<td>Processes requests to the DUSR and APUE gates of the dump domain</td>
</tr>
</tbody>
</table>
Dump domain (DU)

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHDUSU</td>
<td>Processes domain subroutine requests of format DUSU</td>
</tr>
<tr>
<td>DFHDUSVC</td>
<td>System dump</td>
</tr>
<tr>
<td>DFHDUTM</td>
<td>Dump table manager</td>
</tr>
<tr>
<td>DFHDUXD</td>
<td>Invoked by DFHDUDU with a DUDD format parameter list to control the transaction dump process</td>
</tr>
<tr>
<td>DFHDUXW</td>
<td>Processes domain subroutine requests of format DUXW</td>
</tr>
</tbody>
</table>

Transaction dump formatting routines

The following routines are invoked by DFHDUXD to dump the storage areas associated with a particular CICS component. They are passed a DUXF format parameter list. They are all part of the DFHSIP load module.

<table>
<thead>
<tr>
<th>Routine</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHDLXDF</td>
<td>DL/I related areas</td>
</tr>
<tr>
<td>DFHFCXDF</td>
<td>File control related areas</td>
</tr>
<tr>
<td>DFHPCXDF</td>
<td>Program related areas</td>
</tr>
<tr>
<td>DFHSAXDF</td>
<td>Common areas such as CSA, TCA, and so on</td>
</tr>
<tr>
<td>DFHSMXDF</td>
<td>Task subpools</td>
</tr>
<tr>
<td>DFHTCXDF</td>
<td>Terminal control related areas</td>
</tr>
<tr>
<td>DFHTRXDF</td>
<td>The internal trace table</td>
</tr>
<tr>
<td>DFHXDXDF</td>
<td>Information such as register contents, headers, and so on</td>
</tr>
<tr>
<td>DFHXRXDF</td>
<td>XRF related areas</td>
</tr>
</tbody>
</table>

Copy books

<table>
<thead>
<tr>
<th>Copy book</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHDUDCC</td>
<td>Contains the definitions of all DU control blocks.</td>
</tr>
<tr>
<td>DFHDUXDC</td>
<td>Provides common definitions for the transaction dump formatting routines DFHxxXDF.</td>
</tr>
<tr>
<td>DFHDUXDS</td>
<td>Common routine for the transaction dump formatting routines to convert responses from DFHDUXW into responses for DFHDUXD.</td>
</tr>
<tr>
<td>DFHDUXDV</td>
<td>Common abend recovery routine for the transaction dump formatting routines.</td>
</tr>
</tbody>
</table>

Exits

The dump domain exits are listed below. See the [CICS Customization Guide](https://www.ibm.com) for details of each exit.

**XDUREQ**

The dump request exit, driven for each transaction and system dump request.

**XDUREQ**

The dump request close exit, driven after a transaction or system dump has been taken (or failed or suppressed).
Dump domain (DU)

**XDUOUT**
The output exit, driven before each buffer is written to the transaction dump data set.

**XDUCLSE**
The dump data set close exit, driven after each close of a transaction dump data set.

---

**Trace**

The point IDs for the dump domain are of the form DU xxxx; the corresponding trace levels are DU 1, DU 2, and Exc.

For more information about the trace points, see the [CICS Trace Entries](#). For more information about using traces in problem determination, see the [CICS Problem Determination Guide](#).

---

**Dumps**

A formatted system dump contains the DU anchor (DUA) block and the DU open block.

System dumps requested by DU fall into two categories:

**Dump code DUnnnn**
For these there is a preceding console message DFHDUnnnn. See the [CICS Messages and Codes](#) manual for details.

**Dump code KERNDUMP**
If an error occurs in DFHDUDM during PRE INITIALIZE processing, that is, before the system dump function is available, DU uses MVS WTO to write message DFHDU0103, and the kernel dump function to take an SDUMP.
Dump domain (DU)
Chapter 30. Dump utility program (DFHDU530)

The dump utility program (DFHDU530) runs offline (in batch mode) to produce a printout of the CICS transaction dumps from a CICS transaction dump data set (DFHDMPA or DFHDMPB).

Design overview

DFHDU530 operates in batch mode while one of the dump data sets is closed. Each area, program, and table entry is identified, formatted, and printed separately, with both actual and relative addresses to facilitate analysis. You can select single or double spacing of dumps when the dump utility program is executed.

The CICS dump data set (DFHDMPA or DFHDMPB) contains a number of CICS transaction dumps. These are produced as the result of a transaction abend or a user-application EXEC CICS DUMP TRANSACTION request.

DFHDU530 runs as a stand-alone program in batch mode to format and print the contents of a transaction dump data set. Parameters specified on the SYSIN data set can be used to print only selected dumps or an index of the dumps in the data set.

For further details about DFHDU530, see the CICS Operations and Utilities Guide.

Data sets

There are three sources of data for DFHDU530:

Parameters on JCL EXEC statement
A character string of keywords that can be specified to control the layout and format of the dumps.

SYSIN
Records specifying the criteria to be used in selecting which of the dumps on the data set are to be printed.

DFHDMPS
The transaction dump data set.

There are two output files:

DFHPRINT
The print file for the formatted transaction dump.

DFHTINDX
The print file for the index of dumps on the data set.

Processing

Figure 53 on page 420 shows the dump utility program interfaces.
The overall flow of the processing within DFHDU530 is as follows. Unless otherwise indicated, all processing is performed by DFHDUPR, the main component of DFHDU530.

1. Process the EXEC parameters if they are present.
2. Call DFHDUPP to open the print data set DFHPRINT.
3. Open the dump data set DFHDMPDS.
4. Read the dumps from DFHDMPDS. For each dump there are four categories of records:

   **Dump header record**
   Call DFHDUPS to see whether this dump is required for printing. On the first time through, DFHDUPS reads the selective print information from SYSIN. DFHDUPS also calls DFHDUPH to add the dump to the dump index data set DFHTINDX. DFHDUPH opens DFHTINDX on its first invocation.

   **Module index records**
   DFHDUPM is called to accumulate the module index information in a table in main storage.

   **Other data records**
   The data is formatted into print lines and DFHDUPP is invoked to write them to DFHPRINT.

   **Dump trailer record**
   DFHDUPM is invoked to sort and format the module index records. DFHDUPP is called to write them to DFHPRINT.

5. When the end of the dump data set is encountered:
   a. DFHDUPP is called to close DFHPRINT.
   b. DFHDUPH is called to close DFHTINDX.
   c. DFHDUPR closes DFHDMPDS.
6. DFHDU530 terminates.

### Modules

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHDUPR</td>
<td>Controlling routine, responsible for reading information from the dump data set DFHDMPDS.</td>
</tr>
</tbody>
</table>
Module | Function
---|---
DFHDUPS | Receives the address of a dump header record from the dump data set, and decides whether this dump fulfills the criteria for printing. On first entry, reads and stores the selective print parameters from SYSIN.
DFHDUPP | Is responsible for all access to the print file DFHPRINT, namely for OPEN, CLOSE, and PUT requests.
DFHDUPH | Writes line to dump index for each dump header record encountered. On first entry, opens the index file DFHTINDX.
DFHDUPM | Invoked for each module index entry found to save information. Invoked when dump trailer record found to format and print the complete module index.

Copy books

<table>
<thead>
<tr>
<th>Copy book</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHDUPSC</td>
<td>Contains the definition of the parameter list passed to DFHDUPS.</td>
</tr>
<tr>
<td>DFHDUPMC</td>
<td>Contains the definition of the parameter list passed to DFHDUPM.</td>
</tr>
<tr>
<td>DFHDUPPC</td>
<td>Contains the definition of the parameter list passed to DFHDUPP.</td>
</tr>
</tbody>
</table>

Exits

Global user exit points are not applicable to offline utilities.

Trace

Trace points are not applicable to offline utilities.
Chapter 31. Dynamic allocation sample program (IBM 3270 only)

Any data set defined to file control can be allocated to CICS dynamically when the file is opened, rather than at CICS job initiation time. This allocation takes place automatically if job control statements for the data set are not included in the CICS job stream, and if both the data set name and the disposition have been specified in the file control table when the data set is opened.

The dynamic allocation sample program provides an alternative way to perform dynamic allocation. When used with a terminal of the IBM 3270 Information Display System, it gives the user access to the functions of DYNALLOC (SVC 99) in MVS. This can be used, in conjunction with master terminal functions and suitable operating procedures, to allocate and deallocate any file that CICS can dynamically open and close.

Design overview

The program runs as a CICS transaction, using CICS functions at the command level wherever possible. It does not modify any CICS control blocks. Only the DYNALLOC function is available through the program; any manipulation of the environment before or after the DYNALLOC request must be done by other means.

CICS supplies sample resource definitions for the program load module, DFH99, and the transaction, ADYN, that invokes it. These definitions are in the group DFHSUTIL. Note that DFH99 must be defined with EXECKEY(CICS).

The flow in a normal invocation is as follows. The main program, DFH99M, receives control from CICS, and carries out initialization. This includes determining the screen size and allocating input and output buffer sections, and issuing initial messages. It then invokes DFH99GI to get the input command from the terminal. Upon return, if the command was null, the main program terminates, issuing a final message.

The command obtained has its start and end addresses stored in the global communication area, COMM. The main program allocates storage for tokenized text, and calls DFH99TK to tokenize the command. If errors were detected at this stage, further analysis of the command is bypassed.

Following successful tokenizing, the main program calls DFH99FP to analyze the verb keyword. DFH99FP calls DFH99LK to look up the verb keyword in the table, DFH99T. DFH99LK calls DFH99MT if an abbreviation is possible. Upon finding the matching verb, DFH99FP puts the address of the operand section of the table into COMM, and puts the function code into the DYNALLOC request block.

The main program now calls DFH99KO to process the operand keywords. Each keyword in turn is looked up in the table by calling DFH99LK, and the value coded for the keyword is checked against the attributes in the table. DFH99KO then starts off a text unit with the appropriate code, and, depending on the attributes the value should have, calls a conversion routine.
Dynamic allocation sample program

- For character and numeric strings, DFH99CC is called. It validates the string, and puts its length and value into the text unit.
- For binary variables, DFH99BC is called. It validates the value, converts it to binary of the required length, and puts its length and value into the text unit.
- For keyword values, DFH99KC is called. It looks up the value in the description part of the keyword table using DFH99LK, and puts the coded equivalent value and its length into the text unit.

When a keyword specifying a returned value is encountered, DFH99KO makes an entry on the returned value chain, which is anchored in COMM. This addresses the keyword entry in DFH99T, the text unit where the value is returned, and the next entry. In this case the conversion routine is still called, but it only reserves storage in the text unit, setting the length to the maximum and the value to zeros.

When all the operand keywords have been processed, DFH99KO returns to the main program, which calls DFH99DY to issue the DYNALLOC request.

DFH99DY sets up the remaining parts of the parameter list, and if no errors too severe have been detected, a subtask is attached to issue the DYNALLOC SVC. A WAIT EVENT is then issued against the subtask termination ECB. When the subtask ends, and CICS dispatches the program again, the DYNALLOC return code is captured from the subtask ECB, with the error and reason codes from the DYNALLOC request block, and a message is issued to give these values to the terminal.

DFH99DY then returns to the main program, which calls DFH99RP to process returned values. DFH99RP scans the returned value chain, and for each element issues a message containing the keyword and the value found in the text unit. If a returned value corresponds to a keyword value, DFH99KR is called to look up the value in the description, and issue the message.

Processing of the command is now complete, and the main program is reinitialized for the next one, and loops back to the point where it calls DFH99GI.

Messages are issued at many places, using macros. The macro expansion ends with a call to DFH99MP, which ensures that a new line is started for each new message, and calls DFH99ML, the message editor. Input to the message editor is a list of tokens, and each one is picked up in turn and converted to displayable text. For each piece of text, DFH99TX is called, which inserts the text into the output buffer, starting a new line if necessary. This ensures that a word is never split over two lines.

When the command has been processed, the main program calls DFH99MP with no parameters, which causes it to send the output buffer to the terminal, and initialize it to empty.

Control blocks

The sample program does not have any control blocks.

Modules

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFH99BC</td>
<td>Convert to binary target</td>
</tr>
</tbody>
</table>
Module Function
DFH99CC Character and number string conversion
DFH99DY Issue SVC 99 and analyze result
DFH99FP Process function keyword
DFH99GI Format display and get input
DFH99KC Keyword value conversion
DFH99KH List keywords for help
DFH99KO Process operator keywords
DFH99KR Convert returned value to keyword
DFH99LK Search key set for given token
DFH99ML Build message text from token list
DFH99MM Main control program (entry point DFH99M)
DFH99MP Message filing routine
DFH99MT Match abbreviation with keyword
DFH99RP Process returned values
DFH99T Table of keywords
DFH99TK Tokenize input command
DFH99TX Text display routine
DFH99VH List description for help

Exits

No global user exit points are provided for this function.

Trace

This sample program makes no entries in the trace, over and above the normal entries one would see for a CICS user transaction.

External interfaces

SVC 99—MVS DYNALLOC SVC.
Chapter 32. Enqueue Domain (NQ)

The NQ domain provides UOW based locking services. This is provided to the local clients FC, TD and TS. It also services the EXEC CICS ENQ/DEQ requests.

The most common functions provided by the NQ domain are:

**CREATE_ENQUEUE_POOL**
This function creates a separate enqueue pool for the caller. A token is returned which the caller specifies on all requests associated with that pool.

**DEACTIVATE**
This function converts an active enqueue into retained state. The caller must already own the enqueue.

**REACQUIRE_ENQUEUE**
NQ domain doesn’t recover enqueues over a CICS restart. Instead resource owners use this function to reacquire enqueues that were held by inflight and indoubt UOWs.

**ENQUEUE**
This function obtains an enqueue from the specified enqueue pool in active state.

**DEQUEUE**
This function releases an active enqueue owned by the current UOW from the specified enqueue pool.

**INQUIRE_NQRNAME**
This function calls INQ_NQRNAME to see if an enqueue name entry exists in NQRNAME_LIST. If the name is either an exact or generic match, INQUIRE_NQRNAME returns the 4-character SCOPE name, enqmodel STATE and ann OK RESPONSE. Otherwise it returns an EXCEPTION REASON(NQRNAME_NOT_FOUND).

**ADD_REPLACE_ENQMODEL**
This function adds an enqmodel definition to both the NQRN directory (keyed by enqmodel name, and to the NQRNAME_LIST (keyed by the variable length NQRNAME). If the enqmodel already exists the entry is replaced.

**DISCARD_ENQMODEL**
Remove an enqmodel definition from both the NQRN directory and from the NQRNAME_LIST. If the enqmodel is not installed, exception ‘ENQMODEL_NOT_FOUND’ is returned.

**INQUIRE_ENQMODEL**
Uses directory DDLO_LOCATE to retrieve information about a specified enqmodel definition in the NQRN directory.

If found, it returns the 1 to 255 character NQRNAME, the 4-character SCOPE name, the enqmodel STATE and ann OK RESPONSE. Otherwise it returns an EXCEPTION REASON(ENQMODEL_NOT_FOUND).

**SET_ENQMODEL**
This function uses directory DDLO_LOCATE to see if an enqmodel entry exists in the NQRN directory. If found, it enables or disables the entry.
Otherwise it returns an EXCEPTION REASON(ENQMODEL_NOT_FOUND).

Enqueue domain’s specific gates

Table 48 summarizes the NQ domain’s specific gate. It shows the level-1 trace point IDs of the modules providing the functions for the gate and the functions provided by the gate. The DFHNQEDX XPI macro provides ENQUEUE and DEQUEUE functions for the NQ domain.

Table 48. NQ domain’s specific gates

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>NQNQ</td>
<td>NQ 0201</td>
<td>CREATE_ENQUEUE_POOL</td>
</tr>
<tr>
<td></td>
<td>NQ 0202</td>
<td>DEACTIVATE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>REACQUIRE_ENQUEUE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SET_NQRNAME_LIST</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DEQUEUE_TASK</td>
</tr>
<tr>
<td>NQED</td>
<td>NQ 0301</td>
<td>ENQUEUE</td>
</tr>
<tr>
<td></td>
<td>NQ 0302</td>
<td>DEQUEUE</td>
</tr>
<tr>
<td>NQIB</td>
<td>NQ 0401</td>
<td>INQUIRE_ENQUEUE</td>
</tr>
<tr>
<td></td>
<td>NQ 0402</td>
<td>START_BROWSE_ENQUEUE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GET_NEXT_ENQUEUE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>END_BROWSE_ENQUEUE</td>
</tr>
<tr>
<td>NQRN</td>
<td>NQ 0601</td>
<td>INQUIRE_NQRNAME</td>
</tr>
<tr>
<td></td>
<td>NQ 0602</td>
<td>ADD_REPLACE_ENQMODEL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DISCARD_ENQMODEL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>REMOVE_ENQMODEL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INQUIRE_ENQMODEL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>START_BROWSE_ENQMODEL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GET_NEXT_ENQMODEL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>END_BROWSE_ENQMODEL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SET_ENQMODEL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COMMIT_ENQMODEL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RESTORE_DIRECTORY</td>
</tr>
<tr>
<td>NQIE</td>
<td>NQ FF50</td>
<td>INTERPRET_ENQUEUE</td>
</tr>
<tr>
<td></td>
<td>NQ FF51</td>
<td></td>
</tr>
</tbody>
</table>

NQNQ gate, CREATE_ENQUEUE_POOL function

This function creates a separate enqueue pool for the caller. A token is returned which the caller specifies on all requests associated with that pool.

Input parameters:

POOL_NAME
The eight character name of the new enqueue pool.

EXPECTED_NAME_LENGTH
The expected length for enqueue names in the pool.

For pools with fixed length enqueue names this should be the length of the names that are going to be enqueued upon.

For pools that are to contain variable length enqueue names this should be a length that would satisfy ‘most’ of the requests to be made in the pool.

Note that is no maximum length for enqueue names. However, requests will only be handled inline if the length of the enqueue name is less than
or equal to the EXPECTED_NAME_LENGTH. The inline macro only copes with names of less than or equal to 256 characters. For this reason an error will be diagnosed if a value of greater than 256 is specified for this parameter.

**SHUNT_ACTION**
Indicates the default action that is to be performed to UOW lifetime enqueues in this pool if their owning UOW is shunted. Note that most enqueue pools will require the same action to be performed for all enqueues in that pool. However, the ENQUEUE function allows this default to be overridden for particular enqueue requests.

The possible values are as follows:

**RELEASE**
The enqueue(s) will be released if the owning UOW is shunted.

**RETAIN**
The enqueue(s) will be retained if the owning UOW is shunted.

**IGNORE**
The shunt will be ignored. The enqueue(s) will remain in the same state as currently held in.

Transaction lifetime enqueues are automatically released when a shunt occurs.

**ERROR_LEVEL**
Indicates the severity of the error response that is to be returned for the following errors made while using this pool:

- **DEQUEUE**
  - Enqueue_not_owned
  - Enqueue_locked
- **REACQUIRE_ENQUEUE**
  - Enqueue_locked
  - Enqueue_active
- **DEACTIVATE**
  - Enqueue_not_owned
  - Enqueue_not_active

The possible values for ERROR_LEVEL are as follows:

**EXCEPTION_RESPONSE**
The above errors are to be returned with an exception response.

**INVALID_RESPONSE**
The above errors are to be returned with an invalid response. (i.e. FFDC is to be performed).

**Note:** It is expected that only the EXEC and the KC enqueue pools will specify EXCEPTION_RESPONSE since the DFHKC service previously used by them allowed these sorts of error to go by undetected.

**EXEC_INTERPRETER**
Indicates how enqueues belonging to the enqueue pool are to be interpreted by the EXEC CICS INQUIRE UOWENQ command.

The possible values are as follows:
Enqueue Domain (NQ)

NONE
No interpreter has been supplied so enqueues belonging to this pool will be ignored by the INQUIRE UOWENQ command.

DEFAULT
Enqueues are to be returned by the INQUIRE UOWENQ command. The default NQ domain interpreter will be called to perform the interpretation. This will map the outputs of the INQUIRE UOWENQ command as follows:

TYPE Will be the CVDA corresponding to the ENQUEUE_TYPE parameter supplied on this call.

RESOURCE
Will be ENQUEUE_NAME1 as supplied on the NQED_ENQUEUE function.

QUALIFIER
Will be ENQUEUE_NAME2 if supplied on the NQED_ENQUEUE function. If not then no QUALIFIER data will be returned.

OWN
Enqueues are to be returned by the INQUIRE UOWENQ command. A routine provided by the pool owner will perform the interpretation. In this case the entry point of the routine must be passed in the INTERPRETER_ADDR parameter.

Note: The routine will be called by a kernel subroutine call, not by a domain call. Consequently it will execute in the domain of the caller (i.e. AP domain).

OWN_INTERPRETER_ADDRESS
Entry point of interpreter routine for this pool. Should only be supplied for pools which specify a value of OWN for the EXEC_INTERPRETER parameter.

ENQUEUE_TYPE
The enqueue type that is to be returned by the default interpreter. Should only be supplied for pools which specify a value of DEFAULT for the EXEC_INTERPRETER parameter.

The possible values are as follows and these map onto the CVDA values for the TYPE field as detailed under the EXEC CICS INQUIRE UOWENQ command.

- DATASET
- EXECENQ
- EXECENQADDR
- EXECENQPplex
- FILE
- TDQUEUE
- TSQUEUE
- DISPATCHER

Output parameters:

POOL_TOKEN
Token returned which identifies the newly created enqueue pool.

RESPONSE
is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED
[REASON]
is returned when RESPONSE is DISASTER or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>INVALID</td>
<td>INTERPRETER_ADDR_EXPECTED, ENQUEUE_TYPE_EXPECTED, DUPLICATE_POOL_NAME, INVALID_NAME_LENGTH</td>
</tr>
</tbody>
</table>

**NQNQ gate, DEACTIVATE function**

This function converts an active enqueue into retained state. The caller must already own the enqueue.

**Input parameters:**

**POOL_TOKEN**
Token representing enqueue pool from which the enqueue is to be deactivated.

**ENQUEUE_TOKEN**
Token representing the enqueue that is to be deactivated.

Slightly better performance is achieved for callers that use the token method for this function.

**ENQUEUE_NAME1**
A block (addr,len) identifying the name of the enqueue to be deactivated.

Or alternatively identifies the prefix of the enqueue name which when combined with the ENQUEUE_NAME2 parameter forms the name of the enqueue to be deactivated.

**ENQUEUE_NAME2**
A block (addr,len) identifying the second half of the enqueue name.

**Output parameters:**

**RESPONSE**
is the domain’s response to the call. It can have any of these values: OK, EXCEPTION, DISASTER, INVALID, KERNERROR, PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>ENQUEUE_NOT_owned, ENQUEUE_NOT_ACTIVE</td>
</tr>
<tr>
<td>INVALID</td>
<td>ENQUEUE_NOT_owned, ENQUEUE_NOT_ACTIVE, TRANSACTION_ENQUEUE, INVALID_POOL_TOKEN</td>
</tr>
</tbody>
</table>

**NQNQ gate, REACQUIRE_ENQUEUE function**

NQ domain doesn’t recover enqueues over a CICS restart. Instead resource owners use this function to reacquire enqueues that were held by inflight and indoubt UOWs.
The enqueue can be reacquired in either active or retained state. The calling UOW must currently be shunted.

No MAX_LIFETIME input is provided since such enqueues are only ever associated with a single UOW.

The same rules as documented for the mainline ENQUEUE function apply to the shunt action that will be associated with the reacquired enqueue.

**Input parameters:**

**POOL_TOKEN**
Token representing enqueue pool from which the enqueue is to be allocated from.

**ENQUEUE_NAME1**
A block (addr,len) identifying the name of the enqueue.

Or alternatively identifies the prefix of the enqueue name which when combined with the ENQUEUE_NAME2 parameter forms the name being enqueued on.

**ENQUEUE_NAME2**
A block (addr,len) identifying the second half of the enqueue name.

**STATE**
The state that the enqueue is to be reacquired in.

The possible states are as follows:

**ACTIVE**
The enqueue is to be reacquired in active state.

**RETAINED**
The enqueue is to be reacquired in retained state.

**SHUNT_ACTION**
Indicates the action that is to be performed if the UOW reacquiring the enqueue is shunted again. This parameter acts as an override, if not supplied then the default shunt action specified when the pool was created is assumed for this request.

The possible overrides are as follows:

**RELEASE**
The enqueue will be released if the UOW is shunted again.

**RETAIN**
The enqueue will be retained if the UOW is shunted again.

**IGNORE**
The shunt will be ignored. The enqueue will remain in the same state as it is currently held in.

**Output parameters:**

**ENQUEUE_TOKEN**
Token returned to represent the enqueue that has been successfully reacquired.

**RESPONSE**
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED
[REASON] is returned when RESPONSE is DISASTER, Exception, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>ENQUEUE_LOCKED, ENQUEUE_ACTIVE</td>
</tr>
<tr>
<td>INVALID</td>
<td>ENQUEUE_LOCKED, ENQUEUE_ACTIVE, CALLER_NOT_SHUNTED, INVALID_POOL_TOKEN</td>
</tr>
</tbody>
</table>

**NQNQ gate, SET_NQRNAME_LIST function**

This function is called from three places in dfhnqn:

- **discard_enqmodel**
  If nqrmodel delete is set THEN the specified nqrmodel is removed from nqrname_list.

- **Add_replace_enqmodel**
  If nqrmodel add is set THEN the specified nqrmodel is added to nqrname_list.

- **set_nqrmodel**
  If neither delete or add is set THEN the specified nqrmodel is set disabled.

**Input parameters:**

- **MODEL_TOKEN**
  The address of the nqrmodel to be set or added to nqrname_list.

- **POOL_TOKEN**
  The pool to be searched for matching enqueues

- **POOL_TWO**
  An optional second pool to be searched for matching enqueues

**Output parameters:**

- **FREE_TOKEN**
  Address of Model being removed.

- **RESPONSE**
  is the domain’s response to the call. It can have any of these values: OK|EXCEPTION|DISASTER|PURGED

[REASON] is returned when RESPONSE is DISASTER, Exception or PURGED. Possible values are:

- **NQRMODEL_NOT_FOUND**
  The nqrmodel could not be found in nqrname_list

- **FREE_NQRMODEL**
  A nqrmodel has been removed and must be freemained. Its address is in free_token.

**NQED gate, ENQUEUE function**

This function obtains an enqueue from the specified enqueue pool in active state.
Enqueue Domain (NQ)

**Input parameters:**

**POOL_TOKEN**
Token representing enqueue pool from which the enqueue is to be allocated.

**ENQUEUE_NAME1**
A block (addr,len) identifying the name being enqueued on.

Or alternatively identifies the prefix of the enqueue name which when combined with the ENQUEUE_NAME2 parameter forms the name being enqueued on.

**ENQUEUE_NAME2**
A block (addr,len) identifying the second half of the enqueue name.

**MAX_LIFETIME**
Indicates the maximum duration that the enqueue is to be held for. The possible values are as follows:

- **UOW** The enqueue will be released if it is held when the current UOW commits. This is the default value when not supplied on the call.
- **TRANSACTION** The enqueue will be released if it is held when the last UOW in the current transaction commits.
- **DISPATCHER_TASK** The enqueue will be released if it is held when a DEQUEUE_ALL request is issued by the owning dispatcher task. This is the only value permitted when POOL_TOKEN is not supplied on the call.

**WAIT** Indicates whether the caller wishes to wait if the requested enqueue is currently held in the pool by a different UOW. The possible values are as follows:

- **YES** The caller will be suspended if the enqueue is busy. This is the default value when not supplied on the call.
- **NO** The ENQUEUE_BUSY exception is returned to the caller if the enqueue is busy.

Note that callers specifying WAIT(NO) should still expect to suspend for the NQ domain lock.

**SHUNT_ACTION** Indicates the action that is to be performed if this UOW is shunted whilst it owns the enqueue. This parameter acts as an override, if not supplied then the default shunt action specified when the pool was created is assumed for this enqueue request.

The shunt action is only applicable to UOW lifetime enqueues. An error is diagnosed if this parameter is supplied on a request for a transaction lifetime enqueue.

The possible overrides are as follows:

- **RELEASE** The enqueue will be released if the UOW is shunted.
- **RETAIN** The enqueue will be retained if the UOW is shunted.
The shunt will be ignored. The enqueue will remain in the same state as it is currently held in.

Output parameters:

**ENQUEUE_TOKEN**
Token returned to represent the enqueue that has been successfully returned.
The token can then be used on the corresponding DEQUEUE request.

**DUPLICATE_REQUEST**
When an OK is returned this indicates whether the caller already owned the enqueue or not:

**YES** The caller already owned the enqueue.

**NO** The caller didn’t already own the enqueue.

**RESPONSE** is the domain’s response to the call. It can have any of these values:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>ENQUEUE_BUSY, ENQUEUE_LOCKED, ENQUEUE_DISABLED, LIMIT_EXCEEDED, SYSENQ_FAILURE</td>
</tr>
<tr>
<td>PURGED</td>
<td>TASK_CANCELLED, TIMED_OUT</td>
</tr>
<tr>
<td>INVALID</td>
<td>SHUNT_ACTION_NOTEXPECTED, INVALID_POOL_TOKEN</td>
</tr>
</tbody>
</table>

**NQED gate, DEQUEUE function**
This function releases an active enqueue owned by the current UOW from the specified enqueue pool.

Input parameters:

**POOL_TOKEN**
Token representing enqueue pool from which the enqueue is to be released.

**ENQUEUE_TOKEN**
Token representing the enqueue that is to be released.
Slightly better performance is achieved for callers that use the token method for releasing their enqueues.

**ENQUEUE_NAME1**
A block (addr, len) identifying the name of the enqueue being released.
Or alternatively identifies the prefix of the enqueue name which when combined with the ENQUEUE_NAME2 parameter forms the name of the enqueue being released.

**ENQUEUE_NAME2**
A block (addr, len) identifying the second half of the enqueue name.
Enqueue Domain (NQ)

**MAX_LIFETIME**
Indicates the maximum duration of the enqueue being released. The possible values are as follows:

- **UOW**  The enqueue was acquired with a duration of the current UOW. This is the default value when not supplied on the call.
- **TRANSACTION**  The enqueue was acquired with a duration of the last UOW of the current transaction.

**Output parameters:**

**RESPONSE**
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[**REASON**]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>ENQUEUE_NOT_OWNED, ENQUEUE_LOCKED</td>
</tr>
<tr>
<td>INVALID</td>
<td>ENQUEUE_NOT_OWNED, ENQUEUE_LOCKED, INVALID_POOL_TOKEN</td>
</tr>
</tbody>
</table>

**NQIB gate, INQUIRE_ENQUEUE function**
This functions returns information about a particular enqueue. Note that the pool containing the enqueue must be passed since it is a logical extension to the enqueue name.

For inquiries by token it is the caller’s responsibility to ensure that the enqueue which the token represents is still held.

**Input parameters:**

**POOL_TOKEN**  The token identifying the pool from which the enqueue being inquired about belongs.

**ENQUEUE_TOKEN**  Token representing the enqueue that is being inquired upon.

**ENQUEUE_NAME1**
A block (addr,len) identifying the name of the enqueue be inquired upon.

Or alternatively identifies the prefix of the enqueue name which when combined with the ENQUEUE_NAME2 parameter forms the name of the enqueue being inquired upon.

**ENQUEUE_NAME2**
A block (addr,len) identifying the second half of the enqueue name.

**Output parameters:**

**ENQUEUE_NAME_OUT**
A buffer into which the enqueue name is returned. The caller specifies the address and maximum length of the data area into which the enqueue name will be returned. If the enqueue name is too big for the buffer then
the data is truncated and an OK response is returned. The actual length of
the name is returned in enqueue_name_out_n.

Typically this parameter will only be of interest to callers inquiring by
enqueue token.

POOL_NAME
The name of the pool containing the enqueue.

STATE
The state that the enqueue is held in.

ACTIVE
The enqueue is held in active state.

RETAINED
The enqueue is held in retained state.

LOCAL_UOWID
The local UOWID of the UOW which owns the enqueue.

UOW_LIFETIME
The number of times the enqueue is held with UOW lifetime.

TRANSACTION_LIFETIME
The number of times the enqueue is held with TRANSACTION lifetime.

NUM_WAITERS
The number of transactions waiting for this enqueue.

NUM_LOCKED_FAILURES
Returns the number of failed requests for this enqueue whilst it is held in
retained state.

SHUNT_ACTION
The action that would be performed to this enqueue should its owning
UOW be shunted.

The possible values are as follows:

RELEASE
The enqueue will be released.

RETAIN
The enqueue will be retained.

IGNORE
The shunt will be ignored and the enqueue will remain in the same
state.

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK, EXCEPTION, DISASTER, INVALID, KERNERROR, PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID.
Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>ENQUEUE_NOT_FOUND</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_POOL_TOKEN</td>
</tr>
</tbody>
</table>
Enqueue Domain (NQ)

**NQIB gate, START_BROWSE_ENQUEUE function**

This function initiates a browse of all enqueues currently in the system or currently associated with a given UOW.

The browse returns both enqueue owners and enqueue waiters. The RELATION output parameter on GET_NEXT_ENQUEUE indicates whether the data being returned is associated with the enqueue owner or a UOW waiting for that enqueue.

When a system wide browse is initiated the first enqueue in the system is returned with RELATION(OWNER). If the enqueue has any waiters then the same enqueue will be returned again for each of the waiters but this time with RELATION(WAITER). The data returned will be that associated with that particular waiter. After the last waiter has been returned the next owned enqueue will be returned.

If the browse is restricted to only a particular UOW then only the enqueues that UOW owns will be returned. If the UOW is waiting for an enqueue this will also be returned.

The order in which the enqueues are returned is undefined, however enqueue waiters are always returned consecutively after their enqueue owner.

As with other types of CICS browses the state isn’t locked for the duration of the browse. Thus for example, there is no guarantee that the owner returned on a previous GET_NEXT_ENQUEUE is still the owner by the time each of its waiters are returned.

**Input parameters:**

`LOCAL_UOWID`

Identifies the unit of work if the browse is to be restricted to only those enqueues owned and being waited for by a particular UOW.

If omitted then browse will return all enqueue owners and waiters in the system.

`STABLE_ENQUEUES`

Specifies that the caller will complete the browse without issuing any further ENQ or DEQ requests. Applies only if LOCAL_UOWID is also specified and names the caller’s own UOWID.

**Output parameters:**

`BROWSE_TOKEN`

Token to be used by the caller on subsequent operations associated with this browse.

`RESPONSE` is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
</tbody>
</table>
NQIB gate, GET_NEXT_ENQUEUE function

This function returns information about the next enqueue owner or waiter in a browse.

**Input parameters:**

<table>
<thead>
<tr>
<th>BROWSE_TOKEN</th>
</tr>
</thead>
<tbody>
<tr>
<td>The token for the current browse.</td>
</tr>
</tbody>
</table>

**Output parameters:**

<table>
<thead>
<tr>
<th>ENQUEUE_NAME_OUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>A buffer into which the enqueue name is returned. The caller specifies the address and maximum length of the data area into which the enqueue name will be returned. If the enqueue name is too big for the buffer then the data is truncated and an OK response is returned. The actual length of the name is returned in enqueue_name_out_n.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RELATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Indicates whether the data being returned is associated with owner or a UOW waiting for the enqueue.</td>
</tr>
</tbody>
</table>

| OWNER |
| The data is associated with the owner of the returned enqueue. |

| WAITER |
| The data is associated with a waiter of the returned enqueue. |

| POOL_NAME |
| The name of the pool containing the enqueue. |

| STATE |
| The state that the enqueue is held in. |

| ACTIVE |
| The enqueue is held in active state. |

| RETAINED |
| The enqueue is held in retained state. |

| LOCAL_UOWID |
| The local UOWID of the UOW which owns or is waiting for the enqueue. |

| UOW_LIFETIME |
| For an enqueue returned with RELATION(OWNER) the number of times it is held with UOW lifetime. |

For an enqueue returned with RELATION(WAITER) a count of one indicates that the enqueue was requested with UOW lifetime.

| TRANSACTION_LIFETIME |
| For an enqueue returned with RELATION(OWNER) the number of times it is held with TRANSACTION lifetime. |

For an enqueue returned with RELATION(WAITER) a count of one indicates that the enqueue was requested with TRANSACTION lifetime.

| NUM_WAITERS |
| The number of transactions waiting for this enqueue. |
Enqueue Domain (NQ)

**NUM_LOCKED_FAILURES**
Returns the number of failed requests for this enqueue whilst it is held in retained state.

**SHUNT_ACTION**
The action that would be performed to this enqueue should its owning UOW be shunted.

The possible values are as follows:

**RELEASE**
The enqueue will be released.

**RETAIN**
The enqueue will be retained.

**IGNORE**
The shunt will be ignored and the enqueue will remain in the same state.

**INTERPRETER_ADDRESS**
The address of a routine which should be called with the INTERPRET_ENQUEUE function in order to interpret the enqueue for the EXEC CICS INQUIRE UOWENQ command.

If a zero address is returned then the enqueue isn’t to be returned by the INQUIRE UOWENQ command.

**POOL_TOKEN**
Token which identifies the pool which the enqueue owner or waiter belongs.

**ENQUEUE_NAME2_LENGTH**
The length of the second part of the enqueue name if the enqueue was originally specified in two parts (i.e. ENQUEUE_NAME1 and ENQUEUE_NAME2).

If the ENQUEUE_NAME2 parameter wasn’t originally specified for this enqueue then zero will be returned.

**ENQUEUE_TOKEN**
Token returned only when the enqueue is owned by the caller. Parameter is set to zero for all other enqueues returned on the browse.

**RESPONSE**
is the domain’s response to the call. It can have any of these values:
0K|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**[REASON]**
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>BROWSE_END</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_BROWSE_TOKEN</td>
</tr>
</tbody>
</table>

**NQIB gate, END_BROWSE_ENQUEUE function**
This function terminates a browse of the enqueues.
Input parameters:

BROWSE_TOKEN
   The token for the browse that is to be terminated.

Output parameters:

RESPONSE
   is the domain’s response to the call. It can have any of these values:
   OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
   is returned when RESPONSE is DISASTER or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_BROWSE_TOKEN</td>
</tr>
</tbody>
</table>

NQRN gate, ENQUEUE function

This function calls INQ_NQRNAME to see if an enqueue name entry exists in NQRNAME_LIST.

If the name is either an exact or generic match, INQUIRE_NQRNAME returns the 4-character SCOPE name, enqmodel STATE and ann OK RESPONSE. Otherwise it returns an EXCEPTION REASON(NQRNAME_NOT_FOUND).

Input parameters:

NQRNAME
   A buffer giving a 1 to 255 char name and length of the resource to be located.

MSG0105
   YES|NO, indicating whether message DFHNQ0105 is to be issued if the matching enqmodel is disabled or in the waiting state.

Output parameters:

SCOPE
   The 4-character scope identifier for the resource. Four blanks indicates that the enqueue has local scope.

STATE

   ENABLED
      Matching ENQ/DEQ requests should be processed.

   DISABLED
      Matching ENQ/DEQ requests should be rejected, and the issuing task abended abcode ENQ_DISABLED.

   WAITING
      Matching ENQ/DEQ requests should be rejected, and the issuing task abended abcode ENQ_DISABLED. There are INSTALL/CREATE/DISCARD requests waiting to be processed.

RESPONSE
   is the domain’s response to the call. It can have any of these values:
   OK|EXCEPTION|DISASTER
Enqueue Domain (NQ)

(REASON)

is returned when RESPONSE is DISASTER, EXCEPTION, PURGED or INVALID. Possible values are:

NQRNAME_NOT_FOUND
The name does not exist in the table.

ACQUIRE_LOCK_FAILED
Attempt to acquire a shared NQRNAME lock failed.

RELEASE_LOCK_FAILED
Attempt to release a shared NQRNAME lock failed.

NQRN gate, ADD_REPLACE_ENQMODEL function

This function adds an enqmodel definition to both the NQRN directory (keyed by enqmodel name, and to the NQRNAME_LIST (keyed by the variable length NQRNAME).

If the enqmodel already exists the entry is replaced. The replace is a discard then add operation.

If an attempt is made to create a deep enqmodel nesting, or if another enqmodel with the same nqrname is already installed, then msg NQ0106 is issued and a ‘DUPLICATE_NQRNAME’ exception is returned.

Input parameters:

CALLER
COLDINST, RDOINST or RESTART indicating A cold start, An online install or The input is in the MODEL_TOKEN respectively.

CATALOG
YES or NO indicating whether the record should be cataloged.

ENQMODEL
The 8-character identifier of the resource to be added.

MODEL_TOKEN
The address of the record obtained from the catalogue to be restored.

SCOPE
The 4-character scope identifier for the resource. If omitted or specified as blanks, matching ENQs will have LOCAL scope.

STATE
ENABLED | DISABLED is the state in which to install the enqmodel. If omitted, ENABLED is assumed.

NQRNAME
A buffer giving the 1 to 255 character name and length of the ENQ name or stem* to be added.

Output parameters:

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK | EXCEPTION | DISASTER | INVALID | PURGED

(REASON)

is returned when RESPONSE is DISASTER, EXCEPTION, PURGED or INVALID. Possible values are:
INVALID_PARAMETERS
One of the input parameters is invalid

DUPLICATE_NQRNAME
An attempt has been made to create a deep enqmodel nesting, or another enqmodel with the same nqrname is already installed.

DUPLICATE_ENABLED
An attempt to create an enabled enqmodel failed, because a less specific enqmodel is enabled.

CATALOG_WRITE_FAILED
COMMIT was specified but the record was not written to the catalogue.

GETMAIN_FAILED
The getmain for the NQRN storage failed.

DIRECTORY_ADD_FAILED
The DFHDDDIM ADD_ENTRY failed to add the ENQMODEL entry.

DIRECTORY_DELETE_FAILED
The DFHDDDIM DELETE_ENTRY failed to delete the ENQMODEL entry.

ACQUIRE_LOCK_FAILED
Attempt to acquire an exclusive NQRNAME lock failed.

RELEASE_LOCK_FAILED
Attempt to release an exclusive NQRNAME lock failed.

NQRN gate, DISCARD_ENQMODEL function
Remove an enqmodel definition from both the NQRN directory and from the NQRNAME_LIST.

If the enqmodel is not installed, an ‘ENQMODEL_NOT_FOUND’ exception is returned.

The ENQMODEL is put into the WAITING state until there are no enqueues in the local system which match the ENQNAME pattern. It is then removed from the local system.

Input parameters:
ENQMODEL
The 8-character identifier of the resource to be DELETED.

Output parameters:
RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER

[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

ENQMODEL_NOT_FOUND
The name is not in the NQRN directory.
Enqueue Domain (NQ)

CATALOG_DELETE_FAILED
An attempt to delete the ENQMODEL ENTRY from the GCD failed.

ACQUIRE_LOCK_FAILED
Attempt to acquire an exclusive NQRNAME lock failed.

RELEASE_LOCK_FAILED
Attempt to release an exclusive NQRNAME lock failed.

NQRN gate, INQUIRE_ENQMODEL function
Uses directory DDLO_LOCATE to retrieve information about a specified enqmodel definition in the NQRN directory.

If found, it returns the 1 to 255 character NQRNAME, the 4-character SCOPE name, the enqmodel STATE and ann OK RESPONSE. Otherwise it returns an EXCEPTION REASON(ENQMODEL_NOT_FOUND).

Input parameters:

ENQMODEL
The 8-character identifier of the entry to be returned.

Output parameters:

NQRNAME
A buffer returning the 1 to 255 character name and length of the ENQ name or generic stem*

SCOPE
Returns the 4-character scope identifier for the resource. Four blanks indicates that the enqueue has local scope.

STATE

ENABLED
Matching ENQ/DEQ requests should be processed.

DISABLED
Matching ENQ/DEQ requests should be rejected, and the issuing task abended abcode ENQ.Disabled.

WAITING
Matching ENQ/DEQ requests should be rejected, and the issuing task abended abcode ENQ.Disabled. There are INSTALL/CREATE/DISCARD requests waiting to be processed.

RESPONSE
is the domain’s response to the call. It can have any of these values: OK|EXCEPTION|DISASTER

[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

ENQMODEL_NOT_FOUND
The name does not exist in the directory.

DIRECTORY_LOCATE_FAILED
Directory DDLO_LOCATE failed with something other than NOT_FOUND.
ACQUIRE_LOCK_FAILED
Attempt to acquire a shared NQRNAME lock failed.

RELEASE_LOCK_FAILED
Attempt to release a shared NQRNAME lock failed.

NQRN gate, SET_ENQMODEL function
This function uses directory DDLO_LOCATE to see if an enqmodel entry exists in the NQRN directory. If found, it calls SET_ENQMODEL to enable or disable the entry. Otherwise it returns an EXCEPTION REASON(ENQMODEL_NOT_FOUND).

Enqmodels forming nested generic nqrnames must be enabled in order, from the most to the least specific. I.e. A more specific enqmodel may not be enabled if a less specific enqmodel is enabled. If attempted, msg NQ0107 is issued and EXCEPTION ‘DUPLICATE_ENABLED’ is returned to the caller.

You cannot enable/disable an enqmodel which is in the waiting state. If attempted, EXCEPTION ‘ENQMODEL_WAITING’ is returned to the caller.

Input parameters:

ENQMODEL
The 8-character identifier of the entry to be enabled/disabled.

STATE

ENABLED
The enqmodel is to be enabled.

DISABLED
The enqmodel is to be disabled.

Output parameters:

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER

[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

ENQMODEL_NOT_FOUND
The name does not exist in the directory.

ENQMODEL_WAITING
The enqmodel is in the WAITING state.

DUPLICATE_ENABLED
Attempt to enable/disable an enqmodel failed, because a less specific enqmodel is enabled.

DIRECTORY_LOCATE_FAILED
A DDLO_LOCATE failed with something other than NOT_FOUND.

CATALOG_UPDATE_FAILED
Attempt to update the enqmodel on the global catalog failed.

ACQUIRE_LOCK_FAILED
Attempt to acquire an exclusive NQRNAME lock failed.
Enqueue Domain (NQ)

**RELEASE_LOCK_FAILED**
Attempt to release an exclusive NQRNAME lock failed.

**NQIE gate, INTERPRET_ENQUEUE function**
This function interprets the passed enqueue prior to it being returned by the EXEC CICS INQUIRE UOWENQ command. The function takes the enqueue to be interpreted as input and returns ENQUEUE_TYPE, RESOURCE and QUALIFIER to the caller (EXEC layer).

Each enqueue pool can either
- not have an interpreter and consequently not have its enqueues returned by the INQUIRE UOWENQ command
- rely upon a default interpreter supplied by NQ domain, (DFHNQIE)
- supply its own interpreter routine.

This is specified when the pool is created.

**Input parameters:**

**POOL_NAME**
Name of the pool containing the enqueue to be interpreted.
Note that an interpreter may interpret enqueues from more than one pool.

**POOL_TOKEN**
Token corresponding to the pool containing the enqueue to be interpreted

**ENQUEUE_NAME**
A block (addr,len) identifying the full name of the enqueue to be interpreted.

**ENQUEUE_NAME2_LENGTH**
The length of the second part of the enqueue name if the enqueue was originally specified in two parts (i.e. ENQUEUE_NAME1 and ENQUEUE_NAME2).

If the ENQUEUE_NAME2 parameter wasn't originally specified for this enqueue then this will contain zero.

**Output parameters:**

**RESOURCE_BUFFER**
A buffer into which the data for the RESOURCE field is returned. The caller specifies the address and maximum length of the data area into which the RESOURCE data will be returned. If the data is too big for the buffer then the data is truncated and an OK response is returned. The actual length of the name is returned in resource_buffer_n.

**QUALIFIER_BUFFER**
A buffer into which the data for the QUALIFIER field is returned. The caller specifies the address and maximum length of the data area into which the QUALIFIER data will be returned. If the data is too big for the buffer then the data is truncated and an OK response is returned. The actual length of the name is returned in qualifier_buffer_n.

If there is no QUALIFIER data then no data should be returned and the length of the data (qualifier_buffer_n) be returned as zero.

**ENQUEUE_TYPE**
The TYPE of the enqueue being returned.
The possible values are as follows and these map onto the CVDA values for the TYPE field as detailed under the EXEC CICS INQUIRE UOWENQ command.

- DATASET
- EXECENQ
- EXECENQADDR
- FILE
- TDQUEUE
- TSQUEUE

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

**[REASON]**

is returned when RESPONSE is DISASTER or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_ENQUEUE</td>
</tr>
</tbody>
</table>

### Enqueue domain’s generic gates

Table 49 summarizes the NQ domain’s generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

**Table 49. NQ domain’s generic gates**

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMDM</td>
<td>NQ 0101</td>
<td>INITIALISE_DOMAIN</td>
<td>DMDM</td>
</tr>
<tr>
<td></td>
<td>NQ 0102</td>
<td>QUIESCE_DOMAIN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TERMINATE_DOMAIN</td>
<td></td>
</tr>
<tr>
<td>STST</td>
<td>NQ 0501</td>
<td>COLLECT_STATISTICS</td>
<td>STST</td>
</tr>
<tr>
<td></td>
<td>NQ 0502</td>
<td>COLLECT_RESOURCE_STATS</td>
<td></td>
</tr>
<tr>
<td>RMRO</td>
<td>NQ 0201</td>
<td>PERFORM_PREPARE</td>
<td>RMRO</td>
</tr>
<tr>
<td></td>
<td>NQ 0202</td>
<td>PERFORM_COMMIT</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PERFORM_SHUNT</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PERFORM_UNSHUNT</td>
<td></td>
</tr>
</tbody>
</table>

For descriptions of these functions and their input and output parameters, refer to the §§ dealing with the corresponding generic formats:

**Functions and parameters**


Format STST—“System programming command flows” on page 658.

Format RMRO—“Chapter 68. Recovery Manager Domain (RM)” on page 973.

PERFORM_PREPARE is a no-op. PERFORM_COMMIT releases enqueues. PERFORM_SHUNT makes active enqueues retained. PERFORM_UNSHUNT makes retained enquires active.
Enqueue Domain (NQ)

The Domain Manager gates perform normal internal state initialisation and termination functions.

### Modules

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHNQDM</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>INITIALISE_DOMAIN</td>
</tr>
<tr>
<td></td>
<td>QUIESCE_DOMAIN</td>
</tr>
<tr>
<td></td>
<td>TERMINATE_DOMAIN</td>
</tr>
<tr>
<td>DFHNQDUF</td>
<td>Formats the NQ domain control blocks in a CICS system.</td>
</tr>
<tr>
<td>DFHNQNQ</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>CREATE_ENQUEUE_POOL</td>
</tr>
<tr>
<td></td>
<td>REACQUIRE_ENQUEUE</td>
</tr>
<tr>
<td></td>
<td>DEACTIVATE</td>
</tr>
<tr>
<td></td>
<td>SET_NQRNAME_LIST</td>
</tr>
<tr>
<td></td>
<td>DEQUEUE_TASK</td>
</tr>
<tr>
<td>DFHNQED</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>ENQUEUE</td>
</tr>
<tr>
<td></td>
<td>DEQUEUE</td>
</tr>
<tr>
<td>DFHNQEDI</td>
<td>Inline version of DFHNQED.</td>
</tr>
<tr>
<td>DFHNQIB</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>INQUIRE_ENQUEUE</td>
</tr>
<tr>
<td></td>
<td>START_BROWSE_ENQUEUE</td>
</tr>
<tr>
<td></td>
<td>GET_NEXT_ENQUEUE</td>
</tr>
<tr>
<td></td>
<td>END_BROWSE_ENQUEUE</td>
</tr>
<tr>
<td>DFHNQRN</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>INQUIRE_NQRNAME</td>
</tr>
<tr>
<td></td>
<td>ADD_REPLACE_ENQMODEL</td>
</tr>
<tr>
<td></td>
<td>DISCARD_ENQMODEL</td>
</tr>
<tr>
<td></td>
<td>REMOVE_ENQMODEL</td>
</tr>
<tr>
<td></td>
<td>INQUIRE_ENQMODEL</td>
</tr>
<tr>
<td></td>
<td>START_BROWSE_ENQMODEL</td>
</tr>
<tr>
<td></td>
<td>GET_NEXT_ENQMODEL</td>
</tr>
<tr>
<td></td>
<td>END_BROWSE_ENQMODEL</td>
</tr>
<tr>
<td></td>
<td>SET_ENQMODEL</td>
</tr>
<tr>
<td></td>
<td>COMMIT_ENQMODEL</td>
</tr>
<tr>
<td></td>
<td>RESTORE_DIRECTORY</td>
</tr>
<tr>
<td>DFHNQIE</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>INTERPRET_ENQUEUE</td>
</tr>
<tr>
<td>DFHNQST</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>COLLECT_STATISTICS</td>
</tr>
<tr>
<td></td>
<td>COLLECTRESOURCE_STATS</td>
</tr>
<tr>
<td>DFHNQTRI</td>
<td>Provides a trace interpretation routine for CICS dumps and traces.</td>
</tr>
</tbody>
</table>

### Exits

The XNQEREQ and XNQEREQC global user exit points are invoked respectively before and after each EXEC ENQ or DEQ request to the NQ domain.

### Trace

The point IDs for the NQ domain are of the form NQ xxxx; the corresponding trace levels are NQ 1, NQ 2, and Exc.
For more information about the trace points, see the CICS Trace Entries. For more information about using traces in problem determination, see the CICS Problem Determination Guide.
Chapter 33. Enterprise Java domain (EJ)

The Enterprise Java (EJ) domain is logically divided into three parts:

- **Elements**, which covers the manipulation of the EJ Resources of CorbaServers (EJCG), DJars (EJDG) and Beans (EJB)
- **Object Stores**, used to store stateful Session Beans, and to hold the EJB Directory (EJOS and EJOB)
- **Directory**, used to record the association of OTS transactions and object instances with Request Processors (EJDI).

### EJ domain’s specific gates

Table 50 summarizes the EJ domain’s specific gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and whether or not the functions are available through the exit programming interface (XPI).

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>XPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>EJBB</td>
<td>EJ 0Cxx</td>
<td>START_BROWSE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GET_NEXT</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>END_BROWSE</td>
<td>NO</td>
</tr>
<tr>
<td>EJBG</td>
<td>EJ 0Bxx</td>
<td>ADD_BEAN</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CONFIRM_ALL_BEANS</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DELETE_ALL_BEANS</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DELETE_BEAN</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GET_BEAN_DD</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INQUIRE_BEAN</td>
<td>NO</td>
</tr>
<tr>
<td>EJCB</td>
<td>EJ 08xx</td>
<td>START_BROWSE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GET_NEXT</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>END_BROWSE</td>
<td>NO</td>
</tr>
<tr>
<td>EJCG</td>
<td>EJ 07xx</td>
<td>ADD_CORBASERVER</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ACTION_CORBASERVER</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AMEND_CORBASERVER</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DELETE_CORBASERVER</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INQUIRE_CORBASERVER</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RESOLVE_CORBASERVER</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SET_ALL_STATE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WAIT_FOR_CORBASERVER</td>
<td>NO</td>
</tr>
<tr>
<td>EJDB</td>
<td>EJ 0Axx</td>
<td>START_BROWSE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GET_NEXT</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>END_BROWSE</td>
<td>NO</td>
</tr>
<tr>
<td>EJDG</td>
<td>EJ 09xx</td>
<td>ADD_DJAR</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ACTION_DJAR</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AMEND_DJAR</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COUNT_FOR_CS</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DELETE_ALL_DJARS</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DELETE_DJAR</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INQUIRE_DJAR</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RESOLVE_DJAR</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SET_ALL_STATE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WAIT_FOR_DJAR</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WAIT_FOR_USABLE_DJARS</td>
<td>NO</td>
</tr>
<tr>
<td>EJDI</td>
<td>EJ 05xx</td>
<td>ADD_ENTRY</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INITIALIZE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LOOKUP_ENTRY</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>REMOVE_ENTRY</td>
<td>NO</td>
</tr>
<tr>
<td>EJDU</td>
<td>EJ 06xx</td>
<td>DUMP_DATA</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DUMP_STACK</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INQUIRE_TRACE_FLAGS</td>
<td>NO</td>
</tr>
</tbody>
</table>
### EJ domain (EJ)

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>XPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>EJGE</td>
<td>EJ 0Dxx</td>
<td>INITIALIZE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>QUIESCE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TERMINATE</td>
<td>NO</td>
</tr>
<tr>
<td>EJO</td>
<td>EJ 0Fx</td>
<td>RESOLVE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RESOLVE_CSERVERS</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RESOLVE_DJARS</td>
<td>NO</td>
</tr>
<tr>
<td>EJO</td>
<td>EJ 0Exx</td>
<td>ADD_Bean</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>END_Bean_BROWSE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GET_Bean_DD</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GET_NEXT_Bean</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INQUIRE_CORBASERVER</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>START_Bean_BROWSE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WAIT_FOR_CORBASERVER</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WAIT_FOR_USABLE_DJARS</td>
<td>NO</td>
</tr>
<tr>
<td>EJMI</td>
<td>EJ 50xx</td>
<td>ADD_Bean</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADD_METHOD</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DISCARD_METHOD_INFO</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GET_METHOD_INFO</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INITIALIZE</td>
<td>NO</td>
</tr>
<tr>
<td>EJOB</td>
<td>EJ 03xx</td>
<td>END_BROWSE_OBJECT</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GET_NEXT_OBJECTSTORE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INQUIRE_OBJECT</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INQUIRE_STORES</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RETRIEVE_STATISTICS</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>START_BROWSE_OBJECT</td>
<td>NO</td>
</tr>
<tr>
<td>EJOS</td>
<td>EJ 02xx</td>
<td>ACTIVATE_OBJECT</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CLOSE_OBJECT_STORE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OPEN_OBJECT_STORE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>REMOVE_OBJECT</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>REMOVE_STORE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>STORE_OBJECT</td>
<td>NO</td>
</tr>
</tbody>
</table>

#### EJBB gate, START_BROWSE Function

The START_BROWSE function of the EJBB gate initiates the browse upon the chain of Beans. Positioning of the start of the Browse is not supported. Selection by Bean is not provided, but selection by owning CorbaServer and owning DJar is. The end_browse condition is not returned if there are no suitable Beans (this is postponed until the get_next). The returned browsetoken must be used for subsequent GET_NEXT operations. This operation is available from EJJO and so the definitions must be consistent. The browsemode parameter controls which Beans are selected. Only BROWSEMODE(VALIDONLY) should be used by the SPI-layers.

- BROWSEMODE(ALL) selects all Beans (setting not usually used)
- BROWSEMODE(VALIDONLY) selects the Beans whose status has been confirmed (those which are not temporarily present during the install of all the Beans from a DJar). This is the usual (and default) setting.
- BROWSEMODE(INDOUBTONLY) selects the Beans whose status is temporary (those which are temporarily present during the install of all the Beans from a DJar).

#### Input parameters

- corbaserver
  - Name of the CorbaServer to be Browsed
- djar
  - Name of the DJar for this Bean
- [browsemode]
  - Controls which Beans are to be selected for Bean Browse
Output parameters

browsetoken
The pointer set up by START_BROWSE which points to the first DJar in the chain to be Browsed

RESPONSE
is the domain’s response to the call. It can have any of these values OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>EJB_INACTIVE INVALID_BROWSEMODE INVALID_CORBASERVER INVALID_DJAR LOCK_ERROR LOOP PARMS_STORAGE_ERROR SETUP_ERROR STORAGE_ERROR</td>
</tr>
</tbody>
</table>

**EJBB gate, GET_NEXT Function**

The GET_NEXT function of the EJBB gate returns the next Bean Control Block in the list of Beans that meets the selection criteria. The ordering of Beans returned is not specified (the order is not alpha order but LastIn-FirstOut for Browse purposes). This operation is available from EJJO and so the definitions must be consistent. The POINTAT parameter is used to enable a Browse to proceed when the aim of the Browse is to locate a Bean to be deleted.

- POINTAT(NORMAL) should be used in all cases by the SPI layers and general users (and is the default).
- POINTAT(PRIOR) shows the deletion intent. POINTAT(PRIOR) should never be coded in normal circumstances and may result in an infinite loop if used without a delete.

Input parameters

browsetoken
The pointer set up by START_BROWSE which points to the first DJar in the chain to be browsed

[pointat]
Indicates whether to advance the browse pointer to point to the next item in the chain (NORMAL | PRIOR). NORMAL will return the next item in the chain, whereas PRIOR will always return the same item, unless that item has been deleted

Output parameters

bean Name of the Bean
[corbaserver]
Name of the CorbaServer for this DJar
[ddlen] Length of the deployment/meta data area. Used particularly to contain the length of the data if the size is larger than the maximum length for ddareaforin block
[djar] Name of DJar for this Bean
Enterprise Java domain (EJ)

The state of the Bean being Browsed (NORMAL or TEMPORARY).
Indicates that a Bean has been confirmed

RESPONSE is the domain’s response to the call. It can have any of these values
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID.
Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>BROWSE_TOKEN EJB_INACTIVE END_OF_BROWSE INVALID_BROWSE_TOKEN INVALID_POINTAT LOCK_ERROR LOOP PARMS_STORAGE_ERROR SETUP_ERROR</td>
</tr>
</tbody>
</table>

EJBB gate, END_BROWSE Function

The END_BROWSE function of the EJBB gate ends the browse operation and deletes the browsetoken. This operation is available from EJJO and so the definitions must be consistent.

Input parameters

browsetoken
The pointer which points to the first DJar in the chain to be deleted

Output parameters

None There are no output parameters

RESPONSE is the domain’s response to the call. It can have any of these values
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID.
Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>EJB_INACTIVE INVALID_BROWSE_TOKEN LOCK_ERROR LOOP PARMS_STORAGE_ERROR SETUP_ERROR STORAGE_ERROR</td>
</tr>
</tbody>
</table>

EJBG gate, ADD_BEAN function

The ADD_BEAN function of the EJBG gate:

- Creates the Bean element in memory and chains it appropriately. The XRSINDI exit is called to notify the creation of this element.
- The ADDMODE parameter controls the compartmentalization of this operation for usage purposes (this defaults to NORMAL which does both the creation of the Control Block and its replacement in temporary mode). Usage of this verb is always via the Java layers and so ADDMODE(NORMAL) is used.
- The namespace for the Bean is at the CorbaServer level - the Bean cannot already have been installed from a different DJar (with the same name).
- The Bean itself is not actually installed until all the Beans within the Djar have been installed. Therefore, the Bean is added as a Control Block in a temporary
state until all the Beans from the DJar have been so processed. When all the
Beans in the DJar have been (successfully) added, the CONFIRM_ALL_BEANS
call is made to alter the temporary state in the CB into a proper state. Otherwise
the DELETE_ALL_BEANS call is made which deletes the Control Block. As the
DJar was invalid it will be removed.

- On Warm restart, the owning Container and DJar will already have been
restored from the Global Catalog.

**Input parameters**

<table>
<thead>
<tr>
<th><strong>bean</strong></th>
<th>Name of the Bean to be added</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>corbaserver</strong></td>
<td>Name of the CorbaServer for this Bean</td>
</tr>
<tr>
<td><strong>djar</strong></td>
<td>Name of the DJar for this Bean</td>
</tr>
<tr>
<td><strong>ddareaforin</strong></td>
<td>Block for Bean deployment/meta data input</td>
</tr>
</tbody>
</table>

[addmode]

The type of create done for the Bean

[message]

Controls whether a message is issued when a CorbaServer is created

**Output parameters**

None  There are no output parameters

RESPONSE is the domain’s response to the call. It can have any of these values

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID.

Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>BEAN_ALREADY_PRESENT</td>
</tr>
<tr>
<td>DDAREA_FORIN_ABSENT</td>
<td>DJAR_ABSENT</td>
</tr>
<tr>
<td>INVALID_BROWSE_TOKEN</td>
<td>INVALID_CORBASERVER</td>
</tr>
<tr>
<td>INVALID_DD_ZERO_LENGTH</td>
<td>INVALID_DD_ZERO_POINTER</td>
</tr>
<tr>
<td>NAMESPACE_CONFLICT</td>
<td>PARMS_STORAGE_ERROR</td>
</tr>
</tbody>
</table>

**EJBG gate, CONFIRM_ALL_BEANS Function**

The CONFIRM_ALL_BEANS function of the EJBG gate hardens all Beans
associated with the given DJar within the relevant CorbaServer namespace. This
just switches the state of a suitable Bean from temporary to normal. This will run
when all Beans in the DJar have been correctly installed. The key is CS+DJar for
this multiple status changing.

**Input parameters**

| **corbaserver** | The name of the CorbaServer for this Bean |
| **djar** | The name of the DJar for this Bean |
Output parameters

None There are no output parameters

RESPONSE
is the domain’s response to the call. It can have any of these values
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

REASON
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID.
Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>BEAN_ABSENT EJB_INACTIVE LOCK_ERROR LOOP PARMS_STORAGE_ERROR SETUP_ERROR</td>
</tr>
</tbody>
</table>

EJBG gate, DELETE_ALL_BEANS Function

The DELETE_ALL_BEANS function of the EJBG gate is executed when all of the Beans within the Djar did not install or when the owning Djar itself is deleted. All relevant Bean Control Blocks (whatever their state) are deleted. This works via the usual Browse mechanism (BROWSEMODE(ALL)) with POINTAT(PRIOR) enabled to delete each individual Bean. The key of CS+DJar+Bean is required.

Input parameters

corbaserver
The name of the CorbaServer for this Bean
djar
the name of the Djar for this Bean

Output parameters

None There are no output parameters

RESPONSE
is the domain’s response to the call. It can have any of these values
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

REASON
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID.
Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>BEAN_ABSENT BROWSE_ERROR EJB_INACTIVE LOCK_ERROR LOOP PARMS_STORAGE_ERROR SETUP_ERROR</td>
</tr>
</tbody>
</table>

EJBG gate, DELETE_BEAN Function

The DELETE_BEAN function of the EJBG gate deletes the Bean Control Block. The XRSINDI exit is also called to notify the removal. The full key of CS+DJar+Bean is required.

Input parameters

bean The name of the Bean
corbaserver
The name of the CorbaServer for this Bean
djar  The name of the DJar for the Bean

Output parameters
None  There are no output parameters

RESPONSE
is the domain’s response to the call. It can have any of these values
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID.
Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>BEAN_ABSENT EJB_INACTIVE LOCK_ERROR LOOP PARMS_STORAGE_ERROR SETUP_ERROR STORAGE_ERROR</td>
</tr>
</tbody>
</table>

EJBG gate, GET_BEAN_DD Function
The GET_BEAN_DD function of the EJBG gate returns the saved Deployment/Meta Data for the Bean (key is CS+Bean) in a buffer. The Buffer Pointer (_P) must be non-zero and point to a suitable area of length _M bytes. After the Deployment/Meta Data has been moved to the _P area, the _N item will be set to the length of the Deployment/Meta Data as is the DDLEN parameter. If _M is too small for the Deployment/Meta Data, then it is not moved into the _P area, _N is set to zero and the required length returned in DDLEN. Note that DDLEN and _N are only valid for OK operation or length errors - they are not available for use in other circumstances.

This operation is available via EJJO and so parameters should be kept consistent.

Input parameters
bean  The name of the Bean
corbaserver
The name of the CorbaServer for this Bean
ddareaforupd
A buffer for Bean the deployment/meta data update area

Output parameters
[ddlen]
The length of the deployment/meta data area. Used particularly to contain the length of the data if the size is larger than the maximum length for the ddareaforin block

[djar]  The name of the DJar for this Bean

RESPONSE
is the domain’s response to the call. It can have any of these values
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID.
Possible values are:
RESPONSE Possible REASON values

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>BEAN_ABSENT CORBASERVER_ABSENT CORBASERVER_INVALID_STATE DJAR_ABSENT DJAR_INVALID_STATE DDAREAFORUPD_ABSENT DD_AREA_TOO_SMALL EJB_INACTIVE INVALID_DDAREAFORUPD INVALID_DD_ZERO_LENGTH INVALID_DD_ZERO_POINTER LOCK_ERROR LOOP_PARMS_STORAGE_ERROR SETUP_ERROR</td>
</tr>
</tbody>
</table>

**EJBG gate, INQUIRE_BEAN Function**

The INQUIRE_BEAN function of the EJBG gate extracts information from the named Bean Control Block (key is CS+Bean). Note that the length of the Deployment/Meta Data is returned, but this XML is obtained via get_bean_dd.

This function can be used to determine the DJar which sourced the Bean.

**Input parameters**

- **bean** The name for the Bean
- **corbaserver** The name of the CorbaServer for this Bean

**Output parameters**

- **[ddlen]** The length of the deployment/meta data area. Used particularly to contain the length of the data if the size is larger than the maximum length of the ddareaforinp block
- **[djar]** The name of the DJar for this Bean
- **[status]** The state of the Bean being inquired upon

**RESPONSE** is the domain’s response to the call. It can have any of these values

- OK
- EXCEPTION
- DISASTER
- INVALID
- KERNERROR
- PURGED

**[REASON]** is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>BEAN_ABSENT EJB_INACTIVE LOCK_ERROR LOOP_PARMS_STORAGE_ERROR SETUP_ERROR</td>
</tr>
</tbody>
</table>

**EJCB gate, START_BROWSE function**

The START_BROWSE function of the EJCB gate initiates the browse upon the chain of CorbaServers. Positioning of the start of the Browse is not supported. Selection by CorbaServer is not provided. The end_browse condition is not returned if there are no suitable CorbaServers (this is postponed until the get_next). The returned browsetoken must be used for subsequent GET_NEXT operations.

**Input parameters**

- **None** No input parameters.
Output parameters

browsetoken
   The pointer set up by START_BROWSE which points to the first CorbaServer in the chain to be browsed

RESPONSE
   is the domain’s response to the call. It can have any of these values
   OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
   is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>EJB_INACTIVE LOCK_ERROR LOOP_SETUP_ERROR STORAGE_ERROR</td>
</tr>
</tbody>
</table>

EJCB gate, GET_NEXT function

The GET_NEXT function of the EJCB gate returns the next CorbaServer Control Block in the list of CorbaServers. The ordering of CorbaServers returned is not specified (the order is not alpha order but Last-FirstOut for Browse purposes). The POINTAT parameter is used to enable a Browse to proceed when the aim of the Browse is to locate a CorbaServer to be deleted.

- POINTAT(NORMAL) should be used in all cases by the SPI layers and general users (and is the default).
- POINTAT(PRIOR) shows the deletion intent. POINTAT(PRIOR) should never be coded in normal circumstances and may result in an infinite loop if used without a delete.

Input parameters

browsetoken
   The pointer set up by START_BROWSE which points to the first CorbaServer in the chain to be browsed

[pointat]
   Indicates whether to advance the browse pointer to point to the next item in the chain (NORMAL | PRIOR). NORMAL will return the next item in the chain, whereas PRIOR will always return the same item, unless that item has been deleted

Output parameters

corbaserver
   Name of the CorbaServer

[state]
   Indicates the current Resolution State and whether it is available for use or not.

[timeout]
   The elapsed time period (in seconds) of inactivity after which a session Bean can be discarded

[port]
   The binary TCP/IP port number for non-SSL communication included in IORs exported from this CorbaServer
### Enterprise Java domain (EJ)

| SSLport | The binary TCP/IP port number for SSL communication included in IORs exported from this CorbaServer |
| SSL     | the SSL type for this CorbaServer (YES|NO|CLIENTAUTH) |
| Jndiprefix | The prefix to use at runtime when publishing the JNDI |
| Shelf   | The fully qualified name of a directory (a ‘shelf’ for ‘jars’) on HFS |
| Host    | The TCP/IP hostname or the dotted decimal TCP/IP address included in IORs exported from this CorbaServer |
| Cert    | The label of a certificate within the keyring that is to be used as a client certificate in the SSL handshake for outbound IIOP connections |

**RESPONSE**

is the domain’s response to the call. It can have any of these values

| OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED |

**[REASON]**

is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

| RESPONSE | Possible REASON values |
| DISASTER | ABEND |
| EXCEPTION | BROWSE_TOKEN EJB_INACTIVE END_OF_BROWSE INVALID_BROWSE_TOKEN INVALID_POINTAT LOCK_ERROR LOOP SETUP_ERROR |

### EJCB gate, END_BROWSE function

The END_BROWSE function of the EJCB gate ends the browse operation and deletes the browsetoken.

**Input parameters**

browsetoken

The pointer set up by START_BROWSE which points to the first CorbaServer in the chain to be browsed

**Output parameters**

None There are no output parameters

**RESPONSE**

is the domain’s response to the call. It can have any of these values

| OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED |

**[REASON]**

is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

| RESPONSE | Possible REASON values |
| DISASTER | ABEND |
| EXCEPTION | EJB_INACTIVE INVALID_BROWSE_TOKEN LOCK_ERROR LOOP SETUP_ERROR STORAGE_ERROR |
EJCG gate, ACTION_CORBASERVER function

The ACTION_CORBASERVER function of the EJCG gate is a gate which tells another party that something is to be done on the CorbaServer. The implemented actions are to manipulate the External Namespace for the named CorbaServer.

Input parameters

<table>
<thead>
<tr>
<th>corbaserver</th>
<th>Name of the CorbaServer on which the action is to be done</th>
</tr>
</thead>
<tbody>
<tr>
<td>actionmode</td>
<td>the action to perform on the CorbaServer (PUBLISH</td>
</tr>
</tbody>
</table>

Output parameters

RESPONSE

is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]

is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>CORBASESERVER_ABSENT CORBASESERVER_INVALID_STATE DJAR_INVALID_STATE EJB_INACTIVE INVALID_ACTION LOCK_ERROR LOOP PARMS_STORAGE_ERROR PUBLISH_ERROR RETRACT_ERROR SETUP_ERROR STORAGE_ERROR</td>
</tr>
</tbody>
</table>

EJCG gate, ADD_CORBASERVER function

The ADD_CORBASERVER function creates a CorbaServer Control Block:

- creates the CorbaServer in memory, chains it appropriately, and saves an entry in the Global Catalog for Warm restart purposes.
- The XRSINDI exit is called to notify the creation of this element.
- The ADDMODE parameter controls the scope of this operation for restart purposes (this defaults to NORMAL which does both creation of the Control Block and its cataloging). Usage of this verb via the SPI/RDO layers should always code ADDMODE(NORMAL).

Input parameters

<table>
<thead>
<tr>
<th>corbaserver</th>
<th>Name of the CorbaServer to be added</th>
</tr>
</thead>
<tbody>
<tr>
<td>state</td>
<td>Indicates the current Resolution State and whether it is available for use or not.</td>
</tr>
<tr>
<td>timeout</td>
<td>The elapsed time (in seconds) of inactivity after which a session Bean can be discarded</td>
</tr>
<tr>
<td>port</td>
<td>The binary TCP/IP port number for non-SSL communication included in IORs exported from this CorbaServer</td>
</tr>
<tr>
<td>sslport</td>
<td>The binary TCP/IP port number for SSL communication included in IORs exported from this CorbaServer</td>
</tr>
</tbody>
</table>
Enterprise Java domain (EJ)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ssl</td>
<td>The SSL type for this CorbaServer (YES</td>
</tr>
<tr>
<td>jndiprefix</td>
<td>The prefix to use at runtime when publishing to JNDI</td>
</tr>
<tr>
<td>shelf</td>
<td>The fully qualified name of a directory (a ‘shelf’ for ‘jars’) on HFS</td>
</tr>
<tr>
<td>host</td>
<td>The TCP/IP hostname or the dotted decimal TCP/IP address included in IORs exported from this CorbaServer</td>
</tr>
<tr>
<td>cert</td>
<td>The label of a certificate within the keyring that is to be used as a client certificate in the SSL handshake for outbound IIOP connections</td>
</tr>
<tr>
<td>[addmode]</td>
<td>The type of create done for the CorbaServer</td>
</tr>
<tr>
<td>[message]</td>
<td>Controls whether a message is issued when a CorbaServer is created</td>
</tr>
</tbody>
</table>

**Output parameters**

None

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**[REASON]**

is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>ATTACH_ERROR CATALOG_ERROR CERT_ERROR CORBASERVER_ALREADY_THERE EJB_INACTIVE INVALID_CERT INVALID_CORBASERVER INVALID_HOST INVALID_JNDIPREFIX INVALID_PORT INVALID_SHELF INVALID_SSL INVALID_SSLPORT INVALID_STATE INVALID_TIMEOUT LOCK_ERROR LOOP_PARMS_STORAGE_ERROR SETUP_ERROR STORAGE_ERROR</td>
</tr>
</tbody>
</table>

**EJCG gate, AMEND_CORBASERVER function**

The AMEND_CORBASERVER function of the EJCG gate changes information held within the CorbaServer Control Block. It does not harden this information over a CICS restart, nor does the change get communicated to the executing JVMs.

**Input parameters**

**corbaserver**

Name of the CorbaServer in which the information is to be changed

**[timeout]**

The elapsed time (in seconds) of inactivity after which a session Bean can be discarded

**[state]**

Specifies the state into which the CorbaServer is to be put (values are PENDINIT|INITING|UNUSABLE|PENDRESOLV|RESOLVING|INSERV|UNRESOLVED|DELETING)

**[current_state]**

Used as a check, must match the existing state of the CorbaServer (values are PENDINIT|INITING|UNUSABLE|PENDRESOLV|RESOLVING|INSERV|UNRESOLVED|DELETING)
Output parameters

RESPONSE

is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]

is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>CORBASERVER_ABSENT CORBASERVER_INVALID_STATE CORBASERVER_STATE_CHANGED EJB_INACTIVE EJOS_ERROR INVALID_STATE INVALID_STATE_CHANGE INVALID_TIMEOUT LOCK_ERROR LOOPtparams_STORAGE_ERROR SETUP_ERROR</td>
</tr>
</tbody>
</table>

EJCG gate, DELETE_CORBASERVER function

The DELETE_CORBASERVER function of the EJCG gate removes a CorbaServer.

- Deletes the CorbaServer Control Block and removes the saved entry in the Global catalog. The XRSINDI exit is called to notify the removal.
- The Java layers are informed that the CorbaServer has been deleted.
- This operation has a side effect in that all DJars associated with the CorbaServer are also deleted, and then all the Beans from the DJar.

Input parameters

corbaserver

Name of the CorbaServer to be deleted

Output parameters

RESPONSE

is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]

is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>CATALOG_ERROR CORBASERVER_ABSENT CORBASERVER_DELETING DELDJAR_ERROR EJB_INACTIVE LOCK_ERROR LOOPtparams_STORAGE_ERROR SETUP_ERROR STORAGE_ERROR</td>
</tr>
</tbody>
</table>

EJCG gate, INQUIRE_CORBASERVER function

The INQUIRE_CORBASERVER function of the EJCG gate extracts information from the named CorbaServer Control Block. It is also executed indirectly from the EJJO gate.

Input parameters

corbaserver

Name of the CorbaServer from which to extract information
Enterprise Java domain (EJ)

**Output parameters**

- **[state]** The state of the CorbaServer. Used to indicate whether the CorbaServer is available to use or not

- **[timeout]** The elapsed time (in seconds) of inactivity after which a session Bean can be discarded

- **[port]** The binary TCP/IP port number for non-SSL communication included in IORs exported from this CorbaServer

- **[sslport]** The binary TCP/IP port number for SSL communication included in IORs exported from this CorbaServer

- **[ssl]** The SSL type for this CorbaServer (YES|NO|CLIENTAUTH)

- **[jndiprefix]** The prefix to use at runtime when publishing to JNDI

- **[shelf]** The fully qualified name of a directory (a ‘shelf’ for ‘jars’) on HFS

- **[host]** The TCP/IP hostname or the dotted decimal TCP/IP address included in IORs exported from this CorbaServer

- **[cert]** The label of a certificate within the keyring that is to be used as a client certificate in the SSL handshake for outbound IIOP connections

**RESPONSE** is the domain’s response to the call. It can have any of these values:

- OK
- EXCEPTION
- DISASTER
- INVALID
- KERNERROR
- PURGED

**[REASON]** is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>CORBASESERVER_ABSENT EJB_INACTIVE LOCK_ERROR LOOP PARMS_STORAGE_ERROR SETUP_ERROR</td>
</tr>
</tbody>
</table>

**EJCG gate, RESOLVE_CORBASESERVER function**

The RESOLVE_CORBASESERVER function of the EJCG gate makes the CorbaServer available for use by Resolution (called by the CEJR transaction). The Java layers are informed that the CorbaServer has been created.

**Input parameters**

- **corbaserver** Name of the CorbaServer

**Output parameters**

- **did_stage** The output from Resolve function which indicates which stage of resolution was done (STAGE1 or STAGE2)

**RESPONSE** is the domain’s response to the call. It can have any of these values:

- OK
- EXCEPTION
- DISASTER
- INVALID
- KERNERROR
- PURGED
[REASON] is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>BAD_STATE_SET CATALOG_ERROR CORBASERVER_ABSENT CORBASERVER_INVALID_STATE EJB_INACTIVE EJOS_ERROR IILS_ERROR INVALID_CORBASERVER LOCK_ERROR LOOP PARMS_STORAGE_ERROR SETUP_ERROR</td>
</tr>
</tbody>
</table>

**EJCG gate, SET_ALL_STATE function**

The SET_ALL_STATE function sets the state of all the CorbaServers.

**input parameters**

- **state** Indicates the current Resolution State and whether it is available for use or not.

**Output parameters**

- **RESPONSE** is the domain’s response to the call. It can have any of these values:
  - OK
  - EXCEPTION
  - DISASTER
  - INVALID
  - KERNERROR
  - PURGED

[REASON] is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>EJB_INACTIVE INVALID_STATE LOCK_ERROR LOOP PARMS_STORAGE_ERROR SETUP_ERROR</td>
</tr>
</tbody>
</table>

**EJCG gate, WAIT_FOR_CORBASERVER function**

The WAIT_FOR_CORBASERVER function of the EJCG gate will wait until the CorbaServer enters the required state.

**Input parameters**

- **corbaserver** Name of the CorbaServer being waited on
- **state** Indicates the current Resolution State and whether it is available for use or not.

**Output parameters**

- **RESPONSE** is the domain’s response to the call. It can have any of these values:
  - OK
  - EXCEPTION
  - DISASTER
  - INVALID
  - KERNERROR
  - PURGED

[REASON] is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
</tbody>
</table>
Enterprise Java domain (EJ)

RESPONSE  Possible REASON values
EXCEPTION  CORBASERVER_ABSENT CORBASERVER_UNRESOLVED CORBASERVER_UNUSABLE EJB_INACTIVE
           INVALID_STATE LOCK_ERROR LOOP PARMS_STORAGE_ERROR SETUP_ERROR WAIT_ERROR

EJCP gate, DISCARD_DJAR function

The DISCARD_DJAR function of the EJCP gate is called by the EJ domain during DJar Discard processing, but the DJar is unavailable for inquire.

The jar file is removed from its shelf and the shelf directory is deleted.

Input parameters

corbaserver  Name of the CorbaServer (container) in which this DJar is installed. This is the CorbaServer name as specified in the DJar definition.
djar          Name of the DJar to be discarded

Output parameters

RESPONSE  is the domain’s response to the call. It can have any of these values
          OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED
[REASON]  is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>SHELF_ACCESS_ERROR</td>
</tr>
</tbody>
</table>

EJCP gate, INSTALL_DJAR function

The INSTALL_DJAR function of the EJCP gate is called by the EJ domain when a DJar is installed as the second part of the DJar installation processing (after the DJar has been copied to the shelf).

The Java Container should initiate the processing of the DJar and pass the obtained Bean to the EJ domain.

The EJ domain expects to be called back for inquire_corbaserver and inquire_djar during the processing of this call, so the DJar must be available for inquire.

Input parameters

corbaserver  Name of the CorbaServer (container) into which this DJar is to be installed. This is the CorbaServer name as specified in the DJar definition.
djar          Name of the DJar to be installed
hfsfile       The fully qualified name of the jar file to be installed. The name must be a valid HFS filename and must not have any trailing blanks.

Output parameters

RESPONSE  is the domain’s response to the call. It can have any of these values
Enterprise Java domain (EJ)

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>CONTAINER_ERROR HFSFILE_NOT_FOUND HFS_ACCESS_ERROR SHELF_ACCESS_ERROR</td>
</tr>
</tbody>
</table>

EJCP gate, PRE_INSTALL_DJAR function

The PRE_INSTALL_DJAR function of the EJCP gate is called by the EJ domain when a DJar is installed to copy the hfsfile comprising the DJar onto the shelf.

The Java Container should create a copy of the DJar file on the shelf for the associated CorbaServer.

The EJ domain expects to be called back for inquire_corbaserver and inquire_djar during the processing of this call (EJJO gate), so the DJar must be available for inquire.

Input parameters

corbaserver
  Name of the CorbaServer (container) into which this DJar is to be installed.
  This is the CorbaServer name as specified in the DJar definition.

djarn
  Name of the DJar to be installed

hfsfile
  The fully qualified name of the jar file to be installed. The name must be a valid HFS filename and must not have any trailing blanks.

Output parameters

RESPONSE
  is the domain’s response to the call. It can have any of these values

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>CONTAINER_ERROR HFSFILE_NOT_FOUND HFS_ACCESS_ERROR SHELF_ACCESS_ERROR</td>
</tr>
</tbody>
</table>

EJCP gate, PUBLISH_CORBASERVER function

The PUBLISH_CORBASERVER function of the EJCP gate may be invoked by the EJ domain or the exec interface layer. All beans in the specified logical server are published to JNDI. This is not to be confused with the IILS publish function.

The Java container calls back to the EJ domain with a browse of all installed Beans for the logical server during processing of the function.
Enterprise Java domain (EJ)

**Input parameters**

`corbaserver`

Name of the CorbaServer for which to publish Beans.

**Output parameters**

**RESPONSE**

is the domain’s response to the call. It can have any of these values: `OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

[REASON]

is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>CONTAINER_ERROR HFS_ACCESS_ERROR JNDI_ACCESS_ERROR</td>
</tr>
</tbody>
</table>

**EJCP gate, PUBLISH_DJAR function**

The PUBLISH_DJAR function of the EJCP gate may be invoked by the EJ domain or the exec interface layer. All beans in the specified logical server are published to JNDI.

The Java container calls back to the EJ domain with a browse of all installed Beans for the DJar during processing of the function.

**Input parameters**

`corbaserver`

Name of the CorbaServer for which to publish Beans.

`djar`

Name of the DJar for which to publish all Beans.

**Output parameters**

**RESPONSE**

is the domain’s response to the call. It can have any of these values: `OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED`

[REASON]

is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>CONTAINER_ERROR HFS_ACCESS_ERROR JNDI_ACCESS_ERROR</td>
</tr>
</tbody>
</table>

**EJCP gate, RETRACT_CORBASERVER function**

The RETRACT_CORBASERVER function of the EJCP gate may be invoked by the EJ domain or the exec interface layer. All beans in the specified logical server are retracted from JNDI. This is not to be confused with the IILS retract function.

The Java container calls back to the EJ domain with a browse of all installed Beans for the logical server during processing of the function.
Input parameters

**corbaserver**
Name of the CorbaServer for which to retract Beans.

**Output parameters**

**RESPONSE**
is the domain’s response to the call. It can have any of these values

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**[REASON]**
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>HFS_ACCESS_ERROR JNDI_ACCESS_ERROR</td>
</tr>
</tbody>
</table>

**EJCP gate, RETRACT_DJAR function**
The RETRACT_DJAR function of the EJCP gate may be invoked by the EJ domain or the exec interface layer. All beans in the specified DJar server are retracted from JNDI.

The Java container calls back to the EJ domain with a browse of all installed Beans for DJar or the logical server during processing of the function.

**Input parameters**

**corbaserver**
Name of the CorbaServer for which to retract Beans.

**djar**
Name of the DJar for which to retract all Beans.

**Output parameters**

**RESPONSE**
is the domain’s response to the call. It can have any of these values

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**[REASON]**
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>HFS_ACCESS_ERROR JNDI_ACCESS_ERROR</td>
</tr>
</tbody>
</table>

**EJDB gate, START_BROWSE function**
The START_BROWSE function of the EJDB gate initiates the browse upon the chain of DJars. Positioning of the start of the Browse is not supported. Selection by DJars is not provided, but selection by owning CorbaServer is. The end_browse condition is not returned if there are no suitable DJars (this is postponed until the get_next). The returned browsetoken must be used for subsequent GET_NEXT operations.
Enterprise Java domain (EJ)

Input parameters

**corbaserver**

The name of the CorbaServer for this DJar

Output parameters

**browsetoken**

The pointer set up by START_BROWSE which points to the first CorbaServer in the chain to be browsed

**RESPONSE**

is the domain’s response to the call. It can have any of these values

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**[REASON]**

is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID.

Possible responses are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>EJB_INACTIVE INVALID_CORBASEVER LOCK_ERROR LOOP_SETUP_ERROR STORAGE_ERROR</td>
</tr>
</tbody>
</table>

EJDB gate, GET_NEXT function

The GET_NEXT function of the EJDB gate returns the next DJar Control Block in the list of DJars that meets the selection criteria. The ordering of DJars returned is not specified (the order is not alpha order but LastIn-FirstOut for Browse purposes). The POINTAT parameter is used to enable a Browse to proceed when the aim of the browse is to locate a DJar to be deleted.

- POINTAT(NORMAL) should be used in all cases by the SPI layers and general users (and is the default).
- POINTAT(PRIOR) shows the deletion intent. POINTAT(PRIOR) should never be coded in normal circumstances and may result in an infinite loop if used without a delete.

Input parameters

**browsetoken**

The pointer set up by START_BROWSE which points to the first DJar in the chain to be browsed

**[pointat]**

Indicates whether to advance the browse pointer to point to the next item in the chain (NORMAL | PRIOR). NORMAL will return the next item in the chain, whereas PRIOR will always return the same item, unless that item has been deleted

Output parameters

**djar**

The name of the DJar

**[corbaserver]**

The name of the CorbaServer for this DJar

**[hfsfile]**

The fully qualified name of the deployed jar file on HFS

**[state]**

The state of the DJar. Used to indicate whether the DJar is available for use or not
RESPONSE is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>BROWSE_TOKEN EJB_INACTIVE END_OF_BROWSE INVALID_BROWSE_TOKEN INVALID_POINTAT LOCK_ERROR LOOP SETUP_ERROR</td>
</tr>
</tbody>
</table>

**EJDB gate, END_BROWSE function**

The END_BROWSE function of the EJDB gate ends the browse operation and deletes the browsetoken.

**Input parameters**

**browsetoken**

The pointer set up by START_BROWSE which points to the first DJar in the chain to be browsed.

**Output parameters**

None There are no output parameters

RESPONSE is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>EJB_INACTIVE INVALID_BROWSE_TOKEN LOCK_ERROR LOOP SETUP_ERROR</td>
</tr>
</tbody>
</table>

**EJDG gate, ACTION_DJAR function**

The ACTION_DJAR function of the EJDG gate tells another party that something is to be done on the DJar. The implemented actions are to manipulate the External Namespace for the named DJar.

**Input parameters**

**djar** The name of the DJar

**actionmode**

The action to perform on the DJar (PUBLISH|RETRACT)

**Output parameters**

None There are no output parameters

RESPONSE is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

Enterprise Java domain (EJ)

[REASON]

is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID.

Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>DJAR_ABSENT DJAR_INVALID_STATE EJB_INACTIVE INVALID_ACTION LOCK_ERROR LOOP PARMS_STORAGE_ERROR PUBLISH_ERROR RETRACT_ERROR SETUP_ERROR STORAGE_ERROR</td>
</tr>
</tbody>
</table>

EJDG gate, ADD_DJAR function

The ADD_DJAR function of the EJDG gate creates a DJar Control Block.

- Creates the DJar element in memory, chains it appropriately, and saves an entry in the Global Catalog for Warm restart purposes.
- The XRSINDI exit is called to notify the creation of the element. The Java layers are informed that the DJar has been created.
- The ADDMODE parameter controls the compartmentalization of this operation for restart purposes (this defaults to NORMAL which does both the creation of the Control Block and its cataloging). Usage of this verb via the SPI/RDO layers should always code ADDMODE(NORMAL).

Input parameters

djar  The name of the DJar to be added
corbaserver  The name of the CorbaServer for this DJar
hfsfile  The fully qualified name of the deployed jar file on HFS
state  Indicates the current Resolution State and whether it is available for use or not.

Output parameters

None  There are no output parameters

RESPONSE  is the domain’s response to the call. It can have any of these values
OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON]

is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID.

Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>ATTACH_ERROR CATALOG_ERROR CORBASERVER_ABSENT CORBASERVER_INVALID_STATE DJAR_ALREADY_THERE EJB_INACTIVE INVALID_ACTION CORBASERVER INVALID_DJAR INVALID_HFSNAME INVALID_STATE LOCK_ERROR LOOP PARMS_STORAGE_ERROR SETUP_ERROR STORAGE_ERROR</td>
</tr>
</tbody>
</table>

EJDG gate, AMEND_DJAR function

The AMEND_DJAR function of the EJDG gate alters the DJar Control Block, but does not catalog the change or tell Java about the amendment.

Input parameters

djar  The name of the DJar to be changed
[**state**] Specifies the state into which the DJar is to be put (values are PENDINIT|INITING|UNUSABLE|PENDRESOLV|RESOLVING|INSERV|UNRESOLVED|DELETING)

[**current_state**] Used as a check, must match the existing state of the DJar (values are PENDINIT|INITING|UNUSABLE|PENDRESOLV|RESOLVING|INSERV|UNRESOLVED|DELETING)

**Output parameters**

**None** There are no output parameters

**RESPONSE** is the domain’s response to the call. It can have any of these values OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**[REASON]** is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>BAD_STATE_CHANGE DJAR_ABSENT DJAR_STATE_CHANGED EJB_INACTIVE INVALID_STATE LOCK_ERROR LOOP FARGS_STORAGE_ERROR SETUP_ERROR</td>
</tr>
</tbody>
</table>

**EJDG gate, COUNT_FOR_CS function**

The COUNT_FOR_CS function of the EJDG gate totals the number of DJars in each state for the owning CorbaServer

**Input parameters**

corbaserver

The name of the CorbaServer

**Output parameters**

[**ndjars**]
The number of DJars in this Corbaserver

[**n pendinit**]
The number of DJars which are in pendinit state in the CorbaServer

[**nininiting**]
The number of DJars which are in initing state in the CorbaServer

[**nunusable**]
The number of DJars which are in unusable state in the CorbaServer

[**n pendresolv**]
The number of DJars which are in pendresolve state in the CorbaServer

[**n resolving**]
The number of DJars which are in resolving state in the CorbaServer

[**n inserv**]
The number of DJars which are in inservice state in the CorbaServer

[**n unresolved**]
The number of DJars which are in unresolved state in the CorbaServer
Enterprise Java domain (EJ)

[ndeleting]
The number of DJars which are in deleting state in the CorbaServer

**RESPONSE**
is the domain’s response to the call. It can have any of these values

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID.

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>BAD_STATE_CHANGE DJAR_ABSENT DJAR_STATE_CHANGED EJB_INACTIVE INVALID_STATE LOCK_ERROR LOOP PARMS_STORAGE_ERROR SETUP_ERROR</td>
</tr>
</tbody>
</table>

**EJDG gate, DELETE_DJAR function**
The DELETE_DJAR function of the EJDG gate

- Deletes the DJar Control Block and removes the saved entry in the Global Catalog. the XRSINDI exit is also called to notify the removal.
- The Java layers are informed that the DJar has been deleted. However, this notification is not done if the deletion has been initiated by the deletion of the owning CorbaServer (this is notified by the delmode parameter - DELMODE(CASCADE) showing this CorbaServer initiated deletion and DELMODE(NORMAL) showing that the deletion has been initiated from the SPI/CEMT layers).
- This operation has a side effect in that all Beans associated with the DJar are also deleted.

**Input parameters**

djar  The name of the DJar to be deleted

delmode
Indicates what type of deletion is being done:

- DELMODE(CASCADE) indicates an owning CorbaServer initiated the deletion of this DJar
- DELMODE(NORMAL) indicated deletion is for SPI/CEMT delete DJar request

**Output parameters**

None  There are no output parameters

**RESPONSE**
is the domain’s response to the call. It can have any of these values

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID.

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>CATALOG_ERROR DJAR_ABSENT DJAR_DELETING EJB_INACTIVE LOCK_ERROR LOOP PARMS_STORAGE_ERROR SETUP_ERROR STORAGE_ERROR</td>
</tr>
</tbody>
</table>
**EJDG gate, DELETE_ALL_DJARS function**

The DELETE_ALL_DJARS function of the EJDG gate is called when the owning CorbaServer is deleted which forces the cascaded deletion of all the DJars associated with the CorbaServer. This gate eventually uses EJDG.DELETE_DJAR with DELMODE(CASCADE) as part of its operation.

**Input parameters**

corbaserver

The name of the CorbaServer for these DJars

**Output parameters**

None There are no output parameters

**RESPONSE**

is the domain’s response to the call. It can have any of these values

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**[REASON]**

is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID.

Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>CORBASESERVER_ABSENT EJB_INACTIVE LOCK_ERROR LOOP PARMS_STORAGE_ERROR SETUP_ERROR STORAGE_ERROR</td>
</tr>
</tbody>
</table>

**EJDG gate, INQUIRE_DJAR function**

The INQUIRE_DJAR function of the EJDG gate extracts information from the named DJar Control Block

**Input parameters**

djar

The name of the DJar

**Output parameters**

[corbaserver]

The name of the CorbaServer for this DJar

[hfsfile]

The fully qualified name of the deployed jar file on HFS

[state]

The state of the DJar. Used to indicate whether the DJar is available for use or not

**RESPONSE**

is the domain’s response to the call. It can have any of these values

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**[REASON]**

is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID.

Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>DJAR_ABSENT EJB_INACTIVE LOCK_ERROR LOOP PARMS_STORAGE_ERROR SETUP_ERROR</td>
</tr>
</tbody>
</table>
EJDG gate, SET_ALL_STATE function

The SET_ALL_STATE function of the EJDG gate sets the state of all the DJars.

Input parameters

state Indicates the current Resolution State and whether it is available for use or not.

Output parameters

None There are no output parameters

RESPONSE is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>EJB_INACTIVE INVALID_STATE LOCK_ERROR LOOP PARMS_STORAGE_ERROR SETUP_ERROR</td>
</tr>
</tbody>
</table>

EJDG gate, WAIT_FOR_DJAR function

The WAIT_FOR_DJAR function of the EJDG gate waits until the DJars enter the required state.

Input parameters

djar The name of the DJar being waited on
state Indicates the current Resolution State and whether it is available for use or not.

Output parameters

None There are no output parameters

RESPONSE is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>DJAR_ABSENT DJAR_UNRESOLVED DJAR_UNUSABLE EJB_INACTIVE INVALID_STATE LOCK_ERROR LOOP PARMS_STORAGE_ERROR SETUP_ERROR WAIT_ERROR</td>
</tr>
</tbody>
</table>

EJDG gate, WAIT_FOR_USABLE_DJARS function

The WAIT_FOR_USABLE_DJARS function of the EJDG gate waits until all the DJars associated with a CorbaServer are INSERV.
### Input parameters
**corbaserver**
The name of the CorbaServers for these DJars

### Output parameters
None There are no output parameters

### RESPONSE
is the domain’s response to the call. It can have any of these values:
- OK
- EXCEPTION
- DISASTER
- INVALID
- KERNERROR
- PURGED

### [REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID.

### Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>CORBASERVER_ABSENT CORBASERVER_ERROR CORBASERVER_INVALID_STATE COUNT_ERROR DJAR_ABSENT DJAR_UNRESOLVED DJAR_UNUSABLE EJB_INACTIVE LOCK_ERROR LOOP PARMS_STORAGE_ERROR SETUP_ERROR WAIT_ERROR</td>
</tr>
</tbody>
</table>

## EJDI gate, ADD_ENTRY Function

The ADD_ENTRY function of the EJDI gate adds a new entry to the Directory partition for the specified LogicalServer.

No entry with the same name should exist in the specified LogicalServer partition. In the case of a transaction entry, no existing entry should refer to the same request stream, but this is not checked.

### Input parameters
**entry_key**
The key (OTS or Object Key) for the entry

**entry_type**
Indicates whether this is a transaction or object_key entry

**logicalserver**
Name of the LogicalServer for which the entry is to be added

**request_stream_id**
Public ID of the request stream to be put in the entry

### Output parameters
None There are no output parameters.

### RESPONSE
is the domain’s response to the call. It can have any of these values:
- OK
- EXCEPTION
- DISASTER
- INVALID
- KERNERROR
- PURGED

### [REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID.

### Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>DUPLICATE_ENTRY FILE_CONNECT_ERROR FILE_CORRUPT_ERROR FILE_FULL_ERROR FILE_IO_ERROR STORE_NOT_OPEN</td>
</tr>
</tbody>
</table>
EJDI gate, INITIALIZE Function

The INITIALIZE function of the EJDI gate is called when a store_not_open has been detected.

Input parameters
logicalServer
Name of the LogicalServer for which the entry is to be added

Output parameters
None
There are no output parameters.

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID.
Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>FILE_CONNECT_ERROR FILE_CORRUPT_ERROR FILE_IO_ERROR FILE_NOT_FOUND FILE_RECOVERY_ERROR FILE_RECOVERY_UNKNOWN</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_KEYLENGTH INVALID_RECORD_SIZE</td>
</tr>
</tbody>
</table>

EJDI gate, LOOKUP_ENTRY Function

The LOOKUP_ENTRY function of the EJDI gate looks up the given OTS transaction or object_key / LogicalServer pair and returns the associated Request Stream if found.

Input parameters
entry_key
The key (OTS transaction or Object Key) for the entry to be returned

entry_type
Indicates whether this is a transaction or object_key entry

logicalServer
Name of the LogicalServer to search for the entry

request_stream_buffer
Caller supplied buffer to contain the request stream id

Output parameters
None
There are no output parameters.

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID.
Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>BUFFER_TOO_SMALL ENTRY_NOT_FOUND FILE_CONNECT_ERROR FILE_CORRUPT_ERROR FILE_IO_ERROR OBJECT_CORRUPT STORE_NOT_FOUND STORE_NOT_OPEN</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_KEYLENGTH</td>
</tr>
</tbody>
</table>

**EJDI gate, REMOVE_ENTRY Function**

The REMOVE_ENTRY function of the EJDI gate removes a transaction or object key for a given LogicalServer.

**Input parameters**

- **entry_key**
  The key (OTS transaction or Object Key) for the entry to be removed

- **entry_type**
  Indicates whether this is a transaction or object_key entry

- **logicalserver**
  Name of the LogicalServer for which the entry is to be removed

**Output parameters**

None

There are no output parameters.

**RESPONSE** is the domain’s response to the call. It can have any of these values

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID.

Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>ENTRY_NOT_FOUND FILE_CONNECT_ERROR FILE_CORRUPT_ERROR FILE_IO_ERROR STORE_NOT_FOUND STORE_NOT_OPEN</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_KEYLENGTH</td>
</tr>
</tbody>
</table>

**EJDU gate, DUMP_DATA Function**

The DUMP_DATA function of the EJDU gate is used to collect data from a dumping class. It will be placed in the chain of data collected by EJDU and formatted out when a CICS dump occurs.

**Input parameters**

- **data**
  A pointer and length pair containing the data to be stored for inclusion in a dump.

**Output parameters**

None

There are no output parameters.

**RESPONSE** is the domain’s response to the call. It can have any of these values

OK|EXCEPTION|DISASTER
Enterprise Java domain (EJ)

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID.
Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND INTERNAL_ERROR</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>INSUFFICIENT_STORAGE</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

**EJDU gate, DUMP_STACK Function**

The DUMP_STACK function of the EJDU gate is used to collect the stack of a running JVM. The stack is passed as a string to EJDU and will be formatted out separately from the other data collected by EJDU’s DUMP_DATA function. This function should be called before DUMP_DATA as it will free any existing data gathered for the running task.

**Input parameters**
- **data** A pointer and length pair containing the data to be stored for inclusion in a dump.

**Output parameters**
- **None** There are no output parameters.

**RESPONSE**
- is the domain’s response to the call. It can have any of these values: OK|EXCEPTION|DISASTER

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID.
Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND INTERNAL_ERROR</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>INSUFFICIENT_STORAGE</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

**EJDU gate, INQUIRE_TRACE_FLAGS Function**

The INQUIRE_TRACE_FLAGS function of the EJDU gate is used to return the current settings of all the trace flags. It takes into account the master trace flag setting when returning the result. The trace flags are returned as a continuous block of storage with 2 bytes for each flag, in domain order.

**Input parameters**
- **trace_data**
  - A block of data containing the trace flags in domain order, where each trace flag takes up 2 bytes

**Output parameters**
- **None** There are no output parameters.

**RESPONSE**
- is the domain’s response to the call. It can have any of these values: OK|EXCEPTION|DISASTER
EJGE gate, INITIALIZE Function

The INITIALIZE function of the EJGE gate creates the various things in the EJE Anchor Block (Locks, Store Subpools, Statii etc.) and then sets up the initial chains of CorbaServer, DJar and BEan Control Blocks (and the Browse equivalents). These chains all start with a dummy X’00’ element and end with another dummy X’FF’ element. This permits easy chaining and detection of end-of-lists. However, more importantly, this technique enables multi-TCB operations to proceed as there are never any EJ Element wide-locks - all locks are at the CorbaServer, DJar or Bean level. After the EJE anchor block has been setup it is never subsequently amended.

Input parameters

starttype
The startup type for this CICS system (WARM|COLD)

Output parameters

None  There are no output parameters

RESPONSE
is the domain’s response to the call. It can have any of these values
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID.
Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND INTERNAL_ERROR</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>BAD_DOMAIN_TOKEN TRACE_BUFFER_ERROR</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

EJGE gate, QUIESCE Function

The QUIESCE function of the EJGE gate runs when a CEMT P SHUT is executed.

Input parameters

None  There are no input parameters

Output parameters

None  There are no output parameters

RESPONSE
is the domain’s response to the call. It can have any of these values
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID.
Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>CATALOG_ERROR LOOP PARMS_STORAGE_ERROR SETUP_ERROR STORAGE_ERROR</td>
</tr>
</tbody>
</table>
Enterprise Java domain (EJ)

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>LOOP ParMS STORAGE_ERROR SETUP_ERROR STORAGE_ERROR</td>
</tr>
</tbody>
</table>

**EJGE gate, TERMINATE Function**

The TERMINATE function of the EJGE gate runs when a CEMT P IMMED is executed.

**Input parameters**

None  There are no input parameters

**Output parameters**

None  There are no output parameters

**RESPONSE**

is the domain’s response to the call. It can have any of these values

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**[REASON]**

is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID.

Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>LOOP ParMS STORAGE_ERROR SETUP_ERROR STORAGE_ERROR</td>
</tr>
</tbody>
</table>

**EJIO gate, RESOLVE Function**

The RESOLVE function of the EJIO gate controls the operation of Resolution processing. It is called by the CEJR transaction.

**Input parameters**

None  There are no input parameters

**Output parameters**

None  There are no output parameters

**RESPONSE**

is the domain’s response to the call. It can have any of these values

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**[REASON]**

is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID.

Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>BEAN_ADD_ERROR CATALOG_ERROR EJB_INACTIVE ENV_ERROR LOOP MULTIUSE OBJECTSTORE_ERROR PARMS_STORAGE_ERROR PRIORFAIL RESC_BAD_STB RESC_GETNEXT_ERROR RESD_BAD_STB RESD_GETNEXT_ERROR RESOLV_FAIL_CS RESOLV_FAIL_DJAR SETUP_ERROR STORAGE_ERROR</td>
</tr>
</tbody>
</table>
EJIO gate, RESOLVE_CSERSERVERS Function

The RESOLVE_CSERSERVERS function of the EJIO gate scans all existing CorbaServer Control Blocks that have not been fully processed and issues a EJCG.RESOLVE_CORBASERVER on the first such CorbaServer. (both Stage one ‘copying the DJar to the Shelf’ and Stage two ‘Opening Object Stores’ Resolution Processing).

Input parameters

None  There are no input parameters

Output parameters

None  There are no output parameters

RESPONSE

is the domain’s response to the call. It can have any of these values

OK EXCEPTION DISASTER INVALID KERNERROR PURGED

[REASON]

is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID.

Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>BEAN_ADD_ERROR CATALOG_ERROR EJB_INACTIVE ENV_ERROR LOCK_ERROR LOOP OBJECTSTORE_ERROR PARMS_STORAGE_ERROR RESC_BAD_STB RESC_GETNEXT_ERROR SETUP_ERROR STORAGE_ERROR</td>
</tr>
</tbody>
</table>

EJIO gate, RESOLVE_DJARS Function

The RESOLVE_DJARS function of the EJIO gate scans all existing DJar Control Blocks that have not been fully processed and issues a EJDG.RESOLVE_DJAR on the first such DJar (both Stage one ‘copying the DJar to the Shelf’ and Stage two ‘Bean loading’ Resolution Processing).

Input parameters

None  There are no input parameters

Output parameters

None  There are no output parameters

RESPONSE

is the domain’s response to the call. It can have any of these values

OK EXCEPTION DISASTER INVALID KERNERROR PURGED

[REASON]

is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID.

Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>BEAN_ADD_ERROR CATALOG_ERROR EJB_INACTIVE ENV_ERROR LOCK_ERROR LOOP OBJECTSTORE_ERROR PARMS_STORAGE_ERROR RESD_BAD_STB RESD_GETNEXT_ERROR SETUP_ERROR STORAGE_ERROR</td>
</tr>
</tbody>
</table>
EJJO gate
The EJJO Gate provides a subset of the aforementioned EJ Domain Gates for direct use by the CICS-supplied layers residing within Java code. See the native functions for details.

EJMI gate, ADD_BEAN Function
The ADD_BEAN function of the EJMI gate adds the named Bean within the named CorbaServer to the EJMI state.
A duplicate_bean exception is returned if there is already a Bean of that name within the given CorbaServer. (The DJar must be discarded before the Bean can be added again.)

**Input parameters**
- bean  The name of the Bean to be added
- corbaserver  The name of the CorbaServer containing the Bean
- djar  The name of the DJar

**Output parameters**
None  There are no output parameters

**RESPONSE**
is the domain’s response to the call. It can have any of these values
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID.
Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>DUPLICATE_BEAN</td>
</tr>
</tbody>
</table>

EJMI gate, ADD_METHOD Function
The ADD_METHOD function of the EJMI gate adds the information for the named method within the given Bean and CorbaServer.
An unknown_bean exception is returned if there the given Bean and CorbaServer combination is not present in the EJMI state.
A duplicate_method exception is returned if there is already a method of that name within the given Bean and CorbaServer combination.

**Input parameters**
- bean  The name of the Bean to be added
- corbaserver  The name of the CorbaServer containing the Bean
- method  The name of the method
Indicates whether an external OTS transaction coordinator, if there is one, is respected for determining transaction commit or rollback.

Output parameters
None There are no output parameters

RESPONSE is the domain’s response to the call. It can have any of these values
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>DUPLICATE_METHOD UNKNOWN_BEAN</td>
</tr>
</tbody>
</table>

EJMI gate, DISCARD_METHOD_INFO Function
The DISCARD_METHOD_INFO function of the EJMI gate removes from the given CorbaServer all the information about Beans with the given DJar name. If no DJar name is specified all Beans are removed.

Input parameters
corbaserver The name of the CorbaServer for this DJar
djar The name of the DJar for these Beans

Output parameters
None There are no output parameters

RESPONSE is the domain’s response to the call. It can have any of these values
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>UNKNOWN_CORBASERVER</td>
</tr>
</tbody>
</table>

EJMI gate, GET_METHOD_INFO Function
The GET_METHOD_INFO function of the EJMI gate returns the information about the named method within the named Bean and CorbaServer.

An unknown_method exception is returned if the method is not found within the Bean and CorbaServer combination.

Input parameters
bean The name of the Bean
corbaserver The name of the CorbaServer for this Bean
Enterprise Java domain (EJ)

method

The name of the Bean method

Output parameters

xcoord (respected | ignored)

Indicates whether an external OTS transaction coordinator, if there is one, is respected for determining transaction commit or rollback

RESPONSE

is the domain’s response to the call. It can have any of these values

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON]

is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID.

Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>UNKNOWN.Bean, UNKNOWN_CORBASERVER, UNKNOWN_METHOD</td>
</tr>
</tbody>
</table>

EJMI gate, INITIALIZE Function

The INITIALIZE function of the EJMI gate initializes the EJMI state in the EJ anchor block.

Input parameters

None There are no input parameters

Output parameters

None There are no output parameters

RESPONSE

is the domain’s response to the call. It can have any of these values

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

EJOB gate, END_BROWSE_OBJECT Function

The END_BROWSE_OBJECT function of the EJOB gate is called after START_BROWSE_OBJECT to end the Browse of a file or object_store.

Input parameters

browse_token

The token returned by START_BROWSE

Output parameters

None There are no output parameters.

RESPONSE

is the domain’s response to the call. It can have any of these values

OK | EXCEPTION | DISASTER | INVALID | PURGED

[REASON]

is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID.

Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
</tbody>
</table>
### EJOB gate, GET_NEXT_OBJECT Function

The GET_NEXT_OBJECT function of the EJOB gate is called after START_BROWSE_OBJECT to return the next object in the file or object_store.

#### Input parameters

- **browse_token**
  - The token returned by START_BROWSE

- **[key_buffer]**
  - A buffer in which the next object key is returned

- **[object_buffer]**
  - A buffer in which the next object is returned

#### Output parameters

- **[active_timeout]**
  - A full-word giving the number of seconds after which Objects in the Active state may be automatically deleted from the store.

- **[file_name]**
  - The 8-character name of the file containing the Object Store.

- **[last_updated]**
  - The time in STCK seconds when the object was last stored or activated.

- **[object_size]**
  - The size of the object being inquired.

- **[passive_timeout]**
  - A full-word giving the number of seconds after which Objects in the Passive state may be automatically deleted from the store.

- **[status]**
  - The state of the Object, ACTIVE or PASSIVE.

- **[store_name]**
  - The 8-character name of the Object Store.

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

- OK
- EXCEPTION
- DISASTER
- INVALID
- PURGED

**[REASON]**

is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID.

Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>BUFFER_TOO_SMALL, END_BROWSE, FILE_CONNECT_ERROR, FILE_CORRUPT_ERROR, FILE_IO_ERROR,</td>
</tr>
<tr>
<td></td>
<td>FILE_KEY_LENGTH_ERROR, FILE_NOT_FOUND, OBJECT_CORRUPT</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_TOKEN</td>
</tr>
</tbody>
</table>
**EJOB gate, INQUIRE_OBJECT Function**

The **INQUIRE_OBJECT** function of the **EJOB** gate is called to return the Object data and attributes associated with the given key.

**Input parameters**

- **store_name**
  - The 8-character name of the Object Store
- **key_block**
  - A block giving the key of the Object being inquired
- **[object_buffer]**
  - A buffer in which the Object is returned

**Output parameters**

- **[active_timeout]**
  - A full-word giving the number of seconds after which Objects in the Active state may be automatically deleted from the store.
- **[file_name]**
  - The 8-character name of the file containing the Object Store.
- **[last_updated]**
  - The time in STCK seconds when the object was last stored or activated.
- **[object_size]**
  - The size of the Object being inquired.
- **[passive_timeout]**
  - A full-word giving the number of seconds after which Objects in the Passive state may be automatically deleted from the store.
- **[status]**
  - The state of the Object, ACTIVE or PASSIVE.

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

- OK
- EXCEPTION
- DISASTER
- INVALID
- PURGED

**[REASON]**

is returned when **RESPONSE** is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>BUFFER_TOO_SMALL FILE_CONNECT_ERROR FILE_CORRUPT_ERROR FILE_IO_ERROR FILE_KEY_LENGTH_ERROR FILE_NOT_FOUND OBJECT_CORRUPT OBJECT_NOT_FOUND STORE_NOT_OPEN</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_KEYLENGTH</td>
</tr>
</tbody>
</table>

**EJOB gate, INQUIRE_STORES Function**

The **INQUIRE_STORES** function of the **EJOB** gate is called to return a list of the Object Store names associated with the given file. The list is returned as an array of 8-character store names.
Input parameters

[file_name]
The optional 8-character name of the file to be inquired. If omitted then the default file ‘DFHEJOS’ will be used.

object_buffer | subpool
A buffer in which the array of store names is returned. Either an object buffer or a subpool token must be specified. | A storage subpool from which to getmain the object block.

Output parameters

store_count
The number of store names being returned

[object_block]
A block containing the array of 8-character store names. If specified then SUBPOOL must also be specified.

RESPONSE
is the domain’s response to the call. It can have any of these values
OK|EXCEPTION|DISASTER|INVALID|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID.
Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>BUFFER_TOO_SMALL FILE_CONNECT_ERROR FILE_CORRUPT_ERROR FILE_IO_ERROR</td>
</tr>
<tr>
<td></td>
<td>FILE_KEY_LENGTH_ERROR FILE_NOT_FOUND FILE_REC_SIZE_ERROR</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_TOKEN</td>
</tr>
</tbody>
</table>

EJOB gate, RETRIEVE_STATISTICS Function

The RETRIEVE_STATISTICS function of the EJOB gate is called by statistics to return the statistics associated with a supplied store key.

Input parameters

store_name
The name of the store for which the statistics are being retrieved.

[data] A flag indicating if the statistics must be returned

[object_buffer] The statistics buffer which the data must be put into

[reset] A flag indicating that the statistics fields must be reset

Output parameters

None There are no output parameters

RESPONSE
is the domain’s response to the call. It can have any of these values
OK|EXCEPTION|DISASTER|INVALID|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID.
Possible values are:
RESPONSE Possible REASON values

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>DISASTER</th>
<th>EXCEPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ABEND</td>
<td>BUFFER_TOO_SMALL, BUFFER_NOT_SUPPLIED, STORE_NOT_OPEN</td>
</tr>
</tbody>
</table>

### EJOB gate, START_BROWSE_OBJECT Function

The START_BROWSE_OBJECT function of the EJOB gate is called to Browse an Object Store. If STORE_NAME is omitted then all Objects in the file are browsed. If FILE_NAME is omitted then the default file ‘DFHEJOS’ is assumed.

**Input parameters**

- **store_name**: The 8-character name of the Object Store to browse
- **file_name**: The 8-character name of the Object Store to browse

**Output parameters**

- **browse_token**: A token required by GET_NEXT and END_BROWSE

**RESPONSE**

- is the domain’s response to the call. It can have any of these values: OK, EXCEPTION, DISASTER, INVALID, PURGED

**[REASON]**

- is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID.

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>DISASTER</th>
<th>EXCEPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ABEND mass</td>
<td>FILE_CONNECT_ERROR, FILE_CORRUPT_ERROR, FILE_IO_ERROR, FILE_KEY_LENGTH_ERROR, FILE_NOT_FOUND, FILE_REC_SIZE_ERROR, STORE_NOT_FOUND</td>
</tr>
</tbody>
</table>

### EJOS gate, ACTIVATE_OBJECT Function

The ACTIVATE_OBJECT function of the EJOS gate is called to Activate an Object instance.

If DELETE(NO) is specified then the ACTIVATE function will mark Objects as ACTIVE in the store. ACTIVE_TIMEOUT indicates when entries marked ACTIVE may be automatically deleted from the store. If DELETE(YES) is specified then ACTIVE_TIMEOUT is ignored, and the ACTIVATE function will delete the object from the store.

**Input parameters**

- **store_name**: The 8-character name of the Object Store
- **key_block**: A block giving the key of the Object to be activated
- **object_buffer**: The buffer into which the Object is returned
- **delete**: YES means the Object is to be deleted from the while and NO means the Object is to be marked ACTIVE in the file
Output parameters

object_size
The size of the Object being activated

RESPONSE
is the domain’s response to the call. It can have any of these values
OK|EXCEPTION|DISASTER|INVALID|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID.
Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>BUFFER_TOO_SMALL FILE_CONNECT_ERROR FILE_CORRUPT_ERROR FILE_IO_ERROR FILE_KEY_LENGTH_ERROR OBJECT_CORRUPT OBJECT_IS_ACTIVE OBJECT_NOT_FOUND STORE_NOT_OPEN</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_KEYLENGTH</td>
</tr>
</tbody>
</table>

EJOS gate, CLOSE_OBJECT_STORE Function
The CLOSE_OBJECT_STORE function of the EJOS gate is called to Close an Object Store in the local system.

If an Object Store is open with a non-zero timeout value, then a task is scheduled to sweep the store periodically, deleting timed-out Objects. It will, therefore, improve CICS performance if stores are closed when not required.

Input parameters

store_name
The 8-character name of the Object Store

Output parameters

None
There are no output parameters.

RESPONSE
is the domain’s response to the call. It can have any of these values
OK|EXCEPTION|DISASTER|INVALID|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID.
Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>STORE_NOT_OPEN</td>
</tr>
</tbody>
</table>

EJOS gate, OPEN_OBJECT_STORE Function
The OPEN_OBJECT_STORE function of the EJOS gate is called to Open a new or existing Object Store in the local system.

An Object Store must be opened in each region wishing to use it. Many object stores can use the same CICS file, or they can each specify a different file.

If an Object Store of the same name is already open in that region, the existing definition is replaced, and the new file name and timeout values are then used. As
timeout values are stored with the object, changes to the store definition will not affect objects already stored. If file_name is omitted, then the default file 'DFHEJOS' will be used.

**Input parameters**

*activate_timeout*
A full-word giving the number of seconds after which Objects in the Active State may be automatically deleted from the store

*passive_timeout*
A full-word giving the number of seconds after which Objects in the Passive State may be automatically deleted from the store

*recovery*
YES indicates that the file should be recoverable. If it is not, FILE_RECOVERY_ERROR is returned. NO indicates that the file should not be recoverable. If it is then FILE_RECOVERY_ERROR is returned. If CICS is unable to determine whether the file is recoverable then FILE_RECOVERY_UNKNOWN is returned

*store_name*
The 8-character name of the Object Store

[file_name]
The optional 8-character name of the file to be used

**Output parameters**

None There are no output parameters.

RESPONSE is the domain’s response to the call. It can have any of these values

OK|EXCEPTION|DISASTER|INVALID|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>FILE_CONNECT_ERROR FILE_CORRUPT_ERROR FILE_IO_ERROR FILE_KEY_LENGTH_ERROR FILE_NOT_FOUND FILE_RECOVERY_ERROR FILE_RECOVERY_UNKNOWN FILE_REC_SIZE_ERROR</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_OBJECT_TIMEOUT</td>
</tr>
</tbody>
</table>

**EJOS gate, REMOVE_OBJECT Function**

The REMOVE_OBJECT function of the EJOS gate is called to Remove an Object instance from the specified Object Store.

**Input parameters**

*key_block*
A block containing the key of the Object to be removed

*store_name*
The 8-character name of the Object Store

**Output parameters**

None There are no output parameters.
RESPONSE is the domain’s response to the call. It can have any of these values:
OK | EXCEPTION | DISASTER | INVALID | PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>FILE_CONNECT_ERROR FILE_CORRUPT_ERROR FILE_IO_ERROR FILE_KEY_LENGTH_ERROR FILE_NOT_FOUND OBJECT_NOT_FOUND STORE_NOT_OPEN</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_KEYLENGTH</td>
</tr>
</tbody>
</table>

**EJOS gate, REMOVE_STORE Function**

The REMOVE_STORE function of the EJOS gate is called to Remove one or all Object Stores from the specified file.

When a Store is removed, it should be removed or closed in every region in which it is open. If not, then data may be lost.

**Input parameters**

- **store_name** | all
  - The 8-character name of the Object Store

- **[file_name]**
  - The optional 8-character name of the file to be used

**Output parameters**

- **None** There are no output parameters.

RESPONSE is the domain’s response to the call. It can have any of these values:
OK | EXCEPTION | DISASTER | INVALID | PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>FILE_CONNECT_ERROR FILE_CORRUPT_ERROR FILE_IO_ERROR FILE_KEY_LENGTH_ERROR FILE_NOT_FOUND STORE_NOT_OPEN</td>
</tr>
</tbody>
</table>

**EJOS gate, STORE_OBJECT Function**

The STORE_OBJECT function of the EJOS gate is called to Store an Object instance.

The Object is identified by a KEY of from 1 to (recordsize -64) bytes, and the Object can be of any size.

If no Object with that key exists in the store then one is created in the Passive state. If an Object with the same key already exists in the Store, then the action depends on the REPLACE value (YES | NO | ACTIVE | PASSIVE). An exception OBJECT_IS_ACTIVE or OBJECT_IS_PASSIVE indicates why an object was not replaced.
Enterprise Java domain (EJ)

**Input parameters**

**store_name**

The 8-character name of the Object Store

**replace**(yes | no | active | passive)

Yes means that an Object with the same key will be replaced. NO means that an Object with the same key will not be replaced. ACTIVE means that an ACTIVE Object with the same key is replaced. PASSIVE means that a PASSIVE Object with the same key is replaced

**key_block**

A block containing the key of the Object to be stored

**object_block**

A block containing the Object data to be stored

**Output parameters**

None There are no output parameters.

RESPONSE

is the domain’s response to the call. It can have any of these values

OK | EXCEPTION | DISASTER | INVALID | PURGED

[REASON]

is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID.

Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>FILE_CONNECT_ERROR FILE_CORRUPT_ERROR FILE_FULL_ERROR FILE_IO_ERROR FILE_KEY_LENGTH_ERROR FILE_NOT_FOUND OBJECT_IS_ACTIVE OBJECT_IS_PASSIVE STORE_NOT_OPEN</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_KEYLENGTH</td>
</tr>
</tbody>
</table>

**EJ domain’s generic gates**

Table 51 summarizes the EJ domain’s generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>EJDM</td>
<td>EJ 01xx</td>
<td>INITIALISE_DOMAIN QUESCE_DOMAIN TERMINATE_DOMAIN</td>
<td>DMDM</td>
</tr>
<tr>
<td>EJST</td>
<td>EJ 04xx</td>
<td>COLLECT_STATISTICS COLLECT_Resource_STATS</td>
<td>STST</td>
</tr>
</tbody>
</table>

For descriptions of these functions and their input and output parameters, refer to the §§ dealing with the corresponding generic formats:

**Functions and parameters**

Format DMDM—Chapter 28. Domain manager domain (DM) on page 369

Format STST—System programming command flows on page 658
The Domain Manager gates perform normal internal state initialization and termination functions.

**Modules**

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHEJBB</td>
<td>Bean Browse EJBB Gate</td>
</tr>
<tr>
<td>DFHEJBG</td>
<td>Bean General EJBG Gate</td>
</tr>
<tr>
<td>DFHEJCB</td>
<td>CorbaServer Browse EJCB Gate</td>
</tr>
<tr>
<td>DFHEJCG</td>
<td>CorbaServer General EJCG Gate</td>
</tr>
<tr>
<td>DFHEJCP</td>
<td>Command Processor functions EJCP Gate</td>
</tr>
<tr>
<td>DFHEJDB</td>
<td>DJar Browse EJDB Gate</td>
</tr>
<tr>
<td>DFHEJDG</td>
<td>DJar General EJDG Gate</td>
</tr>
<tr>
<td>DFHEJDI</td>
<td>EJB Directory EJDI Gate</td>
</tr>
<tr>
<td>DFHEJDM</td>
<td>EJ Initialize/Terminate EJDM Gate</td>
</tr>
<tr>
<td>DFHEJDU</td>
<td>EJ Dump Interface EJDU Gate</td>
</tr>
<tr>
<td>DFHEJGE</td>
<td>EJ General Initialization/Termination functions EJGE Gate</td>
</tr>
<tr>
<td>DFHEJIO</td>
<td>CEJR Resolution EJIO Gate</td>
</tr>
<tr>
<td>DFHEJJO</td>
<td>Jave Interface EJJO Gate</td>
</tr>
<tr>
<td>DFHEJMI</td>
<td>Method Information function EJMI Gate</td>
</tr>
<tr>
<td>DFHEJOB</td>
<td>Object Store Browse EJOB Gate</td>
</tr>
<tr>
<td>DFHEJOS</td>
<td>Object Store General EJOS Gate</td>
</tr>
<tr>
<td>DFHEJST</td>
<td>Statistics General EJSt Gate</td>
</tr>
</tbody>
</table>

**Exits**

NONE

**Trace**

The point IDs have just been added onto the usual EJ ones.
Chapter 34. Event manager domain (EM)

The event manager domain manages event and timer objects created within CICS BTS activities. For further information regarding these objects see the CICS Business Transaction Services manual.

Event manager domain’s specific gates

Table 52 summarizes the event manager domain’s specific gates. It shows the level-1 trace point IDs of the modules providing the functions for the gate, the functions provided by the gate, and whether or not the functions are available through the exit programming interface (XPI).

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>XPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMEM</td>
<td>EM 0201</td>
<td>ADD_SUBEVENT</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>EM 0202</td>
<td>CHECK_TIMER</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DEFINE_ATOMIC_EVENT</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DEFINE_COMPOSITE_EVENT</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DEFINE_TIMER</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DELETE_EVENT</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DELETE_TIMER</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FIRE_EVENT</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FORCE_TIMER</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INQUIRE_STATUS</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>REMOVE_SUBEVENT</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RESET_EVENT</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RETRIEVE_REATTACH_EVENT</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RETRIEVE_SUBEVENT</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TEST_EVENT</td>
<td>NO</td>
</tr>
<tr>
<td>EMBR</td>
<td>EM 0301</td>
<td>INQUIRE_EVENT</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>EM 0302</td>
<td>START_BROWSE_EVENT</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GET_NEXT_EVENT</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>END_BROWSE_EVENT</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INQUIRE_TIMER</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>START_BROWSE_TIMER</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GET_NEXT_TIMER</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>END_BROWSE_TIMER</td>
<td>NO</td>
</tr>
</tbody>
</table>

EMEM gate, ADD_SUBEVENT function

The ADD_SUBEVENT function adds a subevent to an existing composite event.

Input parameters

EVENT

is the name of the composite event.

SUBEVENT

is the name of the subevent.

Output parameters

RESPONSE

is the domain’s response to the call. It can have any of these values:
Event Manager Domain (EM)

RESPONSE Possible REASON values

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NO_CURRENT_ACTIVITY, EVENT_NOT_FOUND, INVALID_EVENT_TYPE,</td>
</tr>
<tr>
<td></td>
<td>SUBEVENT_NOT_FOUND, INVALID_SUBEVENT</td>
</tr>
</tbody>
</table>

**EMEM gate, CHECK_TIMER function**

The CHECK_TIMER function returns the status of a timer.

**Input parameters**

**TIMER_NAME**

is the name of the timer.

**Output parameters**

**TIMER_STATUS**

returns the status of the timer. It can have one of these values:

- EXPIRED
- FORCED
- UNEXPIRED

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

- OK
- EXCEPTION
- INVALID
- DISASTER
- KERNERROR
- PURGED

[REASON]

is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NO_CURRENT_ACTIVITY, TIMER_NOT_FOUND</td>
</tr>
</tbody>
</table>

**EMEM gate, DEFINE.Atomic EVENT function**

The DEFINE_ATOMIC_EVENT function defines an atomic event of type ACTIVITY or INPUT.

**Input parameters**

**EVENT**

is the name of the event.

**EVENT_TYPE**

is the type of the event. It can have one of these values:

- ACTIVITY
- INPUT

**Output parameters**

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

- OK
- EXCEPTION
- INVALID
- DISASTER
- KERNERROR
- PURGED

[REASON]

is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NO_CURRENT_ACTIVITY, INVALID_EVENT_NAME, DUPLICATE_EVENT</td>
</tr>
</tbody>
</table>
**EMEM gate, DEFINE_COMPOSITE_EVENT function**

The DEFINE_COMPOSITE_EVENT function defines a composite event with an associated predicate which may be AND or OR. Up to eight subevents may be provided.

**Input parameters**

**EVENT**
- is the name of the composite event.

**PREDICATE**
- is the predicate type. It may have either one of these values: AND|OR

**SUBEVENT_LIST**
- is an optional list of up to 8 subevents.

**Output parameters**

**SUBEVENT_IN_ERROR**
- returns the number of the first subevent which is in error (if any).

**RESPONSE**
- is the domain’s response to the call. It can have any of these values: OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

**[REASON]**
- is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NO_CURRENT_ACTIVITY, INVALID_EVENT_NAME, SUBEVENT_NOT_FOUND, INVALID_SUBEVENT, DUPLICATE_EVENT</td>
</tr>
</tbody>
</table>

---

**EMEM gate, DEFINE_TIMER function**

The DEFINE_TIMER function defines a timer.

**Input parameters**

**TIMER_NAME**
- is the name of the timer.

**EVENT**
- is the optional name of an event to be associated with the timer.

**AFTER**
- indicates whether or not the timer is an interval. It may have either of these values: YES|NO

**AT**
- indicates whether or not the timer is a time. It may have either of these values: YES|NO

**DAYS**
- is the number of days for an interval.

**HOURS**
- is the number of hours for an interval or time.

**MINUTES**
- is the number of minutes for an interval or time.
Event Manager Domain (EM)

SECONDS
is the number of seconds for an interval or time.

ON indicates whether or not a date has been specified. It may have either of these values:
YES|NO

YEAR is the year.

MONTH is the month.

DAYOFMONTH is the day of the month.

DAYOFYEAR is the day of the year.

Output parameters
RESPONSE is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NO_CURRENT_ACTIVITY, INVALID_TIMER_NAME, DUPLICATE_TIMER, INVALID_EVENT_NAME, DUPLICATE_EVENT, INVALID_INTERVAL, INVALID_TIME</td>
</tr>
</tbody>
</table>

EMEM gate, DELETE_EVENT function
The DELETE_EVENT function deletes an event.

Input parameters
EVENT
is the name of the event to be deleted.

Output parameters
RESPONSE is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NO_CURRENT_ACTIVITY, EVENT_NOT_FOUND, INVALID_EVENT_TYPE</td>
</tr>
</tbody>
</table>

EMEM gate, DELETE_TIMER function
The DELETE_TIMER function deletes a timer.

Input parameters
TIMER
is the name of the timer to be deleted.
Output parameters

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK\|EXCEPTION\|INVALID\|DISASTER\|KERNERROR\|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NO_CURRENT_ACTIVITY, TIMER_NOT_FOUND</td>
</tr>
</tbody>
</table>

EMEM gate, FIRE_EVENT function
The FIRE_EVENT function causes an event to fire.

Input parameters

EVENT
is the name of the event to be fired.

EVENT_VERSION
is an optional version number for the event.

Output parameters

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK\|EXCEPTION\|INVALID\|DISASTER\|KERNERROR\|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NO_CURRENT_ACTIVITY, EVENT_NOT_FOUND, INVALID_EVENT_TYPE, ALREADY_FIRED, VERSION_NOT_FOUND</td>
</tr>
</tbody>
</table>

EMEM gate, FORCE_TIMER function
The FORCE_TIMER function causes a timer to expire early.

Input parameters

TIMER
is the name of the timer to be forced.

ACQUIRED_PROCESS
indicates whether or not the timer to be forced is owned by the acquired process. It may have either of these values:
YES\|NO

ACQUIRED_ACTIVITY
indicates whether or not the timer to be forced is owned by the acquired activity. It may have either of these values:
YES\|NO

Output parameters

RESPONSE
is the domain’s response to the call. It can have any of these values:
Event Manager Domain (EM)

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NO_CURRENT_ACTIVITY, NO_ACQUIRED_PROCESS, NO_ACQUIRED_ACTIVITY, INVALID_ACTIVITY, TIMER_NOT_FOUND</td>
</tr>
</tbody>
</table>

EMEM gate, INQUIRE_STATUS function

The INQUIRE_STATUS function returns the status of the event pool for the current activity.

**Output parameters**

**PENDING_EVENTS**
indicates whether any events are pending. It may have either of these values:
YES|NO

**PENDING_ACTIVITY_EVENTS**
indicates whether any activity events are pending. It may have either of these values:
YES|NO

**REATTACH**
indicates whether the task should be reattached. It may have either of these values:
YES|NO

**EVENTS_PROCESSED**
indicates whether any events were processed during this activation. It may have either of these values:
YES|NO

**RESPONSE**
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NO_CURRENT_ACTIVITY</td>
</tr>
</tbody>
</table>

EMEM gate, REMOVE_SUBEVENT function

The REMOVE_SUBEVENT function removes a subevent from the named composite event.

**Input parameters**

**EVENT**
is the name of the composite event.

**SUBEVENT**
is the name of the subevent.
Output parameters
RESPONSE is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NO_CURRENT_ACTIVITY, EVENT_NOT_FOUND, INVALID_EVENT_TYPE,</td>
</tr>
<tr>
<td></td>
<td>SUBEVENT_NOT_FOUND, INVALID_SUBEVENT</td>
</tr>
</tbody>
</table>

**EMEM gate, RETRIEVE_REATTACH_EVENT function**

The RETRIEVE_REATTACH_EVENT function retrieves the next event from the current activity’s reattach queue.

Output parameters
EVENT is the name of the retrieved reattach event.

EVENT_TYPE is the type of the retrieved reattach event. It may have one of the following values:
ACTIVITY|COMPOSITE|INPUT|SYSTEM|TIMER

RESPONSE is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NO_CURRENT_ACTIVITY, END_EVENTS</td>
</tr>
</tbody>
</table>

**EMEM gate, RETRIEVE_SUBEVENT function**

The RETRIEVE_SUBEVENT function retrieves the next event from the named composite event’s subevent queue.

Input parameters
EVENT is the name of the composite event.

Output parameters
SUBEVENT is the name of the subevent.

EVENT_TYPE is the type of the retrieved reattach event. It may have one of the following values:
ACTIVITY|COMPOSITE|INPUT|SYSTEM|TIMER

RESPONSE is the domain’s response to the call. It can have any of these values:
**Event Manager Domain (EM)**

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NO_CURRENT_ACTIVITY, EVENT_NOT_FOUND, INVALID_EVENT_TYPE, END_SUBEVENTS, NO_SUBEVENTS</td>
</tr>
</tbody>
</table>

**EMEM gate, TEST_EVENT function**

The TEST_EVENT function returns the fire status of the named event.

**Input parameters**

**EVENT**

is the name of the event to be tested.

**Output parameters**

**FIRED**

returns the fire status of the event. It may have either of these values:

YES|NO

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

**[REASON]**

is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NO_CURRENT_ACTIVITY, EVENT_NOT_FOUND</td>
</tr>
</tbody>
</table>

**EMBR gate, INQUIRE_EVENT function**

The INQUIRE_EVENT function returns information about the named event.

**Input parameters**

**EVENT**

is the name of the event being inquired upon.

**Output parameters**

**EVENT_TYPE**

is the type of the event. It can have one of these values:

ACTIVITY|COMPOSITE|INPUT|SYSTEM|TIMER

**FIRED**

returns the fire status of the event. It may have either of these values:

YES|NO

**PREDICATE**

is the predicate type (for composite events only). It may have either one of these values:

AND|OR

**PARENT**

is the name of the parent (if the event is a subevent).
**Event Manager Domain (EM)**

**TIMER_NAME**  
is the name of the associated timer (if the event is of type timer).

**RESPONSE**  
is the domain’s response to the call. It can have any of these values:  
OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

**[REASON]**  
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>INVALID_ACTIVITY_ID, NO_CURRENT_ACTIVITY, FILE_NOT_AUTH, EVENT_NOT_FOUND, READ_FAILURE, FILE_UNAVAILABLE</td>
</tr>
</tbody>
</table>

**EMBR gate, START_BROWSE_EVENT function**  
The START_BROWSE_EVENT function starts an event browse and returns a token to be used for the browse.

**Input parameters**  
**ACTIVITY_ID**  
is an optional activity id for the activity whose event pool is to be browsed.

**Output parameters**  
**BROWSE_TOKEN**  
returns a token which is used to identify the browse.

**RESPONSE**  
is the domain’s response to the call. It can have any of these values:  
OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

**[REASON]**  
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>INVALID_ACTIVITY_ID, FILE_NOT_AUTH, NO_CURRENT_ACTIVITY, READ_FAILURE, FILE_UNAVAILABLE</td>
</tr>
</tbody>
</table>

**EMBR gate, GET_NEXT_EVENT function**  
The GET_NEXT_EVENT function returns the next name in the browse specified by the browse token, and returns the attributes associated with the event.

**Input parameters**  
**BROWSE_TOKEN**  
is a token which identifies the browse.

**Output parameters**  
**EVENT**  
is the name of the event.

**EVENT_TYPE**  
is the type of the event. It can have one of these values:  
ACTIVITY|COMPOSITE|INPUT|SYSTEM|TIMER
Event Manager Domain (EM)

**FIRED**
returns the fire status of the event. It may have either of these values:
YES | NO

**PREDICATE**
is the predicate type (for composite events only). It may have either one of these values:
AND | OR

**PARENT**
is the name of the parent (if the event is a subevent).

**TIMER_NAME**
is the name of the associated timer (if the event is of type timer).

**RESPONSE**
is the domain’s response to the call. It can have any of these values:
OK | EXCEPTION | INVALID | DISASTER | KERNERROR | PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>INVALID_BROWSE_TOKEN, BROWSE_END</td>
</tr>
</tbody>
</table>

**EMBR gate, END_BROWSE_EVENT function**
The END_BROWSE_EVENT function ends the event browse identified by the browse token.

**Input parameters**

**BROWSE_TOKEN**
is a token which identifies the browse.

**Output parameters**

**RESPONSE**
is the domain’s response to the call. It can have any of these values:
OK | EXCEPTION | INVALID | DISASTER | KERNERROR | PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>INVALID_BROWSE_TOKEN</td>
</tr>
</tbody>
</table>

**EMBR gate, INQUIRE_TIMER function**
The INQUIRE_TIMER function returns information about the named timer.

**Input parameters**

**TIMER**
is the name of the timer being inquired upon.

**Output parameters**

**EVENT**
is the name of the associated event.
**TIMER_STATUS**

is the status of the timer. It can have one of these values:

- EXPIRED
- FORCED
- UNEXPIRED

**ABSTIME**

returns the timer’s expiry time in ABSTIME format.

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

- OK
- EXCEPTION
- INVALID
- DISASTER
- KERNERROR
- PURGED

[REASON]

is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>INVALID_ACTIVITY_ID, NO_CURRENT_ACTIVITY, FILE_NOT_AUTH, TIMER_NOT_FOUND, READ_FAILURE, FILE_UNAVAILABLE</td>
</tr>
</tbody>
</table>

**EMBR gate, START_BROWSE_TIMER function**

The START_BROWSE_TIMER function starts a timer browse and returns a token to be used for the browse.

**Input parameters**

**ACTIVITY_ID**

is an optional activity id for the activity whose event pool is to be browsed.

**Output parameters**

**BROWSE_TOKEN**

returns a token which is used to identify the browse.

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

- OK
- EXCEPTION
- INVALID
- DISASTER
- KERNERROR
- PURGED

[REASON]

is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>INVALID_ACTIVITY_ID, FILE_NOT_AUTH, NO_CURRENT_ACTIVITY, READ_FAILURE, FILE_UNAVAILABLE</td>
</tr>
</tbody>
</table>

**EMBR gate, GET_NEXT_TIMER function**

The GET_NEXT_TIMER function returns the next name in the browse specified by the browse token, and returns the attributes associated with the timer.

**Input parameters**

**BROWSE_TOKEN**

is the token which identifies the browse.

**Output parameters**

**TIMER**

is the name of the timer.
Event Manager Domain (EM)

**EVENT**

is the name of the associated event.

**TIMER_STATUS**

is the status of the timer. It can have one of these values:

<table>
<thead>
<tr>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXPIRED</td>
</tr>
<tr>
<td>FORCED</td>
</tr>
<tr>
<td>UNEXPIRED</td>
</tr>
</tbody>
</table>

**ABSTIME**

returns the timer’s expiry time in ABSTIME format.

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

<table>
<thead>
<tr>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>OK</td>
</tr>
<tr>
<td>EXCEPTION</td>
</tr>
<tr>
<td>INVALID</td>
</tr>
<tr>
<td>DISASTER</td>
</tr>
<tr>
<td>KERNERROR</td>
</tr>
<tr>
<td>PURGED</td>
</tr>
</tbody>
</table>

**[REASON]**

is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>INVALID_BROWSE_TOKEN, BROWSE_END</td>
</tr>
</tbody>
</table>

**EMBR gate, END_BROWSE_TIMER function**

The END_BROWSE_TIMER function ends the timer browse identified by the browse token.

**Input parameters**

**BROWSE_TOKEN**

is a token which identifies the browse.

**Output parameters**

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

<table>
<thead>
<tr>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>OK</td>
</tr>
<tr>
<td>EXCEPTION</td>
</tr>
<tr>
<td>INVALID</td>
</tr>
<tr>
<td>DISASTER</td>
</tr>
<tr>
<td>KERNERROR</td>
</tr>
<tr>
<td>PURGED</td>
</tr>
</tbody>
</table>

**[REASON]**

is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>INVALID_BROWSE_TOKEN</td>
</tr>
</tbody>
</table>

**Event manager domain’s generic gates**

Table 53 summarizes the event manager domain’s generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

<table>
<thead>
<tr>
<th>Table 53. Event manager domain’s generic gates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gate</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td>DMDM</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>EMBA</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
For descriptions of these functions and their input and output parameters, refer to the §§ dealing with the corresponding generic formats:

### Functions and parameters

Format DMDM—"Domain manager domain’s generic formats” on page 377

In initialization, quiesce, and termination processing, the event manager domain performs only internal routines.

## Modules

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHEMDM</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>INITIALIZE_DOMAIN</td>
</tr>
<tr>
<td></td>
<td>QUIESCE_DOMAIN</td>
</tr>
<tr>
<td></td>
<td>TERMINATE_DOMAIN</td>
</tr>
<tr>
<td>DFHEMEM</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>ADD_SUBEVENT</td>
</tr>
<tr>
<td></td>
<td>CHECK_TIMER</td>
</tr>
<tr>
<td></td>
<td>DEFINE_ATOMIC_EVENT</td>
</tr>
<tr>
<td></td>
<td>DEFINE_COMPOSITE_EVENT</td>
</tr>
<tr>
<td></td>
<td>DEFINE_TIMER</td>
</tr>
<tr>
<td></td>
<td>DELETE_EVENT</td>
</tr>
<tr>
<td></td>
<td>DELETE_TIMER</td>
</tr>
<tr>
<td></td>
<td>FIRE_EVENT</td>
</tr>
<tr>
<td></td>
<td>FORCE_TIMER</td>
</tr>
<tr>
<td></td>
<td>INQUIRE_STATUS</td>
</tr>
<tr>
<td></td>
<td>REMOVE_SUBEVENT</td>
</tr>
<tr>
<td></td>
<td>RESET_EVENT</td>
</tr>
<tr>
<td></td>
<td>RETRIEVE_REATTACH_EVENT</td>
</tr>
<tr>
<td></td>
<td>RETRIEVE_SUBEVENT</td>
</tr>
<tr>
<td></td>
<td>TEST_EVENT</td>
</tr>
<tr>
<td>DFHEMBR</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>INQUIRE_EVENT</td>
</tr>
<tr>
<td></td>
<td>START_BROWSE_EVENT</td>
</tr>
<tr>
<td></td>
<td>GET_NEXT_EVENT</td>
</tr>
<tr>
<td></td>
<td>END_BROWSE_EVENT</td>
</tr>
<tr>
<td></td>
<td>INQUIRE_TIMER</td>
</tr>
<tr>
<td></td>
<td>START_BROWSE_TIMER</td>
</tr>
<tr>
<td></td>
<td>GET_NEXT_TIMER</td>
</tr>
<tr>
<td></td>
<td>END_BROWSE_TIMER</td>
</tr>
<tr>
<td>DFHEMBA</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>INQUIRE_DATA_LENGTH</td>
</tr>
<tr>
<td></td>
<td>GET_DATA</td>
</tr>
<tr>
<td></td>
<td>DESTROY_TOKEN</td>
</tr>
<tr>
<td>DFHEMDUF</td>
<td>Formats the EM domain control blocks</td>
</tr>
<tr>
<td>DFHEMTRI</td>
<td>Interprets EM domain trace entries</td>
</tr>
</tbody>
</table>

## Exits

No global user exit points are provided in this domain.
Trace

The point IDs for the event manager domain are of the form EM xxxx; the corresponding trace levels are EM 1, EM 2, and Exc.

For more information about the trace points, see the [CICS Trace Entries](#). For more information about using traces in problem determination, see the [CICS Problem Determination Guide](#).
Chapter 35. EXEC interface

The EXEC interface provides the support for application programs containing EXEC CICS commands.

Design overview

The relevant parts of the EXEC interface are:

- The main EXEC interface module, DFHEIP, which is called when an EXEC CICS command is executed in a user application program.
  A parameter list is passed, in which the first argument (referred to as arg-zero) contains a group code and a function code as the first 2 bytes.
  - The group code in general indicates the CICS component associated with the command being executed. In subsequent processing it is this code alone which determines which EXEC processor module (see below) is called from DFHEIP.
  - The function code identifies the actual command being executed.

  **Note:** DFHEIP is link-edited with other modules to form the application interface program (DFHAIP) load module. DFHEIPA (next to be described) is one of these modules.

- The DFHEIPA module, which handles the allocation and freeing of dynamic storage (mapped by DFHEISTG) for assembler-language application programs in response to DFHEIENT and DFHEIRET calls respectively.

- A set of EXEC processor modules, each of which is called from DFHEIP, and which performs the first level of analysis of the command being executed. The processor then calls the appropriate CICS domain to complete the execution of the command.

- A set of EXEC stubs, one for each of the application languages: COBOL, PL/I, C/370, and assembler language. The appropriate stub must be link-edited at the front of each CICS application program, and provides the mechanism for getting to the correct entry points in DFHEIP.

- The DFHAPLI module, which is called at the initialization and termination of each application program.

Control blocks

The control blocks associated with the EXEC interface are as follows:

**EXEC interface block (EIB) (DSECT name: DFHEIBLK).**
Each task in a command-level environment has a control block called the EXEC interface block (EIB) associated with it. The EIB is used for direct communication between command-level programs and CICS.

The EIB contains information that is useful during the execution of an application program, such as the transaction identifier, the time and date (initially when the task is started, and subsequently, if updated by the application program), and the cursor position on a display device. The EIB also contains information that is helpful when a dump is being used to debug a program. DFHEIBLK defines the layout of an EIB, and is included automatically in the application program, giving access to all of the fields in the EIB by name.
A further EIB, known as the “system” EIB, exists for each task. The system EIB has the same format as the “user” (or “application”) EIB. It is intended for use mainly by CICS system code. In general, application programs have addressability to the user EIB only, which is a copy taken of the system EIB at appropriate times. However, any service programs translated with the SYSEIB option have addressability to the system EIB also, so that they can issue EXEC CICS commands without causing the user EIB to be updated. (See the [CICS Application Programming Guide](#) for further information about the SYSEIB translator option.)

**Figure 54** shows the format of an EIB.

### EXEC interface communication area (DSECT name: DFHEICDS)

The EXEC interface communication area describes the storage that is used
to pass the COMMAREA from one command-level transaction to another using an EXEC CICS RETURN command with the TRANSID, COMMAREA, and LENGTH options.

Figure 55 shows the format of the EXEC interface communication area.

EXEC interface storage (EIS) (DSECT name: DFHEISDS).

The EXEC interface storage is used by DFHEIP as the interface between the application program and CICS control blocks. It contains a system area used by DFHEIP only. EIS is storage acquired by the DFHAPXM module (part of the transaction manager), along with other task-lifetime storage such as the TCA and both system and user EIBs. There is one EIS per transaction (not per program), and it is addressed by TCAEISA in the TCA. (See Figure 56.)

Figure 56. EXEC interface storage (EIS)

See the CICS Data Areas manual for a detailed description of these control blocks.

Modules

The EXEC interface comprises the following modules:
• The main interface module (DFHEIP)
• Prologue and epilogue code for assembler-language programs (DFHEIPA)
• 55 EXEC interface processors
• 4 EXEC stubs.

Of the EXEC interface processors, 16 are coded in Assembler language; the other modules are coded in other languages. All are CICS nucleus modules.
EXEC interface

These processor modules (together with DFHEIP) support the EXEC CICS commands listed in Table 54.

DFHEIP also supports EXEC DLI commands, by passing them through the external resource manager interface program, DFHERM, on their way to DFHEDP for conversion to standard CALL parameter lists acceptable to DL/I.

The following tables list all the EXEC CICS commands, showing the class of each command (basic or special), its group and function codes, also the name and language of the associated EXEC interface processor. Table 54 is ordered by command name. “Chapter 33. Enterprise Java domain (EJ)” on page 451 is ordered by group/function code.

The group and function codes used by the Front End Programming Interface (FEPI) feature are not listed in these tables. However, the EXEC CICS FEPI commands use group codes of 82 (API-type commands) and 84 (SPI-type commands). For details about the EXEC CICS FEPI commands, see the CICS Front End Programming Interface User’s Guide.

Note: An asterisk (*) after a command name in the tables shows that the command is intended for CICS internal use only.

Table 54. EXEC CICS commands ordered by command name

<table>
<thead>
<tr>
<th>Command</th>
<th>Class</th>
<th>Gp/fn code</th>
<th>Module</th>
<th>Lang</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABEND</td>
<td>B</td>
<td>0E 0C</td>
<td>EPC</td>
<td>A</td>
</tr>
<tr>
<td>ACQUIRE TERMINAL</td>
<td>S</td>
<td>86 02</td>
<td>EIACQ</td>
<td>O</td>
</tr>
<tr>
<td>ADDRESS</td>
<td>B</td>
<td>02 02</td>
<td>EEI</td>
<td>A</td>
</tr>
<tr>
<td>ADDRESS SET</td>
<td>B</td>
<td>02 10</td>
<td>EEI</td>
<td>A</td>
</tr>
<tr>
<td>ALLOCATE</td>
<td>B</td>
<td>04 20</td>
<td>ETC</td>
<td>A</td>
</tr>
<tr>
<td>ASKTIME</td>
<td>B</td>
<td>10 02</td>
<td>EEI</td>
<td>O</td>
</tr>
<tr>
<td>ASKTIME ABSTIME</td>
<td>B</td>
<td>4A 02</td>
<td>EIDTI</td>
<td>O</td>
</tr>
<tr>
<td>ASSIGN</td>
<td>B</td>
<td>02 08</td>
<td>EEI</td>
<td>A</td>
</tr>
<tr>
<td>BIF DEEDIT</td>
<td>B</td>
<td>20 02</td>
<td>EBF</td>
<td>A</td>
</tr>
<tr>
<td>BUILD ATTACH</td>
<td>B</td>
<td>04 26</td>
<td>ETC</td>
<td>A</td>
</tr>
<tr>
<td>CANCEL</td>
<td>B</td>
<td>10 0C</td>
<td>EEI</td>
<td>O</td>
</tr>
<tr>
<td>CHANGE TASK</td>
<td>B</td>
<td>5E 06</td>
<td>EIQSK</td>
<td>O</td>
</tr>
<tr>
<td>COLLECT STATISTICS</td>
<td>S</td>
<td>70 08</td>
<td>EIQMS</td>
<td>O</td>
</tr>
<tr>
<td>CONNECT PROCESS</td>
<td>B</td>
<td>04 32</td>
<td>ETC</td>
<td>A</td>
</tr>
<tr>
<td>CONVERSE</td>
<td>B</td>
<td>04 06</td>
<td>ETC</td>
<td>A</td>
</tr>
<tr>
<td>CREATE CONNECTION</td>
<td>S</td>
<td>30 0E</td>
<td>EICRE</td>
<td>O</td>
</tr>
<tr>
<td>CREATE FILE</td>
<td>S</td>
<td>30 14</td>
<td>EICRE</td>
<td>O</td>
</tr>
<tr>
<td>CREATE JOURNALMODEL</td>
<td>S</td>
<td>30 1E</td>
<td>EICRE</td>
<td>O</td>
</tr>
<tr>
<td>CREATE LSRPOOL</td>
<td>S</td>
<td>30 16</td>
<td>EICRE</td>
<td>O</td>
</tr>
<tr>
<td>CREATE MAPSET</td>
<td>S</td>
<td>30 04</td>
<td>EICRE</td>
<td>O</td>
</tr>
<tr>
<td>CREATE PARTITIONSET</td>
<td>S</td>
<td>30 06</td>
<td>EICRE</td>
<td>O</td>
</tr>
<tr>
<td>CREATE PARTNER</td>
<td>S</td>
<td>30 18</td>
<td>EICRE</td>
<td>O</td>
</tr>
<tr>
<td>CREATE PROFILE</td>
<td>S</td>
<td>30 0A</td>
<td>EICRE</td>
<td>O</td>
</tr>
<tr>
<td>CREATE PROGRAM</td>
<td>S</td>
<td>30 02</td>
<td>EICRE</td>
<td>O</td>
</tr>
<tr>
<td>CREATE SESSIONS</td>
<td>S</td>
<td>30 12</td>
<td>EICRE</td>
<td>O</td>
</tr>
<tr>
<td>CREATE TDQUEUE</td>
<td>S</td>
<td>30 1C</td>
<td>EICRE</td>
<td>O</td>
</tr>
<tr>
<td>CREATE TERMINAL</td>
<td>S</td>
<td>30 10</td>
<td>EICRE</td>
<td>O</td>
</tr>
<tr>
<td>CREATE TRANCLASS</td>
<td>S</td>
<td>30 1A</td>
<td>EICRE</td>
<td>O</td>
</tr>
<tr>
<td>CREATE TRANSACTION</td>
<td>S</td>
<td>30 08</td>
<td>EICRE</td>
<td>O</td>
</tr>
<tr>
<td>CREATE TYPETERM</td>
<td>S</td>
<td>30 0C</td>
<td>EICRE</td>
<td>O</td>
</tr>
<tr>
<td>DELAY</td>
<td>B</td>
<td>10 04</td>
<td>EIIC</td>
<td>O</td>
</tr>
</tbody>
</table>
## EXEC interface

Table 54. EXEC CICS commands ordered by command name (continued)

<table>
<thead>
<tr>
<th>Command</th>
<th>Class</th>
<th>Gp/fn code</th>
<th>Module DFH...</th>
<th>Lang</th>
</tr>
</thead>
<tbody>
<tr>
<td>DELETE</td>
<td>B</td>
<td>06 08</td>
<td>EIFC</td>
<td>O</td>
</tr>
<tr>
<td>DELETEQ TD</td>
<td>B</td>
<td>08 06</td>
<td>ETD</td>
<td>A</td>
</tr>
<tr>
<td>DELETEQ TS</td>
<td>B</td>
<td>0A 06</td>
<td>ETS</td>
<td>A</td>
</tr>
<tr>
<td>DEQ</td>
<td>B</td>
<td>12 06</td>
<td>EKC</td>
<td>A</td>
</tr>
<tr>
<td>DISCARD AUTINSTMODEL</td>
<td>S</td>
<td>42 10</td>
<td>EIQTM</td>
<td>O</td>
</tr>
<tr>
<td>DISCARD FILE</td>
<td>S</td>
<td>4C 10</td>
<td>EIQDS</td>
<td>O</td>
</tr>
<tr>
<td>DISCARD JOURNALMODEL</td>
<td>S</td>
<td>92 10</td>
<td>EIQSL</td>
<td>O</td>
</tr>
<tr>
<td>DISCARD JOURNALNAME</td>
<td>S</td>
<td>60 10</td>
<td>EIQSJ</td>
<td>O</td>
</tr>
<tr>
<td>DISCARD PARTNER</td>
<td>S</td>
<td>44 10</td>
<td>EIQPN</td>
<td>O</td>
</tr>
<tr>
<td>DISCARD PROFILE</td>
<td>S</td>
<td>46 10</td>
<td>EIQPF</td>
<td>O</td>
</tr>
<tr>
<td>DISCARD PROGRAM</td>
<td>S</td>
<td>4E 10</td>
<td>EIQSP</td>
<td>O</td>
</tr>
<tr>
<td>DISCARD TRANSACTION</td>
<td>S</td>
<td>50 10</td>
<td>EIQSX</td>
<td>O</td>
</tr>
<tr>
<td>DISABLE</td>
<td>B</td>
<td>22 04</td>
<td>UEM</td>
<td>A</td>
</tr>
<tr>
<td>DUMP</td>
<td>B</td>
<td>1C 02</td>
<td>EDC</td>
<td>A</td>
</tr>
<tr>
<td>DUMP SYSTEM</td>
<td>B</td>
<td>7E 04</td>
<td>EDCP</td>
<td>O</td>
</tr>
<tr>
<td>DUMP TRANSACTION</td>
<td>B</td>
<td>7E 02</td>
<td>EDCP</td>
<td>O</td>
</tr>
<tr>
<td>ENABLE</td>
<td>B</td>
<td>22 02</td>
<td>UEM</td>
<td>A</td>
</tr>
<tr>
<td>ENDBR</td>
<td>B</td>
<td>06 12</td>
<td>EIFC</td>
<td>O</td>
</tr>
<tr>
<td>ENQ</td>
<td>B</td>
<td>12 04</td>
<td>EKC</td>
<td>A</td>
</tr>
<tr>
<td>ENTER TRACEID</td>
<td>B</td>
<td>1A 04</td>
<td>ETR</td>
<td>A</td>
</tr>
<tr>
<td>ENTER TRACENUM</td>
<td>B</td>
<td>48 02</td>
<td>ETRX</td>
<td>O</td>
</tr>
<tr>
<td>EXTRACT ATTACH</td>
<td>B</td>
<td>04 28</td>
<td>ETC</td>
<td>A</td>
</tr>
<tr>
<td>EXTRACT ATTRIBUTES</td>
<td>B</td>
<td>04 3E</td>
<td>ETC</td>
<td>A</td>
</tr>
<tr>
<td>EXTRACT EXIT</td>
<td>B</td>
<td>22 06</td>
<td>UEM</td>
<td>A</td>
</tr>
<tr>
<td>EXTRACT LOGONMSG</td>
<td>B</td>
<td>04 3C</td>
<td>ETC</td>
<td>A</td>
</tr>
<tr>
<td>EXTRACT PROCESS</td>
<td>B</td>
<td>04 2E</td>
<td>ETC</td>
<td>A</td>
</tr>
<tr>
<td>EXTRACT TCT</td>
<td>B</td>
<td>04 2A</td>
<td>ETC</td>
<td>A</td>
</tr>
<tr>
<td>FORMATTIME</td>
<td>B</td>
<td>4A 04</td>
<td>EIDTI</td>
<td>O</td>
</tr>
<tr>
<td>FREE</td>
<td>B</td>
<td>04 22</td>
<td>ETC</td>
<td>A</td>
</tr>
<tr>
<td>FREEMAIN</td>
<td>B</td>
<td>0C 04</td>
<td>ESC</td>
<td>A</td>
</tr>
<tr>
<td>GDS ALLOCATE</td>
<td>B</td>
<td>24 02</td>
<td>EGL</td>
<td>A</td>
</tr>
<tr>
<td>GDS ASSIGN</td>
<td>B</td>
<td>24 04</td>
<td>EGL</td>
<td>A</td>
</tr>
<tr>
<td>GDS CONNECT PROCESS</td>
<td>B</td>
<td>24 0C</td>
<td>EGL</td>
<td>A</td>
</tr>
<tr>
<td>GDS EXTRACT ATTRIBUTES</td>
<td>B</td>
<td>24 1C</td>
<td>EGL</td>
<td>A</td>
</tr>
<tr>
<td>GDS EXTRACT PROCESS</td>
<td>B</td>
<td>24 06</td>
<td>EGL</td>
<td>A</td>
</tr>
<tr>
<td>GDS FREE</td>
<td>B</td>
<td>24 08</td>
<td>EGL</td>
<td>A</td>
</tr>
<tr>
<td>GDS ISSUE ABEND</td>
<td>B</td>
<td>24 0A</td>
<td>EGL</td>
<td>A</td>
</tr>
<tr>
<td>GDS ISSUE CONFIRMATION</td>
<td>B</td>
<td>24 0E</td>
<td>EGL</td>
<td>A</td>
</tr>
<tr>
<td>GDS ISSUE ERROR</td>
<td>B</td>
<td>24 10</td>
<td>EGL</td>
<td>A</td>
</tr>
<tr>
<td>GDS ISSUE PREPARE</td>
<td>B</td>
<td>24 1A</td>
<td>EGL</td>
<td>A</td>
</tr>
<tr>
<td>GDS ISSUE SIGNAL</td>
<td>B</td>
<td>24 12</td>
<td>EGL</td>
<td>A</td>
</tr>
<tr>
<td>GDS RECEIVE</td>
<td>B</td>
<td>24 14</td>
<td>EGL</td>
<td>A</td>
</tr>
<tr>
<td>GDS SEND</td>
<td>B</td>
<td>24 16</td>
<td>EGL</td>
<td>A</td>
</tr>
<tr>
<td>GDS WAIT</td>
<td>B</td>
<td>24 18</td>
<td>EGL</td>
<td>A</td>
</tr>
<tr>
<td>GETMAIN</td>
<td>B</td>
<td>0C 02</td>
<td>ESC</td>
<td>A</td>
</tr>
<tr>
<td>HANDLE ABEND</td>
<td>B</td>
<td>0E 0E</td>
<td>EPC</td>
<td>A</td>
</tr>
<tr>
<td>HANDLE AID</td>
<td>B</td>
<td>02 06</td>
<td>EEI</td>
<td>A</td>
</tr>
<tr>
<td>HANDLE CONDITION</td>
<td>B</td>
<td>02 04</td>
<td>EEI</td>
<td>A</td>
</tr>
<tr>
<td>IGNORE CONDITION</td>
<td>B</td>
<td>02 0A</td>
<td>EEI</td>
<td>A</td>
</tr>
<tr>
<td>INQUIRE AUTINSTMODEL</td>
<td>S</td>
<td>42 02</td>
<td>EIQTM</td>
<td>O</td>
</tr>
<tr>
<td>INQUIRE AUTOINSTALL</td>
<td>S</td>
<td>68 12</td>
<td>EIQVT</td>
<td>O</td>
</tr>
<tr>
<td>INQUIRE CONNECTION</td>
<td>S</td>
<td>58 02</td>
<td>EIQSC</td>
<td>O</td>
</tr>
<tr>
<td>Command</td>
<td>Class</td>
<td>Gp/fn code</td>
<td>Module</td>
<td>Lang</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-------</td>
<td>------------</td>
<td>--------</td>
<td>------</td>
</tr>
<tr>
<td>INQUIRE DCE</td>
<td>S</td>
<td>8E 02</td>
<td>EIQDE</td>
<td>O</td>
</tr>
<tr>
<td>INQUIRE DSNAME</td>
<td>S</td>
<td>7A 02</td>
<td>EIQDN</td>
<td>O</td>
</tr>
<tr>
<td>INQUIRE DUMPS</td>
<td>S</td>
<td>66 02</td>
<td>EIQDU</td>
<td>O</td>
</tr>
<tr>
<td>INQUIRE EXITPROGRAM</td>
<td>S</td>
<td>88 02</td>
<td>EIQUE</td>
<td>O</td>
</tr>
<tr>
<td>INQUIRE FILE</td>
<td>S</td>
<td>4C 02</td>
<td>EIQDS</td>
<td>O</td>
</tr>
<tr>
<td>INQUIRE IRC</td>
<td>S</td>
<td>6E 02</td>
<td>EIQIR</td>
<td>O</td>
</tr>
<tr>
<td>INQUIRE JOURNALMODEL</td>
<td>S</td>
<td>92 02</td>
<td>EIQLS</td>
<td>O</td>
</tr>
<tr>
<td>INQUIRE JOURNALNAME</td>
<td>S</td>
<td>60 12</td>
<td>EIQSJ</td>
<td>O</td>
</tr>
<tr>
<td>INQUIRE JOURNALNUM</td>
<td>S</td>
<td>60 02</td>
<td>EIQSJ</td>
<td>O</td>
</tr>
<tr>
<td>INQUIRE MODENAME</td>
<td>S</td>
<td>5A 02</td>
<td>EIQSM</td>
<td>O</td>
</tr>
<tr>
<td>INQUIRE MONITOR</td>
<td>S</td>
<td>70 12</td>
<td>EIQMS</td>
<td>O</td>
</tr>
<tr>
<td>INQUIRE NETNAME</td>
<td>S</td>
<td>52 06</td>
<td>EIQST</td>
<td>O</td>
</tr>
<tr>
<td>INQUIRE PARTNER</td>
<td>S</td>
<td>44 02</td>
<td>EIQPN</td>
<td>O</td>
</tr>
<tr>
<td>INQUIRE PROFILE</td>
<td>S</td>
<td>46 02</td>
<td>EIQPF</td>
<td>O</td>
</tr>
<tr>
<td>INQUIRE PROGRAM</td>
<td>S</td>
<td>4E 02</td>
<td>EIQSP</td>
<td>O</td>
</tr>
<tr>
<td>INQUIRE REQID</td>
<td>S</td>
<td>8A 02</td>
<td>EIRQR</td>
<td>O</td>
</tr>
<tr>
<td>INQUIRE STATISTICS</td>
<td>S</td>
<td>70 02</td>
<td>EIQMS</td>
<td>O</td>
</tr>
<tr>
<td>INQUIRE STREAMNAME</td>
<td>S</td>
<td>92 12</td>
<td>EIQLS</td>
<td>O</td>
</tr>
<tr>
<td>INQUIRE SYSDUMPCODE</td>
<td>S</td>
<td>66 22</td>
<td>EIQDU</td>
<td>O</td>
</tr>
<tr>
<td>INQUIRE SYSTEM</td>
<td>S</td>
<td>54 02</td>
<td>EIQSA</td>
<td>O</td>
</tr>
<tr>
<td>INQUIRE TASK</td>
<td>S</td>
<td>5E 02</td>
<td>EIQSK</td>
<td>O</td>
</tr>
<tr>
<td>INQUIRE TCLASS</td>
<td>S</td>
<td>5E 12</td>
<td>EIQSK</td>
<td>O</td>
</tr>
<tr>
<td>INQUIRE TDQUEUE</td>
<td>S</td>
<td>5C 02</td>
<td>EIQSQ</td>
<td>O</td>
</tr>
<tr>
<td>INQUIRE TERMINAL</td>
<td>S</td>
<td>52 02</td>
<td>EIQST</td>
<td>O</td>
</tr>
<tr>
<td>INQUIRE TRACEDEST</td>
<td>S</td>
<td>78 02</td>
<td>EIQTR</td>
<td>O</td>
</tr>
<tr>
<td>INQUIRE TRACEFLAG</td>
<td>S</td>
<td>78 12</td>
<td>EIQTR</td>
<td>O</td>
</tr>
<tr>
<td>INQUIRE TRACETYPE</td>
<td>S</td>
<td>78 22</td>
<td>EIQTR</td>
<td>O</td>
</tr>
<tr>
<td>INQUIRE TRANSDUMPCODE</td>
<td>S</td>
<td>66 12</td>
<td>EIQDU</td>
<td>O</td>
</tr>
<tr>
<td>INQUIRE TRANSACTION</td>
<td>S</td>
<td>50 02</td>
<td>EIQSX</td>
<td>O</td>
</tr>
<tr>
<td>INQUIRE TSQUEUE</td>
<td>S</td>
<td>0A 08</td>
<td>EIQTS</td>
<td>O</td>
</tr>
<tr>
<td>INQUIRE VTAM</td>
<td>S</td>
<td>68 02</td>
<td>EIQVT</td>
<td>O</td>
</tr>
<tr>
<td>ISSUE ABEND</td>
<td>B</td>
<td>04 30</td>
<td>ETC</td>
<td>A</td>
</tr>
<tr>
<td>ISSUE ABORT</td>
<td>B</td>
<td>1E 08</td>
<td>EDI</td>
<td>A</td>
</tr>
<tr>
<td>ISSUE ADD</td>
<td>B</td>
<td>1E 02</td>
<td>EDI</td>
<td>A</td>
</tr>
<tr>
<td>ISSUE CONFIRMATION</td>
<td>B</td>
<td>04 34</td>
<td>ETC</td>
<td>A</td>
</tr>
<tr>
<td>ISSUE COPY</td>
<td>B</td>
<td>04 0A</td>
<td>ETC</td>
<td>A</td>
</tr>
<tr>
<td>ISSUE DISCONNECT</td>
<td>B</td>
<td>04 14</td>
<td>ETC</td>
<td>A</td>
</tr>
<tr>
<td>ISSUE END</td>
<td>B</td>
<td>1E 0C</td>
<td>EDI</td>
<td>A</td>
</tr>
<tr>
<td>ISSUE ENDFILE</td>
<td>B</td>
<td>04 1A</td>
<td>ETC</td>
<td>A</td>
</tr>
<tr>
<td>ISSUE ENDOUTPUT</td>
<td>B</td>
<td>04 16</td>
<td>ETC</td>
<td>A</td>
</tr>
<tr>
<td>ISSUE EODS</td>
<td>B</td>
<td>04 08</td>
<td>ETC</td>
<td>A</td>
</tr>
<tr>
<td>ISSUE ERASE</td>
<td>B</td>
<td>1E 04</td>
<td>EDI</td>
<td>A</td>
</tr>
<tr>
<td>ISSUE ERASEAUP</td>
<td>B</td>
<td>04 18</td>
<td>ETC</td>
<td>A</td>
</tr>
<tr>
<td>ISSUE ERROR</td>
<td>B</td>
<td>04 36</td>
<td>ETC</td>
<td>A</td>
</tr>
<tr>
<td>ISSUE LOAD</td>
<td>B</td>
<td>04 0E</td>
<td>ETC</td>
<td>A</td>
</tr>
<tr>
<td>ISSUE NOTE</td>
<td>B</td>
<td>1E 10</td>
<td>EDI</td>
<td>A</td>
</tr>
<tr>
<td>ISSUE PASS</td>
<td>B</td>
<td>04 3A</td>
<td>ETC</td>
<td>A</td>
</tr>
<tr>
<td>ISSUE PREPARE</td>
<td>B</td>
<td>04 38</td>
<td>ETC</td>
<td>A</td>
</tr>
<tr>
<td>ISSUE PRINT</td>
<td>B</td>
<td>04 1C</td>
<td>ETC</td>
<td>A</td>
</tr>
<tr>
<td>ISSUE QUERY</td>
<td>B</td>
<td>1E 0A</td>
<td>EDI</td>
<td>A</td>
</tr>
<tr>
<td>ISSUE RECEIVE</td>
<td>B</td>
<td>1E 0E</td>
<td>EDI</td>
<td>A</td>
</tr>
<tr>
<td>ISSUE REPLACE</td>
<td>B</td>
<td>1E 06</td>
<td>EDI</td>
<td>A</td>
</tr>
<tr>
<td>Command</td>
<td>Class</td>
<td>Gp/fn code</td>
<td>Module</td>
<td>Lang</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-------</td>
<td>------------</td>
<td>--------</td>
<td>------</td>
</tr>
<tr>
<td>ISSUE RESET</td>
<td>B</td>
<td>04 12</td>
<td>ETC</td>
<td>A</td>
</tr>
<tr>
<td>ISSUE SEND</td>
<td>B</td>
<td>1E 14</td>
<td>EDI</td>
<td>A</td>
</tr>
<tr>
<td>ISSUE SIGNAL</td>
<td>B</td>
<td>04 1E</td>
<td>ETC</td>
<td>A</td>
</tr>
<tr>
<td>ISSUE WAIT</td>
<td>B</td>
<td>1E 12</td>
<td>EDI</td>
<td>A</td>
</tr>
<tr>
<td>LINK</td>
<td>B</td>
<td>0E 02</td>
<td>EPC</td>
<td>A</td>
</tr>
<tr>
<td>LOAD</td>
<td>B</td>
<td>0E 06</td>
<td>EPC</td>
<td>A</td>
</tr>
<tr>
<td>MONITOR</td>
<td>B</td>
<td>48 04</td>
<td>ETRX</td>
<td>O</td>
</tr>
<tr>
<td>PERFORM RESETTIME</td>
<td>S</td>
<td>72 02</td>
<td>EIPRT</td>
<td>O</td>
</tr>
<tr>
<td>PERFORM SECURITY</td>
<td>S</td>
<td>64 02</td>
<td>EIPSE</td>
<td>O</td>
</tr>
<tr>
<td>PERFORM SHUTDOWN</td>
<td>S</td>
<td>76 02</td>
<td>EIPSH</td>
<td>O</td>
</tr>
<tr>
<td>PERFORM STATISTICS</td>
<td>S</td>
<td>70 06</td>
<td>EIQMS</td>
<td>O</td>
</tr>
<tr>
<td>POINT</td>
<td>B</td>
<td>04 24</td>
<td>ETC</td>
<td>A</td>
</tr>
<tr>
<td>POP</td>
<td>B</td>
<td>02 0E</td>
<td>EEI</td>
<td>A</td>
</tr>
<tr>
<td>POST</td>
<td>B</td>
<td>10 06</td>
<td>EIC</td>
<td>O</td>
</tr>
<tr>
<td>PURGE MESSAGE</td>
<td>B</td>
<td>18 0A</td>
<td>EMS</td>
<td>A</td>
</tr>
<tr>
<td>PUSH</td>
<td>B</td>
<td>02 0C</td>
<td>EEI</td>
<td>A</td>
</tr>
<tr>
<td>QUERY SECURITY</td>
<td>B</td>
<td>6A 02</td>
<td>ESE</td>
<td>O</td>
</tr>
<tr>
<td>READ</td>
<td>B</td>
<td>06 02</td>
<td>EIFC</td>
<td>O</td>
</tr>
<tr>
<td>READNEXT</td>
<td>B</td>
<td>06 0E</td>
<td>EIFC</td>
<td>O</td>
</tr>
<tr>
<td>READPREV</td>
<td>B</td>
<td>06 10</td>
<td>EIFC</td>
<td>O</td>
</tr>
<tr>
<td>READQ TD</td>
<td>B</td>
<td>08 04</td>
<td>ETD</td>
<td>A</td>
</tr>
<tr>
<td>READQ TS</td>
<td>B</td>
<td>0A 04</td>
<td>ETS</td>
<td>A</td>
</tr>
<tr>
<td>RECEIVE</td>
<td>B</td>
<td>04 02</td>
<td>ETC</td>
<td>A</td>
</tr>
<tr>
<td>RECEIVE MAP</td>
<td>B</td>
<td>18 02</td>
<td>EMS</td>
<td>A</td>
</tr>
<tr>
<td>RECEIVE PARTN</td>
<td>B</td>
<td>18 0E</td>
<td>EMS</td>
<td>A</td>
</tr>
<tr>
<td>RELEASE</td>
<td>B</td>
<td>0E 0A</td>
<td>EPC</td>
<td>A</td>
</tr>
<tr>
<td>RESETBR</td>
<td>B</td>
<td>06 14</td>
<td>EIFC</td>
<td>O</td>
</tr>
<tr>
<td>RESYNC</td>
<td>B</td>
<td>16 04</td>
<td>ESP</td>
<td>A</td>
</tr>
<tr>
<td>RETRIEVE</td>
<td>B</td>
<td>10 0A</td>
<td>EIC</td>
<td>O</td>
</tr>
<tr>
<td>RETURN</td>
<td>B</td>
<td>0E 08</td>
<td>EPC</td>
<td>A</td>
</tr>
<tr>
<td>REWRITE</td>
<td>B</td>
<td>06 06</td>
<td>EIFC</td>
<td>O</td>
</tr>
<tr>
<td>ROUTE</td>
<td>B</td>
<td>18 0C</td>
<td>EMS</td>
<td>A</td>
</tr>
<tr>
<td>SEND</td>
<td>B</td>
<td>04 04</td>
<td>ETC</td>
<td>A</td>
</tr>
<tr>
<td>SEND CONTROL</td>
<td>B</td>
<td>18 12</td>
<td>EMS</td>
<td>A</td>
</tr>
<tr>
<td>SEND MAP</td>
<td>B</td>
<td>18 04</td>
<td>EMS</td>
<td>A</td>
</tr>
<tr>
<td>SEND PAGE</td>
<td>B</td>
<td>18 08</td>
<td>EMS</td>
<td>A</td>
</tr>
<tr>
<td>SEND PARTNSET</td>
<td>B</td>
<td>18 10</td>
<td>EMS</td>
<td>A</td>
</tr>
<tr>
<td>SEND TEXT</td>
<td>B</td>
<td>18 06</td>
<td>EMS</td>
<td>A</td>
</tr>
<tr>
<td>SET AUTOINSTALL</td>
<td>S</td>
<td>68 14</td>
<td>EIQVT</td>
<td>O</td>
</tr>
<tr>
<td>SET CONNECTION</td>
<td>S</td>
<td>58 04</td>
<td>EIQSC</td>
<td>O</td>
</tr>
<tr>
<td>SET DCE</td>
<td>S</td>
<td>8E 04</td>
<td>EIQDE</td>
<td>O</td>
</tr>
<tr>
<td>SET DSNNAME</td>
<td>S</td>
<td>7A 04</td>
<td>EIQDN</td>
<td>O</td>
</tr>
<tr>
<td>SET DUMPDS</td>
<td>S</td>
<td>66 04</td>
<td>EIQDU</td>
<td>O</td>
</tr>
<tr>
<td>SET FILE</td>
<td>S</td>
<td>4C 04</td>
<td>EIQDS</td>
<td>O</td>
</tr>
<tr>
<td>SET IRC</td>
<td>S</td>
<td>6E 04</td>
<td>EIQIR</td>
<td>O</td>
</tr>
<tr>
<td>SET JOURNALNAME</td>
<td>S</td>
<td>60 14</td>
<td>EIQSJ</td>
<td>O</td>
</tr>
<tr>
<td>SET JOURNALNUM</td>
<td>S</td>
<td>60 04</td>
<td>EIQSJ</td>
<td>O</td>
</tr>
<tr>
<td>SET MODENAME</td>
<td>S</td>
<td>5A 04</td>
<td>EIQSM</td>
<td>O</td>
</tr>
<tr>
<td>SET MONITOR</td>
<td>S</td>
<td>70 14</td>
<td>EIQMS</td>
<td>O</td>
</tr>
<tr>
<td>SET NETNAME</td>
<td>S</td>
<td>52 08</td>
<td>EIQST</td>
<td>O</td>
</tr>
<tr>
<td>SET PROGRAM</td>
<td>S</td>
<td>4E 04</td>
<td>EIQSP</td>
<td>O</td>
</tr>
<tr>
<td>SET STATISTICS</td>
<td>S</td>
<td>70 04</td>
<td>EIQMS</td>
<td>O</td>
</tr>
</tbody>
</table>
### Table 54. EXEC CICS commands ordered by command name (continued)

<table>
<thead>
<tr>
<th>Command</th>
<th>Class</th>
<th>Gp/fn code</th>
<th>Module</th>
<th>Lang</th>
</tr>
</thead>
<tbody>
<tr>
<td>SET SYSDUMPCODE</td>
<td>S</td>
<td>66 24</td>
<td>EIQDU</td>
<td>O</td>
</tr>
<tr>
<td>SET SYSTEM</td>
<td>S</td>
<td>54 04</td>
<td>EIQSA</td>
<td>O</td>
</tr>
<tr>
<td>SET TASK</td>
<td>S</td>
<td>5E 04</td>
<td>EIQSK</td>
<td>O</td>
</tr>
<tr>
<td>SET TCLASS</td>
<td>S</td>
<td>5E 14</td>
<td>EIQSK</td>
<td>O</td>
</tr>
<tr>
<td>SET TDQUEUE</td>
<td>S</td>
<td>5C 04</td>
<td>EIQSQ</td>
<td>O</td>
</tr>
<tr>
<td>SET TERMINAL</td>
<td>S</td>
<td>52 04</td>
<td>EIQST</td>
<td>O</td>
</tr>
<tr>
<td>SET TRACEDEST</td>
<td>S</td>
<td>78 04</td>
<td>EIQT</td>
<td>O</td>
</tr>
<tr>
<td>SET TRACEFLAG</td>
<td>S</td>
<td>78 14</td>
<td>EIQTR</td>
<td>O</td>
</tr>
<tr>
<td>SET TRACETYPE</td>
<td>S</td>
<td>78 24</td>
<td>EIQTR</td>
<td>O</td>
</tr>
<tr>
<td>SET TRANDUMPCODE</td>
<td>S</td>
<td>66 14</td>
<td>EIQDU</td>
<td>O</td>
</tr>
<tr>
<td>SET TRANSACTION</td>
<td>S</td>
<td>50 04</td>
<td>EIQSX</td>
<td>O</td>
</tr>
<tr>
<td>SET VTAM</td>
<td>S</td>
<td>68 04</td>
<td>EIQVT</td>
<td>O</td>
</tr>
<tr>
<td>SIGNOFF</td>
<td>B</td>
<td>74 04</td>
<td>ESN</td>
<td>O</td>
</tr>
<tr>
<td>SIGNON</td>
<td>B</td>
<td>74 02</td>
<td>ESN</td>
<td>O</td>
</tr>
<tr>
<td>SPOOLCLOSE</td>
<td>B</td>
<td>56 10</td>
<td>EPS</td>
<td>O</td>
</tr>
<tr>
<td>SPOOLOPEN</td>
<td>B</td>
<td>56 02</td>
<td>EPS</td>
<td>O</td>
</tr>
<tr>
<td>SPOOLREAD</td>
<td>B</td>
<td>56 04</td>
<td>EPS</td>
<td>O</td>
</tr>
<tr>
<td>SPOOLWRITE</td>
<td>B</td>
<td>56 06</td>
<td>EPS</td>
<td>O</td>
</tr>
<tr>
<td>START</td>
<td>B</td>
<td>10 08</td>
<td>EIC</td>
<td>O</td>
</tr>
<tr>
<td>STARTBR</td>
<td>B</td>
<td>06 0C</td>
<td>EIFC</td>
<td>O</td>
</tr>
<tr>
<td>SUSPEND</td>
<td>B</td>
<td>12 08</td>
<td>EKC</td>
<td>A</td>
</tr>
<tr>
<td>SYNCPOINT</td>
<td>B</td>
<td>16 02</td>
<td>ESP</td>
<td>A</td>
</tr>
<tr>
<td>TRACE</td>
<td>B</td>
<td>1A 02</td>
<td>ETR</td>
<td>A</td>
</tr>
<tr>
<td>UNLOCK</td>
<td>B</td>
<td>06 0A</td>
<td>EIFC</td>
<td>O</td>
</tr>
<tr>
<td>WAIT CONVID</td>
<td>B</td>
<td>04 2C</td>
<td>ETC</td>
<td>A</td>
</tr>
<tr>
<td>WAIT EVENT</td>
<td>B</td>
<td>12 02</td>
<td>EKC</td>
<td>A</td>
</tr>
<tr>
<td>WAIT EXTERNAL</td>
<td>B</td>
<td>5E 22</td>
<td>EIQSK</td>
<td>O</td>
</tr>
<tr>
<td>WAIT JOURNALNAME</td>
<td>B</td>
<td>14 08</td>
<td>EJC</td>
<td>A</td>
</tr>
<tr>
<td>WAIT JOURNALNUM</td>
<td>B</td>
<td>14 04</td>
<td>EJC</td>
<td>A</td>
</tr>
<tr>
<td>WAIT SIGNAL</td>
<td>B</td>
<td>04 10</td>
<td>ETC</td>
<td>A</td>
</tr>
<tr>
<td>WAIT TERMINAL</td>
<td>B</td>
<td>04 0C</td>
<td>ETC</td>
<td>A</td>
</tr>
<tr>
<td>WAITCICS</td>
<td>B</td>
<td>5E 32</td>
<td>EIQSK</td>
<td>O</td>
</tr>
<tr>
<td>WRITE FILE</td>
<td>B</td>
<td>06 04</td>
<td>EIFC</td>
<td>O</td>
</tr>
<tr>
<td>WRITE JOURNALNAME</td>
<td>B</td>
<td>14 06</td>
<td>EJC</td>
<td>A</td>
</tr>
<tr>
<td>WRITE JOURNALNUM</td>
<td>B</td>
<td>14 02</td>
<td>EJC</td>
<td>A</td>
</tr>
<tr>
<td>WRITE OPERATOR</td>
<td>B</td>
<td>6C 02</td>
<td>EOP</td>
<td>O</td>
</tr>
<tr>
<td>WRITEQ TD</td>
<td>B</td>
<td>08 02</td>
<td>ETID</td>
<td>A</td>
</tr>
<tr>
<td>WRITEQ TS</td>
<td>B</td>
<td>0A 02</td>
<td>ETS</td>
<td>A</td>
</tr>
<tr>
<td>XCTL</td>
<td>B</td>
<td>0E 04</td>
<td>EPC</td>
<td>A</td>
</tr>
</tbody>
</table>

**Abbreviations:**

- **Class of command:** B = basic, S = special
- **Language of module:** A = assembler, O = other

### Table 55. EXEC CICS commands ordered by group/function code

<table>
<thead>
<tr>
<th>Command</th>
<th>Class</th>
<th>Gp/fn code</th>
<th>Module</th>
<th>Lang</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADDRESS</td>
<td>B</td>
<td>02 02</td>
<td>EEI</td>
<td>A</td>
</tr>
<tr>
<td>HANDLE CONDITION</td>
<td>B</td>
<td>02 04</td>
<td>EEI</td>
<td>A</td>
</tr>
<tr>
<td>HANDLE AID</td>
<td>B</td>
<td>02 06</td>
<td>EEI</td>
<td>A</td>
</tr>
<tr>
<td>ASSIGN</td>
<td>B</td>
<td>02 08</td>
<td>EEI</td>
<td>A</td>
</tr>
</tbody>
</table>
### EXEC interface

#### Table 55. EXEC CICS commands ordered by group/function code (continued)

<table>
<thead>
<tr>
<th>Command</th>
<th>Class</th>
<th>Gp/fn code</th>
<th>Module DFH...</th>
<th>Lang</th>
</tr>
</thead>
<tbody>
<tr>
<td>IGNORE CONDITION</td>
<td>B</td>
<td>02 0A</td>
<td>EEI</td>
<td>A</td>
</tr>
<tr>
<td>PUSH</td>
<td>B</td>
<td>02 0C</td>
<td>EEI</td>
<td>A</td>
</tr>
<tr>
<td>POP</td>
<td>B</td>
<td>02 0E</td>
<td>EEI</td>
<td>A</td>
</tr>
<tr>
<td>ADDRESS SET</td>
<td>B</td>
<td>02 10</td>
<td>EEI</td>
<td>A</td>
</tr>
<tr>
<td>RECEIVE</td>
<td>B</td>
<td>04 02</td>
<td>ETC</td>
<td>A</td>
</tr>
<tr>
<td>SEND</td>
<td>B</td>
<td>04 04</td>
<td>ETC</td>
<td>A</td>
</tr>
<tr>
<td>CONVERSE</td>
<td>B</td>
<td>04 06</td>
<td>ETC</td>
<td>A</td>
</tr>
<tr>
<td>ISSUE EODS</td>
<td>B</td>
<td>04 08</td>
<td>ETC</td>
<td>A</td>
</tr>
<tr>
<td>ISSUE COPY</td>
<td>B</td>
<td>04 0A</td>
<td>ETC</td>
<td>A</td>
</tr>
<tr>
<td>WAIT TERMINAL</td>
<td>B</td>
<td>04 0C</td>
<td>ETC</td>
<td>A</td>
</tr>
<tr>
<td>ISSUE LOAD</td>
<td>B</td>
<td>04 0E</td>
<td>ETC</td>
<td>A</td>
</tr>
<tr>
<td>WAIT SIGNAL</td>
<td>B</td>
<td>04 10</td>
<td>ETC</td>
<td>A</td>
</tr>
<tr>
<td>ISSUE RESET</td>
<td>B</td>
<td>04 12</td>
<td>ETC</td>
<td>A</td>
</tr>
<tr>
<td>ISSUE DISCONNECT</td>
<td>B</td>
<td>04 14</td>
<td>ETC</td>
<td>A</td>
</tr>
<tr>
<td>ISSUE ENDOUTPUT</td>
<td>B</td>
<td>04 16</td>
<td>ETC</td>
<td>A</td>
</tr>
<tr>
<td>ISSUE ERASEAUP</td>
<td>B</td>
<td>04 18</td>
<td>ETC</td>
<td>A</td>
</tr>
<tr>
<td>ISSUE ENDFILE</td>
<td>B</td>
<td>04 1A</td>
<td>ETC</td>
<td>A</td>
</tr>
<tr>
<td>ISSUE PRINT</td>
<td>B</td>
<td>04 1C</td>
<td>ETC</td>
<td>A</td>
</tr>
<tr>
<td>ISSUE SIGNAL</td>
<td>B</td>
<td>04 1E</td>
<td>ETC</td>
<td>A</td>
</tr>
<tr>
<td>ALLOCATE</td>
<td>B</td>
<td>04 20</td>
<td>ETC</td>
<td>A</td>
</tr>
<tr>
<td>FREE</td>
<td>B</td>
<td>04 22</td>
<td>ETC</td>
<td>A</td>
</tr>
<tr>
<td>POINT</td>
<td>B</td>
<td>04 24</td>
<td>ETC</td>
<td>A</td>
</tr>
<tr>
<td>BUILD ATTACH</td>
<td>B</td>
<td>04 26</td>
<td>ETC</td>
<td>A</td>
</tr>
<tr>
<td>EXTRACT ATTACH</td>
<td>B</td>
<td>04 28</td>
<td>ETC</td>
<td>A</td>
</tr>
<tr>
<td>EXTRACT TCT</td>
<td>B</td>
<td>04 2A</td>
<td>ETC</td>
<td>A</td>
</tr>
<tr>
<td>WAIT CONVID</td>
<td>B</td>
<td>04 2C</td>
<td>ETC</td>
<td>A</td>
</tr>
<tr>
<td>EXTRACT PROCESS</td>
<td>B</td>
<td>04 2E</td>
<td>ETC</td>
<td>A</td>
</tr>
<tr>
<td>ISSUE ABEND</td>
<td>B</td>
<td>04 30</td>
<td>ETC</td>
<td>A</td>
</tr>
<tr>
<td>CONNECT PROCESS</td>
<td>B</td>
<td>04 32</td>
<td>ETC</td>
<td>A</td>
</tr>
<tr>
<td>ISSUE CONFIRMATION</td>
<td>B</td>
<td>04 34</td>
<td>ETC</td>
<td>A</td>
</tr>
<tr>
<td>ISSUE ERROR</td>
<td>B</td>
<td>04 36</td>
<td>ETC</td>
<td>A</td>
</tr>
<tr>
<td>ISSUE PREPARE</td>
<td>B</td>
<td>04 38</td>
<td>ETC</td>
<td>A</td>
</tr>
<tr>
<td>ISSUE PASS</td>
<td>B</td>
<td>04 3A</td>
<td>ETC</td>
<td>A</td>
</tr>
<tr>
<td>EXTRACT LOGONMSG</td>
<td>B</td>
<td>04 3C</td>
<td>ETC</td>
<td>A</td>
</tr>
<tr>
<td>EXTRACT ATTRIBUTES</td>
<td>B</td>
<td>04 3E</td>
<td>ETC</td>
<td>A</td>
</tr>
<tr>
<td>READ</td>
<td>B</td>
<td>06 02</td>
<td>EIFC</td>
<td>O</td>
</tr>
<tr>
<td>WRITE FILE</td>
<td>B</td>
<td>06 04</td>
<td>EIFC</td>
<td>O</td>
</tr>
<tr>
<td>REWRITE</td>
<td>B</td>
<td>06 06</td>
<td>EIFC</td>
<td>O</td>
</tr>
<tr>
<td>DELETE</td>
<td>B</td>
<td>06 08</td>
<td>EIFC</td>
<td>O</td>
</tr>
<tr>
<td>UNLOCK</td>
<td>B</td>
<td>06 0A</td>
<td>EIFC</td>
<td>O</td>
</tr>
<tr>
<td>STARTBR</td>
<td>B</td>
<td>06 0C</td>
<td>EIFC</td>
<td>O</td>
</tr>
<tr>
<td>READNEXT</td>
<td>B</td>
<td>06 0E</td>
<td>EIFC</td>
<td>O</td>
</tr>
<tr>
<td>READPREV</td>
<td>B</td>
<td>06 10</td>
<td>EIFC</td>
<td>O</td>
</tr>
<tr>
<td>ENDBR</td>
<td>B</td>
<td>06 12</td>
<td>EIFC</td>
<td>O</td>
</tr>
<tr>
<td>RESETBR</td>
<td>B</td>
<td>06 14</td>
<td>EIFC</td>
<td>O</td>
</tr>
<tr>
<td>WRITEQ TD</td>
<td>B</td>
<td>08 02</td>
<td>ETD</td>
<td>A</td>
</tr>
<tr>
<td>READQ TD</td>
<td>B</td>
<td>08 04</td>
<td>ETD</td>
<td>A</td>
</tr>
<tr>
<td>DELETETD</td>
<td>B</td>
<td>08 06</td>
<td>ETD</td>
<td>A</td>
</tr>
<tr>
<td>WRITEQ TS</td>
<td>B</td>
<td>0A 02</td>
<td>ETS</td>
<td>A</td>
</tr>
<tr>
<td>READQ TS</td>
<td>B</td>
<td>0A 04</td>
<td>ETS</td>
<td>A</td>
</tr>
<tr>
<td>DELETETQ TS</td>
<td>B</td>
<td>0A 06</td>
<td>ETS</td>
<td>A</td>
</tr>
<tr>
<td>INQUIRE TSQUEUE</td>
<td>S</td>
<td>0A 08</td>
<td>EIQTTS</td>
<td>O</td>
</tr>
</tbody>
</table>
Table 55. EXEC CICS commands ordered by group/function code  (continued)

<table>
<thead>
<tr>
<th>Command</th>
<th>Class</th>
<th>Gp/fn code</th>
<th>Module</th>
<th>Lang</th>
</tr>
</thead>
<tbody>
<tr>
<td>GETMAIN</td>
<td>B</td>
<td>0C 02</td>
<td>ESC</td>
<td>A</td>
</tr>
<tr>
<td>FREEMAIN</td>
<td>B</td>
<td>0C 04</td>
<td>ESC</td>
<td>A</td>
</tr>
<tr>
<td>LINK</td>
<td>B</td>
<td>0E 02</td>
<td>EPC</td>
<td>A</td>
</tr>
<tr>
<td>XCTL</td>
<td>B</td>
<td>0E 04</td>
<td>EPC</td>
<td>A</td>
</tr>
<tr>
<td>LOAD</td>
<td>B</td>
<td>0E 06</td>
<td>EPC</td>
<td>A</td>
</tr>
<tr>
<td>RETURN</td>
<td>B</td>
<td>0E 08</td>
<td>EPC</td>
<td>A</td>
</tr>
<tr>
<td>RELEASE</td>
<td>B</td>
<td>0E 0A</td>
<td>EPC</td>
<td>A</td>
</tr>
<tr>
<td>ABEND</td>
<td>B</td>
<td>0E 0C</td>
<td>EPC</td>
<td>A</td>
</tr>
<tr>
<td>HANDLE ABEND</td>
<td>B</td>
<td>0E 0E</td>
<td>EPC</td>
<td>A</td>
</tr>
<tr>
<td>ASKTIME</td>
<td>B</td>
<td>10 02</td>
<td>EIIC</td>
<td>O</td>
</tr>
<tr>
<td>DELAY</td>
<td>B</td>
<td>10 04</td>
<td>EIIC</td>
<td>O</td>
</tr>
<tr>
<td>POST</td>
<td>B</td>
<td>10 06</td>
<td>EIIC</td>
<td>O</td>
</tr>
<tr>
<td>START</td>
<td>B</td>
<td>10 08</td>
<td>EIIC</td>
<td>O</td>
</tr>
<tr>
<td>RETRIEVE</td>
<td>B</td>
<td>10 0A</td>
<td>EIIC</td>
<td>O</td>
</tr>
<tr>
<td>CANCEL</td>
<td>B</td>
<td>10 0C</td>
<td>EIIC</td>
<td>O</td>
</tr>
<tr>
<td>WAIT EVENT</td>
<td>B</td>
<td>12 02</td>
<td>EKC</td>
<td>A</td>
</tr>
<tr>
<td>ENQ</td>
<td>B</td>
<td>12 04</td>
<td>EKC</td>
<td>A</td>
</tr>
<tr>
<td>DEQ</td>
<td>B</td>
<td>12 06</td>
<td>EKC</td>
<td>A</td>
</tr>
<tr>
<td>SUSPEND</td>
<td>B</td>
<td>12 08</td>
<td>EKC</td>
<td>A</td>
</tr>
<tr>
<td>WRITE JOURNALNUM</td>
<td>B</td>
<td>14 02</td>
<td>EJC</td>
<td>A</td>
</tr>
<tr>
<td>WAIT JOURNALNUM</td>
<td>B</td>
<td>14 04</td>
<td>EJC</td>
<td>A</td>
</tr>
<tr>
<td>SYNCPOINT</td>
<td>B</td>
<td>16 02</td>
<td>ESP</td>
<td>A</td>
</tr>
<tr>
<td>RESYNC</td>
<td>B</td>
<td>16 04</td>
<td>ESP</td>
<td>A</td>
</tr>
<tr>
<td>RECEIVE MAP</td>
<td>B</td>
<td>18 02</td>
<td>EMS</td>
<td>A</td>
</tr>
<tr>
<td>SEND MAP</td>
<td>B</td>
<td>18 04</td>
<td>EMS</td>
<td>A</td>
</tr>
<tr>
<td>SEND TEXT</td>
<td>B</td>
<td>18 06</td>
<td>EMS</td>
<td>A</td>
</tr>
<tr>
<td>SEND PAGE</td>
<td>B</td>
<td>18 08</td>
<td>EMS</td>
<td>A</td>
</tr>
<tr>
<td>PURGE MESSAGE</td>
<td>B</td>
<td>18 0A</td>
<td>EMS</td>
<td>A</td>
</tr>
<tr>
<td>ROUTE</td>
<td>B</td>
<td>18 0C</td>
<td>EMS</td>
<td>A</td>
</tr>
<tr>
<td>RECEIVE PARTN</td>
<td>B</td>
<td>18 0E</td>
<td>EMS</td>
<td>A</td>
</tr>
<tr>
<td>SEND PARTNSET</td>
<td>B</td>
<td>18 10</td>
<td>EMS</td>
<td>A</td>
</tr>
<tr>
<td>SEND CONTROL</td>
<td>B</td>
<td>18 12</td>
<td>EMS</td>
<td>A</td>
</tr>
<tr>
<td>TRACE</td>
<td>B</td>
<td>1A 02</td>
<td>ETR</td>
<td>A</td>
</tr>
<tr>
<td>ENTER TRACEID</td>
<td>B</td>
<td>1A 04</td>
<td>ETR</td>
<td>A</td>
</tr>
<tr>
<td>DUMP</td>
<td>B</td>
<td>1C 02</td>
<td>EDC</td>
<td>A</td>
</tr>
<tr>
<td>ISSUE ADD</td>
<td>B</td>
<td>1E 02</td>
<td>EDI</td>
<td>A</td>
</tr>
<tr>
<td>ISSUE ERASE</td>
<td>B</td>
<td>1E 04</td>
<td>EDI</td>
<td>A</td>
</tr>
<tr>
<td>ISSUE REPLACE</td>
<td>B</td>
<td>1E 06</td>
<td>EDI</td>
<td>A</td>
</tr>
<tr>
<td>ISSUE ABORT</td>
<td>B</td>
<td>1E 08</td>
<td>EDI</td>
<td>A</td>
</tr>
<tr>
<td>ISSUE QUERY</td>
<td>B</td>
<td>1E 0A</td>
<td>EDI</td>
<td>A</td>
</tr>
<tr>
<td>ISSUE END</td>
<td>B</td>
<td>1E 0C</td>
<td>EDI</td>
<td>A</td>
</tr>
<tr>
<td>ISSUE RECEIVE</td>
<td>B</td>
<td>1E 0E</td>
<td>EDI</td>
<td>A</td>
</tr>
<tr>
<td>ISSUE NOTE</td>
<td>B</td>
<td>1E 10</td>
<td>EDI</td>
<td>A</td>
</tr>
<tr>
<td>ISSUE WAIT</td>
<td>B</td>
<td>1E 12</td>
<td>EDI</td>
<td>A</td>
</tr>
<tr>
<td>ISSUE SEND</td>
<td>B</td>
<td>1E 14</td>
<td>EDI</td>
<td>A</td>
</tr>
<tr>
<td>BIF DEEDIT</td>
<td>B</td>
<td>20 02</td>
<td>EBF</td>
<td>A</td>
</tr>
<tr>
<td>ENABLE</td>
<td>B</td>
<td>22 02</td>
<td>UEM</td>
<td>A</td>
</tr>
<tr>
<td>DISABLE</td>
<td>B</td>
<td>22 04</td>
<td>UEM</td>
<td>A</td>
</tr>
<tr>
<td>EXTRACT EXIT</td>
<td>B</td>
<td>22 06</td>
<td>UEM</td>
<td>A</td>
</tr>
<tr>
<td>GDS ALLOCATE</td>
<td>B</td>
<td>24 02</td>
<td>EGL</td>
<td>A</td>
</tr>
<tr>
<td>GDS ASSIGN</td>
<td>B</td>
<td>24 04</td>
<td>EGL</td>
<td>A</td>
</tr>
<tr>
<td>GDS EXTRACT PROCESS</td>
<td>B</td>
<td>24 06</td>
<td>EGL</td>
<td>A</td>
</tr>
</tbody>
</table>
### Table 55. EXEC CICS commands ordered by group/function code (continued)

<table>
<thead>
<tr>
<th>Command</th>
<th>Class</th>
<th>Gp/fn code</th>
<th>Module</th>
<th>Lang</th>
</tr>
</thead>
<tbody>
<tr>
<td>GDS FREE</td>
<td>B</td>
<td>24 08</td>
<td>EGL</td>
<td>A</td>
</tr>
<tr>
<td>GDS ISSUE ABEND</td>
<td>B</td>
<td>24 0A</td>
<td>EGL</td>
<td>A</td>
</tr>
<tr>
<td>GDS CONNECT PROCESS</td>
<td>B</td>
<td>24 0C</td>
<td>EGL</td>
<td>A</td>
</tr>
<tr>
<td>GDS ISSUE CONFIRMATION</td>
<td>B</td>
<td>24 0E</td>
<td>EGL</td>
<td>A</td>
</tr>
<tr>
<td>GDS ISSUE ERROR</td>
<td>B</td>
<td>24 10</td>
<td>EGL</td>
<td>A</td>
</tr>
<tr>
<td>GDS ISSUE SIGNAL</td>
<td>B</td>
<td>24 12</td>
<td>EGL</td>
<td>A</td>
</tr>
<tr>
<td>GDS RECEIVE</td>
<td>B</td>
<td>24 14</td>
<td>EGL</td>
<td>A</td>
</tr>
<tr>
<td>GDS SEND</td>
<td>B</td>
<td>24 16</td>
<td>EGL</td>
<td>A</td>
</tr>
<tr>
<td>GDS WAIT</td>
<td>B</td>
<td>24 18</td>
<td>EGL</td>
<td>A</td>
</tr>
<tr>
<td>GDS ISSUE PREPARE</td>
<td>B</td>
<td>24 1A</td>
<td>EGL</td>
<td>A</td>
</tr>
<tr>
<td>GDS EXTRACT ATTRIBUTES</td>
<td>B</td>
<td>24 1C</td>
<td>EGL</td>
<td>A</td>
</tr>
<tr>
<td>CREATE PROGRAM</td>
<td>S</td>
<td>30 02</td>
<td>EICRE</td>
<td>O</td>
</tr>
<tr>
<td>CREATE MAPSET</td>
<td>S</td>
<td>30 04</td>
<td>EICRE</td>
<td>O</td>
</tr>
<tr>
<td>CREATE PARTITIONSET</td>
<td>S</td>
<td>30 06</td>
<td>EICRE</td>
<td>O</td>
</tr>
<tr>
<td>CREATE TRANSACTION</td>
<td>S</td>
<td>30 08</td>
<td>EICRE</td>
<td>O</td>
</tr>
<tr>
<td>CREATE PROFILE</td>
<td>S</td>
<td>30 0A</td>
<td>EICRE</td>
<td>O</td>
</tr>
<tr>
<td>CREATE TYPETERM</td>
<td>S</td>
<td>30 0C</td>
<td>EICRE</td>
<td>O</td>
</tr>
<tr>
<td>CREATE CONNECTION</td>
<td>S</td>
<td>30 0E</td>
<td>EICRE</td>
<td>O</td>
</tr>
<tr>
<td>CREATE TERMINAL</td>
<td>S</td>
<td>30 10</td>
<td>EICRE</td>
<td>O</td>
</tr>
<tr>
<td>CREATE SESSIONS</td>
<td>S</td>
<td>30 12</td>
<td>EICRE</td>
<td>O</td>
</tr>
<tr>
<td>CREATE FILE</td>
<td>S</td>
<td>30 14</td>
<td>EICRE</td>
<td>O</td>
</tr>
<tr>
<td>CREATE LSRPOOL</td>
<td>S</td>
<td>30 16</td>
<td>EICRE</td>
<td>O</td>
</tr>
<tr>
<td>CREATE PARTNER</td>
<td>S</td>
<td>30 18</td>
<td>EICRE</td>
<td>O</td>
</tr>
<tr>
<td>CREATE TRANCLASS</td>
<td>S</td>
<td>30 1A</td>
<td>EICRE</td>
<td>O</td>
</tr>
<tr>
<td>CREATE TDQUEUE</td>
<td>S</td>
<td>30 1C</td>
<td>EICRE</td>
<td>O</td>
</tr>
<tr>
<td>CREATE JOURNALMODEL</td>
<td>S</td>
<td>30 1E</td>
<td>EICRE</td>
<td>O</td>
</tr>
<tr>
<td>INQUIRE AUTINSTMODEL</td>
<td>S</td>
<td>42 02</td>
<td>EIQTMO</td>
<td>O</td>
</tr>
<tr>
<td>DISCARD AUTINSTMODEL</td>
<td>S</td>
<td>42 10</td>
<td>EIQTMO</td>
<td>O</td>
</tr>
<tr>
<td>INQUIRE PARTNER</td>
<td>S</td>
<td>44 02</td>
<td>EIQPNO</td>
<td>O</td>
</tr>
<tr>
<td>DISCARD PARTNER</td>
<td>S</td>
<td>44 10</td>
<td>EIQPNO</td>
<td>O</td>
</tr>
<tr>
<td>INQUIRE PROFILE</td>
<td>S</td>
<td>46 02</td>
<td>EIQPFO</td>
<td>O</td>
</tr>
<tr>
<td>DISCARD PROFILE</td>
<td>S</td>
<td>46 10</td>
<td>EIQPFO</td>
<td>O</td>
</tr>
<tr>
<td>ENTER TRACENUM</td>
<td>B</td>
<td>48 02</td>
<td>ETRX</td>
<td>O</td>
</tr>
<tr>
<td>MONITOR</td>
<td>B</td>
<td>48 04</td>
<td>ETRX</td>
<td>O</td>
</tr>
<tr>
<td>ASKTIME ABSTIME</td>
<td>B</td>
<td>4A 02</td>
<td>EIDTI</td>
<td>O</td>
</tr>
<tr>
<td>FORMATTIME</td>
<td>B</td>
<td>4A 04</td>
<td>EIDTI</td>
<td>O</td>
</tr>
<tr>
<td>INQUIRE FILE</td>
<td>S</td>
<td>4C 02</td>
<td>EIQDS</td>
<td>O</td>
</tr>
<tr>
<td>SET FILE</td>
<td>S</td>
<td>4C 04</td>
<td>EIQDS</td>
<td>O</td>
</tr>
<tr>
<td>DISCARD FILE</td>
<td>S</td>
<td>4C 10</td>
<td>EIQDS</td>
<td>O</td>
</tr>
<tr>
<td>INQUIRE PROGRAM</td>
<td>S</td>
<td>4E 02</td>
<td>EIQSP</td>
<td>O</td>
</tr>
<tr>
<td>SET PROGRAM</td>
<td>S</td>
<td>4E 04</td>
<td>EIQSP</td>
<td>O</td>
</tr>
<tr>
<td>DISCARD PROGRAM</td>
<td>S</td>
<td>4E 10</td>
<td>EIQSP</td>
<td>O</td>
</tr>
<tr>
<td>INQUIRE TRANSACTION</td>
<td>S</td>
<td>50 02</td>
<td>EIQSX</td>
<td>O</td>
</tr>
<tr>
<td>SET TRANSACTION</td>
<td>S</td>
<td>50 04</td>
<td>EIQSX</td>
<td>O</td>
</tr>
<tr>
<td>DISCARD TRANSACTION</td>
<td>S</td>
<td>50 10</td>
<td>EIQSX</td>
<td>O</td>
</tr>
<tr>
<td>INQUIRE TERMINAL</td>
<td>S</td>
<td>52 02</td>
<td>EIQST</td>
<td>O</td>
</tr>
<tr>
<td>SET TERMINAL</td>
<td>S</td>
<td>52 04</td>
<td>EIQST</td>
<td>O</td>
</tr>
<tr>
<td>INQUIRE NETNAME</td>
<td>S</td>
<td>52 06</td>
<td>EIQST</td>
<td>O</td>
</tr>
<tr>
<td>SET NETNAME</td>
<td>S</td>
<td>52 08</td>
<td>EIQST</td>
<td>O</td>
</tr>
<tr>
<td>INQUIRE SYSTEM</td>
<td>S</td>
<td>54 02</td>
<td>EIQA</td>
<td>O</td>
</tr>
<tr>
<td>SET SYSTEM</td>
<td>S</td>
<td>54 04</td>
<td>EIQA</td>
<td>O</td>
</tr>
<tr>
<td>SPOOLOPEN</td>
<td>B</td>
<td>56 02</td>
<td>EPS</td>
<td>O</td>
</tr>
</tbody>
</table>
### EXEC interface

**Table 55. EXEC CICS commands ordered by group/function code (continued)**

<table>
<thead>
<tr>
<th>Command</th>
<th>Class</th>
<th>Gp/fn code</th>
<th>Module</th>
<th>Lang</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPOOLREAD</td>
<td>B</td>
<td>56 04</td>
<td>EPS</td>
<td>O</td>
</tr>
<tr>
<td>SPOOLWRITE</td>
<td>B</td>
<td>56 06</td>
<td>EPS</td>
<td>O</td>
</tr>
<tr>
<td>SPOOLCLOSE</td>
<td>B</td>
<td>56 10</td>
<td>EPS</td>
<td>O</td>
</tr>
<tr>
<td>INQUIRE CONNECTION</td>
<td>S</td>
<td>58 02</td>
<td>EIQSC</td>
<td>O</td>
</tr>
<tr>
<td>SET CONNECTION</td>
<td>S</td>
<td>58 04</td>
<td>EIQSC</td>
<td>O</td>
</tr>
<tr>
<td>INQUIRE MODENAME</td>
<td>S</td>
<td>5A 02</td>
<td>EIQSM</td>
<td>O</td>
</tr>
<tr>
<td>SET MODENAME</td>
<td>S</td>
<td>5A 04</td>
<td>EIQSM</td>
<td>O</td>
</tr>
<tr>
<td>INQUIRE TDQUEUE</td>
<td>S</td>
<td>5C 02</td>
<td>EIQSQ</td>
<td>O</td>
</tr>
<tr>
<td>SET TDQUEUE</td>
<td>S</td>
<td>5C 04</td>
<td>EIQSQ</td>
<td>O</td>
</tr>
<tr>
<td>INQUIRE TASK</td>
<td>S</td>
<td>5E 02</td>
<td>EIQSK</td>
<td>O</td>
</tr>
<tr>
<td>SET TASK</td>
<td>S</td>
<td>5E 04</td>
<td>EIQSK</td>
<td>O</td>
</tr>
<tr>
<td>CHANGE TASK</td>
<td>B</td>
<td>5E 06</td>
<td>EIQSK</td>
<td>O</td>
</tr>
<tr>
<td>INQUIRE TCLASS</td>
<td>S</td>
<td>5E 12</td>
<td>EIQSK</td>
<td>O</td>
</tr>
<tr>
<td>SET TCCLASS</td>
<td>S</td>
<td>5E 14</td>
<td>EIQSK</td>
<td>O</td>
</tr>
<tr>
<td>WAIT EXTERNAL</td>
<td>B</td>
<td>5E 22</td>
<td>EIQSK</td>
<td>O</td>
</tr>
<tr>
<td>WAITCICS</td>
<td>B</td>
<td>5E 32</td>
<td>EIQSK</td>
<td>O</td>
</tr>
<tr>
<td>INQUIRE JOURNALNUM</td>
<td>S</td>
<td>60 02</td>
<td>EIQSJ</td>
<td>O</td>
</tr>
<tr>
<td>SET JOURNALNUM</td>
<td>S</td>
<td>60 04</td>
<td>EIQSJ</td>
<td>O</td>
</tr>
<tr>
<td>INQUIRE JOURNALNAME</td>
<td>S</td>
<td>60 12</td>
<td>EIQSJ</td>
<td>O</td>
</tr>
<tr>
<td>SET JOURNALNAME</td>
<td>S</td>
<td>60 14</td>
<td>EIQSJ</td>
<td>O</td>
</tr>
<tr>
<td>PERFORM SECURITY</td>
<td>S</td>
<td>64 02</td>
<td>EIPSE</td>
<td>O</td>
</tr>
<tr>
<td>INQUIRE DUMPDS</td>
<td>S</td>
<td>66 02</td>
<td>EIQDU</td>
<td>O</td>
</tr>
<tr>
<td>SET DUMPDS</td>
<td>S</td>
<td>66 04</td>
<td>EIQDU</td>
<td>O</td>
</tr>
<tr>
<td>INQUIRE TRANDUMPCODE</td>
<td>S</td>
<td>66 12</td>
<td>EIQDU</td>
<td>O</td>
</tr>
<tr>
<td>SET TRANDUMPCODE</td>
<td>S</td>
<td>66 14</td>
<td>EIQDU</td>
<td>O</td>
</tr>
<tr>
<td>INQUIRE SYSDUMPCODE</td>
<td>S</td>
<td>66 22</td>
<td>EIQDU</td>
<td>O</td>
</tr>
<tr>
<td>SET SYSDUMPCODE</td>
<td>S</td>
<td>66 24</td>
<td>EIQDU</td>
<td>O</td>
</tr>
<tr>
<td>INQUIRE VTAM</td>
<td>S</td>
<td>68 02</td>
<td>EIQVT</td>
<td>O</td>
</tr>
<tr>
<td>SET VTAM</td>
<td>S</td>
<td>68 04</td>
<td>EIQVT</td>
<td>O</td>
</tr>
<tr>
<td>INQUIRE AUTOINSTALL</td>
<td>S</td>
<td>68 12</td>
<td>EIQVT</td>
<td>O</td>
</tr>
<tr>
<td>SET AUTOINSTALL</td>
<td>S</td>
<td>68 14</td>
<td>EIQVT</td>
<td>O</td>
</tr>
<tr>
<td>QUERY SECURITY</td>
<td>B</td>
<td>6A 02</td>
<td>ESE</td>
<td>O</td>
</tr>
<tr>
<td>WRITE OPERATOR</td>
<td>B</td>
<td>6C 02</td>
<td>EOP</td>
<td>O</td>
</tr>
<tr>
<td>CICSMESSAGE *</td>
<td>S</td>
<td>6C 12</td>
<td>EOP</td>
<td>O</td>
</tr>
<tr>
<td>INQUIRE IRC</td>
<td>S</td>
<td>6E 02</td>
<td>EIQIR</td>
<td>O</td>
</tr>
<tr>
<td>SET IRC</td>
<td>S</td>
<td>6E 04</td>
<td>EIQIR</td>
<td>O</td>
</tr>
<tr>
<td>INQUIRE STATISTICS</td>
<td>S</td>
<td>70 02</td>
<td>EIQMS</td>
<td>O</td>
</tr>
<tr>
<td>SET STATISTICS</td>
<td>S</td>
<td>70 04</td>
<td>EIQMS</td>
<td>O</td>
</tr>
<tr>
<td>PERFORM STATISTICS</td>
<td>S</td>
<td>70 06</td>
<td>EIQMS</td>
<td>O</td>
</tr>
<tr>
<td>COLLECT STATISTICS</td>
<td>S</td>
<td>70 08</td>
<td>EIQMS</td>
<td>O</td>
</tr>
<tr>
<td>INQUIRE MONITOR</td>
<td>S</td>
<td>70 12</td>
<td>EIQMS</td>
<td>O</td>
</tr>
<tr>
<td>SET MONITOR</td>
<td>S</td>
<td>70 14</td>
<td>EIQMS</td>
<td>O</td>
</tr>
<tr>
<td>PERFORM RESETTIME</td>
<td>S</td>
<td>72 02</td>
<td>EIPRT</td>
<td>O</td>
</tr>
<tr>
<td>SIGNON</td>
<td>B</td>
<td>74 02</td>
<td>ESN</td>
<td>O</td>
</tr>
<tr>
<td>SIGNOFF</td>
<td>B</td>
<td>74 04</td>
<td>ESN</td>
<td>O</td>
</tr>
<tr>
<td>PERFORM SHUTDOWN</td>
<td>S</td>
<td>76 02</td>
<td>EIPSH</td>
<td>O</td>
</tr>
<tr>
<td>INQUIRE TRACEDEST</td>
<td>S</td>
<td>78 02</td>
<td>EIQTR</td>
<td>O</td>
</tr>
<tr>
<td>SET TRACEDEST</td>
<td>S</td>
<td>78 04</td>
<td>EIQTR</td>
<td>O</td>
</tr>
<tr>
<td>INQUIRE TRACEFLAG</td>
<td>S</td>
<td>78 12</td>
<td>EIQTR</td>
<td>O</td>
</tr>
<tr>
<td>SET TRACEFLAG</td>
<td>S</td>
<td>78 14</td>
<td>EIQTR</td>
<td>O</td>
</tr>
<tr>
<td>INQUIRE TRACETYPE</td>
<td>S</td>
<td>78 22</td>
<td>EIQTR</td>
<td>O</td>
</tr>
<tr>
<td>SET TRACETYPE</td>
<td>S</td>
<td>78 24</td>
<td>EIQTR</td>
<td>O</td>
</tr>
</tbody>
</table>
### Table 55. EXEC CICS commands ordered by group/function code (continued)

<table>
<thead>
<tr>
<th>Command</th>
<th>Class</th>
<th>Gp/fn code</th>
<th>Module</th>
<th>Lang</th>
</tr>
</thead>
<tbody>
<tr>
<td>INQUIRE DSNAME</td>
<td>S</td>
<td>7A 02</td>
<td>EIQDN O</td>
<td>O</td>
</tr>
<tr>
<td>SET DSNAME</td>
<td>S</td>
<td>7A 04</td>
<td>EIQDN O</td>
<td>O</td>
</tr>
<tr>
<td>DUMP TRANSACTION</td>
<td>B</td>
<td>7E 02</td>
<td>EDCP O</td>
<td>O</td>
</tr>
<tr>
<td>DUMP SYSTEM</td>
<td>B</td>
<td>7E 04</td>
<td>EDCP O</td>
<td>O</td>
</tr>
<tr>
<td>INQUIRE JOURNALMODEL</td>
<td>S</td>
<td>92 02</td>
<td>EIQSL O</td>
<td>O</td>
</tr>
<tr>
<td>INQUIRE STREAMNAME</td>
<td>S</td>
<td>92 12</td>
<td>EIQSL O</td>
<td>O</td>
</tr>
</tbody>
</table>

**Abbreviations:**

- **Class of command:** B = basic, S = special
- **Language of module:** A = assembler, O = other

---

### DFHEIP

The EXEC interface program, DFHEIP, has several entry points associated with initialization and termination. Note, however, that DFHEIPAN is in the DFHEIPA module.

**Entry point**

**Function**

**DFHEIPNA**

Formal main entry point

**DFHEIPAN**

Get or free dynamic storage for assembler-language prologue or epilogue

**DFHEIPGM**

Get dynamic storage for COBOL initialization

**DFHEIPFM**

Free dynamic storage for COBOL

**DFHEIPTT**

Take run-unit token routine for COBOL initialization.

DFHEIP has these entry points associated with executing a command issued from an application program:

**Entry point**

**Function**

**DFHEIPRN**

EXEC RMI calls

**DFHEIPCN**

EXEC CICS calls

**DFHEIPDN**

xxxTDLI calls.

It also has many return and entry points for common functions that are called from those processor modules residing in the nucleus:

**Entry point**

**Function**

**EIPNORML**

Normal return on completion of command
EXEC interface

Error point
Function
EIPERROR
  Condition occurred (code in EIBRCODE)
EIPCONDN
  Condition occurred (code in EIBRESP)
EICCER99
  Unsupported function, abend AEY9
EICCDF00
  Subroutine to invoke EDF

Several length-checking routines (EICCLCnn):
Error point
Function
EICCLC30
  Input check, V format only
EICCLC94
  LENGERR flag check

Several program control routines (EICCPCnn):
Error point
Function
EICCPC00
  Process terminating PL/I program
EICCPC40
  HANDLE ABEND processing

Several storage control routines (EICCSCnn):
Error point
Function
EICCSC10
  FREEMAIN
EICCSC20
  GETMAIN shared storage
EICCSC30
  GETMAIN terminal storage
EICCSC70
  GETMAIN user storage init. X’00’
EICCFM10
  FREEMAIN for COMMAREA

Method of calling processor modules

All processor modules reside in the CICS nucleus, and the same calling method is used regardless of the language in which the processor is coded.

CICS initialization puts the address of each module in the CSA optional features list (CSAOPFL), in a table of addresses starting at CSAEXECS, and at an offset corresponding to its group code.

The calling method for the processor modules at execution time uses a table (at label EICC71T in DFHEIP), known as the EXEC command processor module call table. DFHEIP uses this table, and the table of addresses in CSAOPFL.
The EXEC command processor module call table is indexed by the 1-byte group code, which identifies the way that the processor is called:

**Call type**

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
</tr>
<tr>
<td>Has a vector of offsets at its entry point. This vector is indexed by the command function code to locate the actual entry point, to which DFHEIP does an unconditional branch. Return is to label EIPNORML, EIPCONDN, or EIPERROR.</td>
</tr>
<tr>
<td>B</td>
</tr>
<tr>
<td>Has a single entry point, for which DFHEIP issues a DFHAM TYPE=LINK call. The appropriate return address in DFHEIP is set in register 14, an unconditional branch is made to the DFHEIP, which tests the response in EIBRESP.</td>
</tr>
<tr>
<td>C</td>
</tr>
<tr>
<td>Has a single entry point, for which DFHEIP issues a DFHEIEIM call (through the kernel). Return is to the next instruction, where DFHEIP tests the response in EIBRESP.</td>
</tr>
<tr>
<td>D</td>
</tr>
<tr>
<td>Has a single entry point, for which DFHEIP uses a BALR R14,R15 instruction; this type is used only for GDS. The appropriate return address in DFHEIP is set in register 14, an unconditional branch is made to the DFHEIP, the response in the user’s RETCODE field.</td>
</tr>
</tbody>
</table>

**Exits**

The following global user exit points are provided in DFHEIP:

- XEIN
- XEOUT
- XEISPIN
- XEISPOUT

For further information, see the CICS Customization Guide.

**Trace**

The following point ID is provided for DFHEIP:

- AP 00E1, for which the trace level is EI 1.

The following point IDs are provided for DFHEISR:

- AP E110 (entry), for which the trace level is EI 2.
- AP E111 (exit), for which the trace level is EI 2.

Trace entries are made before and after the execution of a command by its EXEC interface processor module.
EXEC interface

For more information about the trace points, see the CICS Trace Entries. For more information about using traces in problem determination, see the CICS Problem Determination Guide.
Chapter 36. Execution diagnostic facility (EDF)

The execution diagnostic facility (EDF) allows users of the CICS command-level programming interface to step through the CICS commands of an application program. This program can be part of a local or remote transaction. At each step, the user can check the validity of each command and make temporary modifications to the program.

Design overview

EDF enables an application programmer to test a command-level application program online without making any modifications to the source program or the program preparation procedure. EDF intercepts execution of the application program at certain points and displays relevant information about the program at these points.

There are seven places in the EXEC interface program (DFHEIP) where the EDF can be called:
1. When program initialization has been done, just before control is passed to the application entry point
2. When program termination is being done, just after control has been received from the application
3. Before a normal EXEC command is passed to its processor module
4. When a normal EXEC command has returned to DFHEIP
5. Before an EXEC CICS GDS command is passed to its processor module
6. When an EXEC CICS GDS command has returned to DFHEIP
7. Before an EXEC CICS FEPI command is passed to its processor module
8. When an EXEC CICS FEPI command has returned to DFHEIP
9. At the end of a PL/I program.

Modules

CEBR transaction (DFHEDFBR)
The temporary-storage browse transaction (CEBR) allows the user to browse, copy, or delete items in a queue. CEBR invokes DFHEDFBR to execute the required action.

EDF display (DFHEDFD)
The EDF display program, DFHEDFD, provides the following functions:
• To display the user program status
• To allow the user to modify argument values and responses
• To allow the user to display and modify the EXEC interface block (EIB) and program working storage
• To allow the user to display any hexadecimal location in the partition user screen
• To allow the user to suppress EDF displays until specified conditions are met.
Execution diagnostic facility (EDF)

Method
1. Data describing user status is passed to DFHEDFD in the TWA.
2. Initialize exception and abend handling.
3. If TS queue for user terminal already exists, read control information; otherwise create control information about TS queue.
4. Check for security violation.
5. If necessary, remember user screen.
6. Build required display by calling DFHEDFS.
7. Send display to EDF screen.
8. Extract modified information by calling DFHEDFS.
9. Analyze request.
10. Set up build information for next display.
11. Go and build required display.
12. When no further displays are required:
   a. Save function display
   b. If necessary, restore user screen
   c. Update control information
   d. If transaction is defined as remote, purge TS queue and any shared storage associated with the EDF task
   e. Return to DFHEDFP.

EDF map set (DFHEDFM)
The EDF map set, DFHEDFM, consists of three maps:

DFHEDFM
To display status information at the various EDF interception points

DFHEDFN
To display the EDF stop conditions

DFHEDFP
To display a dump of storage.

All maps are (24,80). The first two lines of each map contain the transaction ID, program name, status, and so on. The format of these two lines must be identical for all maps. A menu is displayed with each map, and includes a message line and a reply field. The format of the menu must be identical for all maps. The cursor is positioned by symbolic cursor positioning.

EDF control program (DFHEDFP)
The EDF control program, DFHEDFP, provides the CEDF transaction for starting EDF, and is used in two different ways:
1. To control the debugging task
2. To set debug mode on or off.

Input
Input to the DFHEDFP program is provided as follows:

To control the debugging task
Information describing the user task status is written into the debug linkage area (DLA) of CEDF by DFHEDFX.
To set debug mode on or off

The user enters a CEDF request at the debug display terminal using the following syntax:

CEDF termid,ON|OFF

Alternatively, a PF key may be used to switch single-terminal debug mode on.

Note: To use EDF for a remote transaction, only single-terminal mode is available.

Output
Output from the DFHEDFP program is as follows:

To control the debugging task
DFHEDFD displays user program status.

To set debug mode on or off
Switches the debug mode bit either in the user terminal TCTTE or, if an EXEC task is running, in the user task EIS. For two-terminal debugging, creates temporary-storage queue element to connect user terminal with display terminal.

Method

To control the program for debugging a task
If the task is attached by DFHEDFX and if only one terminal is being used for debugging, link to DFHEDFD to display program status. If two terminals are being used for debugging, start CEDF at the display terminal, restore that terminal to the user, resume the user task, then return to CICS.

To set debug mode on or off
If invoked by using a PF key, set the debug mode on for single-terminal debugging in the user TCTTE. If invoked by a CEDF request, extract the user terminal ID (default is display terminal), and extract the debug mode (default is on). If the user terminal ID does not exist, output a diagnostic message. If the EXEC task is running and the task is in debug mode, output a diagnostic message; otherwise switch the debug bit in EIS, or switch the debug bit in TCTTE. Create a temporary-storage queue element naming the debug terminal.

EDF response table (DFHEDFR)
The EDF response table, DFHEDFR, is a table used by DFHEDFD to interpret the responses obtained by EXEC commands.

EDF task switch program (DFHEDFX)
The EDF task switch program, DFHEDFX, is used to attach the debugging task, provide it with all necessary information about the status of the user task, and suspend the user task until the debugging task allows it to resume.

Method

1. Extract information describing the user task status and copy it into the DLA for the attached task
2. Issue wait on user terminal
3. Attach CEDF
4. Suspend the user task
Execution diagnostic facility (EDF)

5. When the user task is resumed by EDF, check if EDF has not abended
6. If the user requests an abend, abend the user task; otherwise, return to caller.

Exits

No global user exit points are provided for this function.

Trace

No trace points are provided for this function.
Chapter 37. Extended recovery facility (XRF)

The extended recovery facility (XRF) enables you to achieve a high level of availability. You can run an alternate CICS system that monitors your active CICS system, and takes over automatically or by operator control if the active system fails. You can also plan and execute a takeover yourself when you want to do maintenance on an active system.

Problems in the active system can be detected and isolated as soon as they occur. The alternate system can recover and restart quickly, like an emergency restart, and the time for reconnection of terminals is reduced.

Design overview

A detailed overview of this function is given in the CICS/ESA 3.3 XRF Guide.

Control blocks

A command list table (CLT) is used by an alternate system when it takes over the running of CICS from an active system. It holds the ID data for the JES system in use, data used to verify its authority to take over, and routing information. If there is more than one active system in two CECs, the CLT also holds VTAM MODIFY commands, and messages to the operator (WTO) to complete the takeover. It is loaded during takeover, and deleted when processed.

See the CICS Data Areas manual for a detailed description of this control block.

Modules

Figure 57 on page 533 shows the modules for XRF.
Extended recovery facility

There is one global user exit point in DFHXRA: XXRSTAT. For further information about this, see the CICS Customization Guide.

Trace

The following point IDs are provided for the CAVM services:

- AP 00C4, AP 00C5, AP 00C6, and AP 00C7, for which the trace level is AP 1.

The following point IDs are provided for the XRF takeover signon/sign-off function:

- AP 0Axx, for which the trace levels are AP 1, AP 2, and Exc.

For more information about the trace points, see the CICS Trace Entries. For more information about using traces in problem determination, see the CICS Problem Determination Guide.
Chapter 38. External CICS interface

The external CICS interface (EXCI) is an integral part of CICS Transaction Server for z/OS, Version 2 Release 1. The function is called an external CICS interface because it enables non-CICS application programs (client programs) running in MVS to call programs (server programs) running in a CICS Transaction Server for z/OS, Version 2 Release 1 region and to pass and receive data by means of a communications area.

Design overview

This section provides an overview of the design of the external CICS interface. For more information about the external CICS interface, see the CICS External Interfaces Guide.

The external CICS interface is an application programming interface that enables a non-CICS program (a client program) running in MVS to call a program (a server program) running in a CICS Transaction Server for z/OS, Version 2 Release 1 region and to pass and receive data by means of a communications area. The CICS application program is invoked as if linked-to by another CICS application program.

This programming interface allows a user to allocate and open sessions (or pipes\(^1\)) to a CICS region, and to pass distributed program link (DPL) requests over them. The multiregion operation (MRO) facility of CICS interregion communication (IRC) facility supports these requests, and each pipe maps onto one MRO session.

Unless the CICS region is running in a sysplex under MVS/ESA 5.1 and therefore able to use cross-system MRO (XCF/MRO), the client program and the CICS server region (the region where the server program runs or is defined) must be in the same MVS image. Although the external CICS interface does not support the cross-memory access method, it can use the XCF access method provided by XCF/MRO in CICS Transaction Server for z/OS, Version 2 Release 1. See the CICS Intercommunication Guide for information about XCF/MRO.

A client program that uses the external CICS interface can operate multiple sessions for different users (either under the same or separate TCBs) all coexisting in the same MVS address space without knowledge of, or interference from, each other.

Where a client program attaches another client program, the attached program runs under its own TCB.

The programming interfaces

The external CICS interface provides two forms of programming interface: the EXCI CALL interface and the EXEC CICS interface.

The EXCI CALL interface

This interface consists of six commands that allow you to:

---

1. pipe. A one-way communication path between a sending process and a receiving process. In an external CICS interface implementation, each pipe maps onto one MRO session, where the client program represents the sending process and the CICS server region represents the receiving process.
Allocate and open sessions to a CICS system from non-CICS programs running under MVS

Issue DPL requests on these sessions from the non-CICS programs

Close and deallocate the sessions on completion of the DPL requests.

The six EXCI commands are:
1. Initialize_User
2. Allocate_Pipe
3. Open_Pipe
4. DPL call
5. Close_Pipe
6. Deallocate_Pipe

The processing of an EXCI CALL-level command is shown in Figure 58 on page 535.

The EXEC CICS interface

The external CICS interface provides a single, composite command—EXEC CICS LINK PROGRAM—that performs all six commands of the EXCI CALL interface in one invocation. The processing of an EXEC CICS LINK command is shown in Figure 59 on page 535.

This command takes the same form as the distributed program link command of the CICS command-level application programming interface.

API restrictions for server programs

A CICS server program invoked by an external CICS interface request is restricted to the DPL subset of the CICS application programming interface. This subset (the DPL subset) of the API commands is the same as for a CICS-to-CICS server program.

For details about the DPL subset for server programs, see the CICS Application Programming Guide.
### Modules

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHXCALL</td>
<td>EXEC-level API macro. Invoked by the CICS translator when processing EXCI EXEC-level requests.</td>
</tr>
<tr>
<td>DFHXCDMP</td>
<td>dump services. Calls the CICS SVC to issue SDUMP macro requests, to take an SDUMP of the EXCI address space.</td>
</tr>
<tr>
<td>DFHXCSTB</td>
<td>stub link-edited with applications that want to use EXCI.</td>
</tr>
<tr>
<td>DFHXCEIP</td>
<td>EXEC-level API handler. The main EXCI module that processes EXCI EXEC-level requests.</td>
</tr>
<tr>
<td>DFHXCO</td>
<td>options macro for generating the DFHXCLOPT options table.</td>
</tr>
</tbody>
</table>
External CICS interface

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHXCLOPT</td>
<td>options table to customize the EXCI environment.</td>
</tr>
<tr>
<td>DFHXCPLD</td>
<td>Assembler-language parameter list definitions. Copybook defining the parameters for use with the EXCI APIs.</td>
</tr>
<tr>
<td>DFHXCPLH</td>
<td>C parameter list definitions. Copybook defining the parameters for use with the EXCI APIs.</td>
</tr>
<tr>
<td>DFHXCPLL</td>
<td>PL/I parameter list definitions. Copybook defining the parameters for use with the EXCI APIs.</td>
</tr>
<tr>
<td>DFHXCPLO</td>
<td>COBOL parameter list definitions. Copybook defining the parameters for use with the EXCI APIs.</td>
</tr>
<tr>
<td>DFHXCPRH</td>
<td>program request handler The main EXCI module that processes EXCI CALL-level requests.</td>
</tr>
<tr>
<td>DFHXCRCD</td>
<td>Assembler-language return code definitions. Copybook defining the return codes for use with the EXCI APIs.</td>
</tr>
<tr>
<td>DFHXCRCH</td>
<td>C return code definitions. Copybook defining the return codes for use with the EXCI APIs.</td>
</tr>
<tr>
<td>DFHXCRCL</td>
<td>PL/I return code definitions. Copybook defining the return codes for use with the EXCI APIs.</td>
</tr>
<tr>
<td>DFHXCRCO</td>
<td>COBOL return code definitions. Copybook defining the return codes for use with the EXCI APIs.</td>
</tr>
<tr>
<td>DFHXCSC</td>
<td>SVC services. Invoked by the CICS SVC to issue an SDUMP macro to take an SDUMP of the EXCI address space.</td>
</tr>
<tr>
<td>DFHXCTAB</td>
<td>language table. Copybook defining the syntax of the EXCI EXEC language for use by the CICS translator.</td>
</tr>
<tr>
<td>DFHXCTRA</td>
<td>global trap program. The EXCI equivalent of the DFHTRAP module, providing the service with ability to collect extra diagnostic information.</td>
</tr>
<tr>
<td>DFHXCTRD</td>
<td>local trap parameter list definition. Defines the parameter list passed to DFHXCTRA and all EXCI trace points used by DFHXCTRA.</td>
</tr>
<tr>
<td>DFHXCTRP</td>
<td>trace services. Writes EXCI trace entries to the EXCI internal trace table.</td>
</tr>
<tr>
<td>DFHXCTRI</td>
<td>trace initialization. Initializes EXCI trace services.</td>
</tr>
<tr>
<td>DFHXCURM</td>
<td>User-replaceable module that allows the user to modify the applid of the CICS region to which the EXCI request is to be issued.</td>
</tr>
</tbody>
</table>

Exits

There are no exit points for the EXCI.

Trace

The EXCI has its own internal trace table in the EXCI address space where the client program is running. EXCI trace entries can also be written to the MVS GTF trace data set.

EXCI trace point IDs are EXxxxx, with a trace level of 1, 2, or Exc.

For more information about EXCI tracing, see the [CICS External Interfaces Guide](#).

For more information about the trace points, see the [CICS Trace Entries](#). For more information about using traces in problem determination, see the [CICS Problem Determination Guide](#).
Chapter 39. Field engineering program

The field engineering program (DFHFEP) is a CICS system service function primarily designed for an IBM field engineer to use when installing new terminals. When CICS is running, this program (invoked by the CSFE transaction) transmits a set of characters to the requesting terminal. In addition, the program can be used to echo a message; that is, it repeats exactly what is keyed at the terminal.

This program also supports some general debugging functions.

Design overview

When used for testing terminals, DFHFEP first prepares for device-dependent conditions. It then issues a storage control FREEMAIN, followed by a GETMAIN for storage for the ENTER message, which it writes using terminal control WRITE, READ, and WAIT macros. Finally, if print was requested, the character set is printed; if end was requested, the completion message is issued; otherwise the input is echoed.

DFHFEP performs all the requests made by the CSFE transaction. In addition to the terminal test function, CSFE can request the activation or deactivation of:

- System spooling interface trace
- Terminal builder trace
- Storage freeze
- Storage violation trap
- Global trap/trace exit.

See the CICS Supplied Transactions manual for details of the command syntax and functions provided.

Modules

DFHFEP

Exits

No global user exit points are provided for this function.

Trace

No trace points are provided for this function.
Chapter 40. File control

File control provides a facility for accessing data sets, files, and data tables, using keyed or relative-byte-address (RBA) access through the virtual storage access method (VSAM), the basic direct access method (BDAM), shared data table services and the coupling facility data tables server. VSAM data sets can be accessed in either RLS or non-RLS mode. RLS mode allows sharing of data sets across a parallel sysplex. File control allows updates, additions, deletions, random retrieval, and sequential retrieval (browsing) of logical data in the data sets. If VSAM is used, access to logical data can be via a VSAM alternate index path, as well as through the base data set.

File control reads from, and writes to, user-defined data sets and data tables, gathers statistics, and acquires dynamic storage for I/O operations. File control uses control information defined by the user in the file control table (FCT). This table describes the physical characteristics of all the data sets, and any logical relationships that may exist between them.

Design overview

File control provides the following services and features:

- Random record retrieval
- Random record update
- Random record addition
- Random record deletion (VSAM only)
- Sequential record retrieval
- BDAM deblocking
- Enabling and disabling of files, making them accessible to applications
- Opening and closing of files for the access method
- Exclusive control of records during update operations
- Mass record insertion (VSAM only)
- Automatic journaling and logging.

Deblocking services for BDAM data sets

CICS provides deblocking of logical records on a direct-access (BDAM) data set. This service is provided for both fixed-length and variable-length records. The data set must have been created according to standard System/370 record-formatting conventions.

Concurrency control

Protection is provided against the concurrent updating (adding, deleting or changing) of a data set record by two or more transactions (or strictly speaking, two or more units of work; a transaction may optionally consist of a sequence of units of work). This protection is in most cases achieved using locking. If a second unit of work attempts to update a record which has been locked by another unit of work, the second unit of work is normally queued until the first releases its lock. If the lock has been converted into a retained lock (this is done if a syncpoint failure occurs) then the second unit of work gets an error response rather than being
An optimized alternative to locking is used to achieve concurrency control for coupling facility data tables. This is described in the section ‘Concurrency control for coupling facility data tables’.

For a VSAM data set being accessed in non-RLS mode, CICS acquires locks (or enqueues) using the NQ domain that prevent the same record from being updated by more than one unit of work at a time. If the file is recoverable, then the lock is not released until syncpoint (that is, the end of the unit of work), otherwise it is released when the request thread completes. A request thread consists, for example, of a read update followed by a rewrite. In non-RLS mode, VSAM also provides a form of concurrency control known as exclusive control. The sphere of exclusive control is the control interval (CI), and this means that two different records cannot be concurrently updated if they are both within the same CI. Exclusive control is only maintained while a record is being updated, and is released as soon as the operation is complete.

For a VSAM data set being accessed in RLS mode, VSAM acquires locks at the record level to prevent the same record from being updated by more than one unit of work within the sysplex at a time. If the data set is recoverable, then the lock is not released until syncpoint, otherwise it is released when the request sequence completes. There is no CI locking with RLS mode.

For a recoverable BDAM file, CICS acquires locks using the NQ domain that prevent the same record from being updated by more than one unit of work at a time.

Concurrency control for coupling facility data tables
Concurrency control for coupling facility data tables is provided by using one of two update models provided by coupling facility data tables support (CFDT support).

The default is the locking update model, in which the CFDT server acquires locks at the record level to prevent the same record from being updated by more than one unit of work within the sysplex at a time. If the data set is recoverable, then the lock is not released until syncpoint, otherwise it is released when the request sequence completes.

The contention update model is an optimized alternative to using locking to achieve update integrity (concurrency control). With this model, which can be specified on a per-data table basis, no locks are acquired when a record is read for update, but if another unit of work subsequently changes or deletes this record, then the first unit of work will be informed that the record has changed (or been deleted) when it comes to rewrite or delete the record itself. The occurrence of such a contention is detected by the CFDT server, and the contention update model is only available for coupling facility data tables.

Sequential retrieval
A facility supported by CICS file control is the sequential retrieval of records from the database. This facility is known as browsing. To initiate a browse operation, the user provides either a specific or generic (partial) record reference (key) for the point at which sequential retrieval is to begin. Each subsequent get request by the user initiates retrieval of the next sequential record. The application, while in browse mode, can issue random get for update requests to a different data set, without interrupting the browse operation. For VSAM files accessed in RLS mode, the application can update the records that it is browsing. For VSAM files accessed in non-RLS mode, and BDAM files, in order to update a record of the same data
set, the application must first terminate the browse operation. The same application can concurrently browse several different data sets and browse the same data set with multiple tasks.

With VSAM data sets, the application can skip forward during a browse operation to bypass unwanted data.

All types of CICS data tables (CICS-maintained, user-maintained and coupling facility) can be browsed.

**Read Integrity**

When a file is accessed in RLS mode, three levels of read integrity are supported:

- **UNCOMMITTED** read integrity is the same level of read integrity as is supported for non-RLS requests. With this level of read integrity, read requests can return data which has not yet been committed, and which might subsequently be backed out.

- **CONSISTENT** read integrity. With this level of read integrity, read requests are serialized with concurrent update activity for the record, so that a read request will wait until data which is being updated has been committed (or until the update has completed, for a non-recoverable data set). This means that read requests will always see commit-consistent data.

- **REPEATABLE** read integrity. With this level of read integrity, additional locking is used so that in addition to waiting for updates to be committed, records that have been read within a unit of work cannot be updated until the unit of work completes. This means that if a read is repeated within a unit of work, the same data will be returned.

**Backout logging**

File control will perform automatic logging of file operations which update recoverable files. This logging is written to the CICS system log stream. In the event of either a system or a transaction failure, the information can subsequently be used to restore the recoverable data set as though the current transaction had never run.

For coupling facility data tables, the CFDT server performs its own logging, and is responsible for backing out updates in the event of a failure.

**Forward Recovery Logging**

If a file (non-RLS VSAM) or data set (RLS or non-RLS VSAM) is defined to be forward recoverable, then CICS will perform automatic logging of file operations which update it. This logging is written to the forward recovery log stream specified on the file definition or data set. In the event of a failure, the information can be used to forward recover from a backup copy of the data set.

Forward recovery support is not provided for user-maintained data tables or coupling facility data tables.

**Automatic journaling and logging**

Except in the case of user-maintained data tables and coupling facility data tables, CICS provides optional automatic journaling and logging facilities for records that are updated, deleted from, or added to a file control data set. Automatic journaling is specified in the file control table, by the user, for each data set affected. For a specified data set, a record read for update, a new record added, or an existing
Use of concurrent tasks

The file control non-RLS VSAM interface program (DFHFCVR) uses a change-mode request to the dispatcher to allow VSAM I/O requests and VSAM UPAD exit code to run under a concurrent task. This provides overlapping of processing in a multiprocessor environment.

RLS requests use a different mechanism: SMSVSAM assigns each request its own SRB, allowing MVS to concurrently schedule requests in an analogous way to that provided by subtasking for non-RLS.

Shared Data table services

Shared data tables (that is, CICS-maintained and user-maintained data tables) are managed by a set of OCO modules, referred to in this book as "data table services". The services are invoked by a branch-and-link interface passing a parameter block.

Services provided include the following:
- Initialization
- Open, close, and load of tables
- Retrieval and update of table records
- Backout and commit of table changes
- Statistics.

For files that are defined by the user as CICS-maintained or user-maintained data tables, file control invokes these services at appropriate points in the processing of application requests.

Coupling facility data tables server

Coupling facility data tables are managed by a OCO modules within the CICS address space, along with a separate address space, referred to as the "Coupling Facility Data Tables Server". The CFDT server provides access to coupling facility data tables residing in a coupling facility data tables pool, so that they can be shared by CICS regions across a parallel sysplex. Refer to the CICS Supplied Transactions for more details about CFDT servers.

For files that are defined by the user as accessing coupling facility data tables, file control makes calls to the CFDT server at appropriate points in the processing of application requests.

How CICS processes file control requests

CICS receives file control requests from applications through the EXEC interface. This section describes only the mainstream processing for such requests. It does not describe exceptional conditions. For guidance about exceptional conditions, see the CICS Application Programming Guide. For general-use programming interface information about exceptional conditions, see the CICS Application Programming Reference manual. This section also does not provide details about the specific processing for requests to any kind of data table.
Processing using VSAM

For VSAM data sets, this section describes the processing followed when the file is being accessed in non-RLS mode. For RLS mode, the processing is broadly similar, although it differs in some of the interfaces used to VSAM, and the locking mechanisms are very different.

Note: File control processing is constrained by the availability of buffers, CICS strings and (for local shared resource (LSR) files) LSR strings. Tasks can get suspended during the execution of any file control request if there are not enough strings or buffers available for the immediate processing that is to be done.

With VSAM RLS, a task waiting for buffers will be suspended in VSAM rather than in CICS.

Processing using Data Tables

For shared data tables (CICS-maintained and user-maintained data tables), processing is broadly similar to that for non-RLS VSAM. The main differences are that, for remote files, non-update requests may be processed locally instead of being function shipped, and that, in cases where a request cannot be satisfied from a data table, it may be converted into a non-RLS or RLS VSAM request to be processed by DFHFCVS or DFHFCRS, or function shipped via DFHFCDTX.

For coupling facility data tables, processing is also broadly similar to that for non-RLS VSAM. The main difference is that instead of issuing the request to VSAM, a call or calls are made to entry points within the CFDT server, which then processes the request and returns the results. A task accessing a coupling facility data table may occasionally be suspended in the CFDT server.

Note that the following processing sections do not describe data table processing explicitly.

General request processing

All file requests, whatever the request and whatever the file access method, follow the same general sequence of steps:

1. User exit XFCREQ is called.
2. The request is converted from EXEC parameter list form to FCFR interface form.
3. If this is the first file access request by the transaction, a FRAB is obtained and its address stored in Recovery Manager’s FC Token. The FRAB provides the anchor for file request state for this transaction.
4. If this is the first request to this file by the transaction, a FLAB is obtained and the file control table entry is located. If the file is remote or an explicit SYSID has been specified on the request, the FLAB is marked with a remote indicator. If this is not the first request to the file, then the FLAB is located that represents accesses made to the file by this transaction.
5. If this is the first, or only, request of a request sequence, a FRTE is obtained. If this is not the first request in a request sequence, the FRTE that represents the sequence is located rather than being function shipped.
6. If the request is to a local file, and if resource security is active, the security check is made, unless a check has already been made within the current UOW for this file.
File control

7. If the request is to a local file and the file is not already open, it is opened and its access method dependent attributes are saved in its file control table entry.
8. The SERVREQ attributes of the file are checked.
9. For READ and browse requests, SET storage is released and/or obtained, as necessary.
10. The access method specific request processor is called as follows:
   - DFHFCVS for non-RLS VSAM files
   - DFHFCRS for RLS VSAM files
   - DFHFCBD for BDAM files
   - DFHFCDR for coupling facility data tables
   - DFHFCDT for user-maintained data tables
   - DFHFCDR for requests to CICS maintained data tables
   - DFHFCVS for update requests to CICS maintained data tables
   - DFHFCRF for requests that are to be shipped to a remote region
11. CICS has checked whether the file is defined as local or remote. If it is remote, the request is function-shipped to the file-owning region, where CICS processes the request as if it had originated locally.
   There is an exception for CICS-maintained and user-maintained data tables, for which non-update requests are treated as local rather than being function shipped.
   Note that RLS support and coupling facility data tables support both provided shared access within a parallel sysplex without the use of function shipping. Files which use either of these types of sharing will be defined as local on all systems which wish to share the data set (in the case of RLS support) or data table (CFDT support).
12. SET storage is obtained for BDAM files or below the line READ requests.
13. The FRTE is released if the request sequence has ended and the file is closed if a close is pending, this FRTE is the last user and the FLAB indicates that the file can be closed.
14. The FCFR responsed are converted to EXEC parameter list responses. In particular, the EIBRCODE and RESP2 values are constructed.
15. User exit XFCREQC is called.

READ request processing

The course of READ request processing depends on the access method, and whether or not the UPDATE option is specified on the request:

VSAM processing:
1. The supplied keylength is validated.
2. A VSAM work area (VSWA) is created. This includes the request parameter list (RPL) that will be passed to VSAM.
   The processing that follows depends on whether the UPDATE option was specified on the READ request.
   UPDATE option not specified:
   a. The RPL is completed, and a call made to VSAM to get the record.
   b. If the request specifies INTO and the record is too large for the user-specified area, the request is reissued specifying a work area large enough to hold the record. The record is then copied to the user-specified area in truncated form, and the LENGERR condition is raised.
   c. The VSWA is freed.
d. The read is journaled if specified in the FCT entry.

**UPDATE option specified:**

a. The UPDATE flag is set in the RPL.

b. An attempt is made to read the record by issuing the VSAM request. READ UPDATE requires exclusive control of the control interval (CI) containing the record. VSAM manages the locking mechanism for control intervals. If the CI is already locked, VSAM returns an error and the requesting task is forced to wait on resource type FCXCWAIT.

c. CICS file control acquires a record lock on the record just read, using a CICS ENQUEUE request. The record lock prevents any other transaction from updating the record before the owning transaction has reached a syncpoint (for recoverable files), or before the REWRITE, DELETE, UNLOCK or syncpoint that completes the request sequence (non-recoverable files).

d. Exclusive control of the CI is retained until the REWRITE, DELETE, or UNLOCK request that follows the READ UPDATE has been completed, or until the next syncpoint.

   The CICS record lock (if any) is retained until the next syncpoint, in case the transaction updating the record abends and dynamic transaction backout processing is necessary.

e. If the file is recoverable the request is logged. If required, the request is also recorded in a user-specified journal.

**BDAM processing:**

a. A file I/O area (FIOA) is obtained.

b. If the UPDATE option has been specified:
   1) The address of the RIDFLD is saved in the FIOA.
   2) If the data set is recoverable, the RIDFLD is ENQUEUEd on to lock the record against other updates. The ENQUEUE is retained until the next syncpoint.

c. The KEYLENGTH is checked for validity.

d. The key field is converted from character string format (TTTTTTTRR) to binary format (TTR), if necessary.

e. A BDAM READ request is issued. If the READ is successful, the required block is returned in the FIOA.

f. The key field returned by BDAM is converted from binary format to character string format, if necessary.

g. If the file is recoverable and UPDATE has been specified, the request is logged. If required, the request is also recorded in a user-specified journal.

h. If deblocking is required, the required record is located in the block that has been returned by BDAM:
   1) If DEBREC has been specified, the record number is used to locate the record.
   2) If DEBKEY has been specified, the embedded key is used to locate the record.

**WRITE request processing**

The course of WRITE request processing depends on the access method, and for VSAM access on whether the file is a data table: **VSAM processing:**

1. The KEYLENGTH is checked for validity. If it is incorrect, the INVREQ condition is raised.
2. A VSAM work area (VSWA) is created. This includes the request parameter list (RPL) that will be passed to VSAM.

_Different paths are now followed depending on the type of file._

**ESDS file:**

a. If the file is recoverable or writes are to be journaled then

1) If this is not the first write of a sequence and the ESDS write lock is being waited for by another transaction, then release the lock and end this sequence, logging the completion if recoverable.

2) If this is (or has become) the first write of a sequence, acquire the ESDS write lock for the data set.

b. If the file is recoverable, the WRITE ADD request is recorded in the CICS system log.

c. If required, the WRITE ADD request is recorded in a user-specified journal.

d. Any fields in the RPL not supplied when the VSWA was created are completed.

e. The RPL is set to point to the user-specified data area. If the user specified a record that is too large for the file, the length in the RPL is set to the maximum length, so that the record is truncated.

f. A VSAM PUT request is issued to write the record.

g. If the file is recoverable, a CICS record lock is obtained for the record that has just been written. The record lock will be retained until the next syncpoint, in case the transaction writing the record abends and dynamic transaction backout processing has to be performed.

h. If the file is recoverable, the after-image of the record is logged for forward recovery and a write complete record is written on the system log.

i. If not a MASSINSERT the ESDS write lock is released, if held.

**KSDS or RRDS file:**

a. For KSDS requests, the RIDFLD key specified in the request is checked against the key field in the record to be written. (The record is currently in the application FROM data area.) If it does not match, the INVREQ condition is raised.

b. If the file is recoverable and not in load mode:

1) A CICS lock is obtained on the record that is to be written, and an attempt is made to read the record (by means of a VSAM GET request) to discover whether it already exists in the file. If it does, the DUPREQ condition will be raised on the write to VSAM.

2) If the file is a KSDS, and if this request is part of a MASSINSERT, or if a MASSINSERT is in progress, the read is issued with GTEQ to find the next record in the base data set. A lock is created, using the key of this next record, to prevent other transactions from inserting records into the empty range.

3) If there is no existing record with the given key, the WRITE ADD request to VSAM is recorded in the CICS system log and, if required, in a user-specified journal.

c. If the file is not recoverable or in load mode, the WRITE request is recorded, if required, in the user-specified journal, and if recoverable a record lock is obtained and the write logged.

d. Any fields in the RPL not supplied when the VSWA was created are completed.
If a data table is associated with the base cluster (the data table will be a CICS-maintained table, as user-maintained and coupling facility data tables follow a separate processing path which is not described here). a data table pre-add is issued to place the record in the table as a not-yet-valid entry. If the file is recoverable, a record lock is already held; if not, a lock is acquired before the data table service is called.

A VSAM request is issued to write the record.

If the file is recoverable, the after-image of the record is logged for forward recovery.

If required, the after-image is recorded in a user-specified journal.

If the file is a data table, a data table request is issued to complete the add to the data table by validating the record. If a record lock was obtained for a non-recoverable file, it is released.

If the MASSINSERT option has not been specified on the WRITE request, the VSWA for the operation is released.

If MASSINSERT has been specified, the VSWA is not released, because it is likely to be needed for subsequent WRITE operations. In this case, the end of MASSINSERT processing is notified to VSAM by the CICS UNLOCK function. (See "UNLOCK request processing" on page 548.) Specifying MASSINSERT causes exclusive control of the CI to be acquired. Exclusive control is released by issuing an UNLOCK request. To avoid deadlocks, this should be done immediately after the last WRITE MASSINSERT request.

**BDAM processing:**

1. The KEYLENGTH is checked for validity. If it is incorrect, the INVREQ condition is raised.
2. The WRITE command input is checked to ensure that MASSINSERT has not been specified—BDAM does not support MASSINSERT processing. If it has, condition INVREQ is raised.
3. A file I/O area (FIOA) is obtained.
4. If the file is recoverable, the record to be written is ENQUEUEd on. The lock is retained until the next syncpoint.
5. The record to be written is copied from the user-supplied data area to the FIOA. If the record is too large, it is truncated.
6. If the file is recoverable, the request is logged. If required, the request is also recorded in a user-specified journal.
7. The key field is converted from character string format to binary format, if necessary, and the BDAM I/O request issued.
8. The key returned by BDAM is converted from binary format to character string format, if necessary, and passed to the application.
9. A supervisor call (SVC 53) is issued to release BDAM exclusive control, if necessary.
10. The FIOA is FREEMAINed.

**REWRITE request processing**

The REWRITE request is used to write a record back to a file following a READ UPDATE request. **VSAM processing:**

1. The RPL is set to point to the user-specified data area. If the user specified a record that is too large for the file, the length in the RPL is set to the maximum length, so that the record is truncated.
2. The RPL is completed.
3. If there is a data table associated with the base cluster (this will be a CICS-maintained table, as user-maintained tables follow data table processing):
   a. If the file is nonrecoverable, a record lock is obtained. (If the file is recoverable, a lock is already held).
   b. A data table request is issued to invalidate the record in the table before the VSAM update.
4. VSAM is called to PUT(UPDATE) the record. Exclusive control of the CI, which was obtained for the preceding READ UPDATE request, is released, but the CICS record lock (for recoverable files) is retained until the next syncpoint, in case the transaction abends and dynamic transaction backout processing is necessary.
5. If there is a data table associated with the data set, the table record is updated and its validity is reinstated, by issuing a call to data table services. If the file is nonrecoverable, the record lock is released.
6. If the file is recoverable, and if the record is successfully rewritten, the after-image is written to the log for forward recovery.
7. The VSWA for the operation is released.

   Note: When a record is updated by way of a path, the corresponding alternate index is updated by VSAM to reflect the change. However, if the record is updated directly by way of the base, or by a different path, the AIX will only be updated by VSAM if it has been defined to VSAM (when created) to belong to the upgrade set of the base data set.

BDAM processing:
1. The FIOA that was used in the corresponding READ UPDATE request is located, and the modified record read into it from the user-specified area. If the record is too long, it is truncated.
2. A FREEMAIN call is issued to release the FWA.
3. If the file is recoverable, the request is logged. If required, the request is also recorded in a user-specified journal.
4. The key field is converted from character string format to binary format, if necessary, and the BDAM I/O request issued.
5. The key returned by BDAM is converted from binary format to character string format, if necessary, and passed to the application.
6. A supervisor call (SVC 53) is issued to release BDAM exclusive control, if necessary.
7. A FREEMAIN call is issued to release the FIOA.

UNLOCK request processing
The UNLOCK request is used to release exclusive control obtained during a READ UPDATE (VSAM or BDAM) or WRITE MASSINSERT (VSAM only) request.

VSAM processing (including CICS-maintained data tables):
1. The VSWA for the operation is released, together with associated storage.
2. An ENDRREQ request is sent to VSAM. This releases exclusive control of the CI, if it is held, and frees any VSAM strings.

BDAM processing:
1. A supervisor call (SVC 53) is issued to release BDAM exclusive control, if necessary.
2. A FREEMAIN call is issued to release the FIOA.

DELETe request processing
The course of DELETE request processing depends on whether a RIDFLD has been specified. The processing for user-maintained data tables differs from that for CICS-maintained data tables. DELETE requests are not valid for VSAM ESDS or for BDAM files.

VSAM processing (including CICS-maintained data tables):
1. If a RIDFLD has been specified:
   a. If a KEYLENGTH has been specified, it is checked for validity.
   b. If the GENERIC option has been specified, and the file is not a KSDS, condition INVREQ is raised.
   c. A VSWA is created.
2. If no RIDFLD was specified, the SERVREQ attribute of the file is checked to ensure that DELETE requests are valid for this file. If not, the INVREQ condition is raised.
   If a RIDFLD has been specified, the cycle of actions described below is performed once if GENERIC has not been specified, or is repeated until there are no more records containing the generic key, if GENERIC has been specified.

   Start of cycle:
3. VSAM is requested to GET for UPDATE a record with the specific or generic key. GET UPDATE processing requires exclusive control of the CI. The record is read into an internal buffer.
   The generic key value, if supplied, is checked against the key contained in the record. If it does not match, there are no more records containing the generic key in the file.
4. If the file is recoverable:
   a. A CICS record lock is obtained for the record. This will be held until the next syncpoint.
   b. The VSAM GET UPDATE request is recorded synchronously on the system log.
   c. A CICS range lock is obtained for the record to be deleted if a MASSINSERT is in progress. This is to prevent an end-of-range record from being deleted while the range is in use for a MASSINSERT sequence.
5. If there is a data table (which will be CICS-maintained) associated with the base cluster, a record lock is acquired if the file is nonrecoverable, and a data table pre-update call is issued to invalidate the record before the VSAM update.
6. A VSAM ERASE request is issued, to delete the record from the file.
7. If there is a data table associated with the base cluster, the record is deleted from the table by issuing a call to data table services. If the file is nonrecoverable, the record lock is released.
8. If a range lock was acquired, it is released.
9. If the file is recoverable, a WRITE DELETE record is written in the system log for forward recovery.
10. If required, a WRITE DELETE record is written to a user-specified journal.
   End of cycle.
11. The VSWA is released.
STARTBR and RESETBR request processing
STARTBR and RESETBR request processing are very similar, and are described together.

VSAM processing:
1. A VSWA is created if STARTBR.
2. The user key is recorded in the VSWA for use in subsequent BROWSE processing.
3. A call is made to VSAM to point to the record, and to acquire shared control of the CI.

BDAM processing:
1. An FIOA is obtained and initialized if STARTBR.
2. The initial key is saved in the FIOA, converting the key from character string format to binary format if necessary.
3. If deblocking is required, the deblocking indicator (DEBREC or DEBKEY) is saved in the FIOA.

READNEXT and READPREV request processing
READNEXT and READPREV request processing are very similar, and are described together.

VSAM processing:
1. A check is made that READPREV with a generic key was not requested. If it was, condition INVREQ is raised.
2. If KEYLENGTH was specified, it is checked for validity. If it is incorrect, the INVREQ condition is raised.
3. The RPL options are set.
4. If SET is specified, an internal work area is obtained and the RPL is set to point to the work area. The area is either above or below the 16MB line, depending on the requirements of the application.
5. If INTO is specified, the RPL is set to point to the user-specified area.
6. A VSAM request is issued to read the record. Shared control of the CI is needed, and the request will not succeed if some other task already has exclusive control. In such a case, a call is made to VSAM to reestablish the correct position in the file. The task then waits until VSAM informs CICS that the CI is available to the task. CICS resumes the task, which can now acquire shared control and obtain the required record.
7. If SET is specified, the SET pointer points to the work area.
8. If INTO is specified, a check is made to see if the record is too large to fit into the user-specified area. If it is too large, the request is reissued using an internal work area, the data is copied from the work area into the user-specified area and truncated, and the LENGERR condition is raised.
9. If required, the request is recorded in a user-specified journal.

BDAM processing—READNEXT requests:
1. A check is made that READPREV was not issued. If it was, condition INVREQ is raised.
2. The FIOA that was created on STARTBR is located.
3. If a new block is required, a BDAM I/O request is issued to get it.
4. If deblocking is required, the required record is located in the block that has been returned by BDAM:
a. If DEBREC has been specified, the record number is used to locate the record.
b. If DEBKEY has been specified, the embedded key is used to locate the record.

5. If INTO is specified, the record or block is moved from the FIOA to the user-specified area. If the record is longer than the user-specified area, it is truncated, and the LENGERR condition is raised.

6. If SET is specified, the SET pointer points to the record in the FIOA.
7. The RIDFLD of the record is returned to the application.
8. The current browse position is recorded in the FIOA.

**ENDBR request processing**
The ENDBR request is used to end a browse session on a file. To avoid deadlocks, ENDBR must be issued when the browse session is complete.

**VSAM processing:**
1. An ENDREQ request is sent to VSAM. This frees any VSAM strings that are held, and relinquishes shared control of the CI.
2. The VSWA for the operation is released.

**BDAM processing:**
- The FIOA that was used for the browse session is FREEMAINed.
Figure 60 on page 0 shows the major control blocks associated with file control. Control blocks which are not shown in this diagram include those relating to coupling facility data tables support.

\[
\begin{array}{|l|}
\hline
| Block Name | Address | Description |
\hline
| Recovery Manager UOW representation | | |
| FC work token | | |
| FRAB | | |
| FRAB_NEXT_FRAB_ADDRESS | X'10' | |
| FRAB_PREV_FRAB_ADDRESS | X'14' | |
| FRAB_FLAB_CHAIN_ADDRESS | X'18' | |
| FRAB_FLLB_CHAIN_ADDRESS | X'1C' | |
| FLAB | | |
| FLAB_NEXT_FLAB_ADDRESS | X'10' | |
| FLAB_FCTE_ADDRESS | X'20' | |
| FLAB_FRTE_CHAIN_ADDRESS | X'28' | |
| FLAB_SET_CONTROL | X'30' | |
| FRTE | | |
| FRTE_NEXT_FRTE_ADDRESS | X'00' | |
| FLAB_FCTE_ADDRESS | X'08' | |
| FRT_NEXT_IN_FILE_CHAIN | X'14' | |
| FRT_WORK_AREA_ADDRESS | X'30' | |
| FLLB | | |
| FLLB_DSNB_ADDRESS | X'00' | |
| FLLB_NEXT_IN_DSNB_CHAIN | X'5C' | |
| FLLB_NEXT_IN_FRAB_CHAIN | X'60' | |
| FCT ENTRY (FCTE) | | |
| FCTDSID | X'00' | |
| FCTDSDP | X'5C' | |
| FCTDSBPC | X'60' | |
| DNAME BLOCK | | |
| FCTBC_FLLB_CHAIN | | |
| VSWA | | |
| VSWAACB | X'28' | Address of ACB |
| VSWAFC | X'54' | Address of FCT entry |
| FIOA | | |
| FCFIODCB | X'0C' | Address of DCB |
| FCFIOFCT | X'34' | Address of FCT entry |
| FIOADBA | X'60' | Data area |
\end{array}
\]
Access method control block (ACB)

The ACB identifies to VSAM the file associated with this VSAM request. It is passed to VSAM by DFHFCRV, for RLS, or DFHFCVR, for non-RLS (it is actually the RPL, which points to the ACB, that is passed) to initiate a VSAM request. The ACB lasts as long as the associated CICS file is open; that is, it is created at file open time and deleted at file close time by DFHFCN for non-RLS or DFHFCRO for RLS.

Note: The pointer to the DSNAME block, FCTSDP, is different from the pointer to the base cluster DSNAME block, FCTDSBCP, only when the FCT entry does not represent a base. DSNAME blocks that do not correspond to bases do not have the base cluster information, although the space is allocated.

These control blocks are described in the §§ “Access method control block (ACB)” through “VSAM work area (VSWA)” on page 563.
File control

The ACB is addressable through a pointer in the associated FCT entry. In addition, a 4-byte field appended (by CICS) to the ACB structure points back to this FCTE.

Note that the ACB is a VSAM control block.

At open time, storage is obtained from a subpool above the 16MB line. A VSAM GENCB macro is issued to generate the ACB with attributes obtained from the FCT entry. At open time, VSAM fills in more information in the ACB. Some of this is subsequently copied back into the FCTE.

The storage for the ACB is freed when the file is closed.

There is one ACB per VSAM FCT entry.

The layout of the ACB is defined by the VSAM IFGACB structure, and also by a DSECT of the same name.

ACBs are not cataloged and are not restored across WARM or emergency starts. The ACB is rebuilt every time a CICS file is opened.

A special type of ACB, known as a base cluster ACB, is created by DFHFCM to allow for the implicit opening of a base cluster, when required by a non-RLS file access through an alternate index path. In this case, the 4-byte field appended to the ACB structure points to the associated DSNAME block for the base cluster.

A second special type of ACB, known as a control ACB, is required for VSAM RLS processing. Storage for the control ACB is obtained by DFHFCCA and filled in using the GENCB macro before registering the control ACB. The storage is freed when the control ACB is unregistered by DFHFCCA. The control ACB is passed to VSAM on calls issued by DFHFCCA. It is used for all requests that are not associated with a specific file.

Data control block (DCB)

The DCB identifies to BDAM the file associated with this BDAM request. It is passed to BDAM by DFHFCBD to initiate a BDAM request, and lasts for the lifetime of the CICS run.

The DCB is addressable through a pointer in the associated FCT entry. In addition, a 4-byte field appended (by CICS) to the DCB structure points back to this FCTE.

Note that the DCB is a BDAM control block.

There is one DCB per BDAM FCT entry.

The layout of the DCB is defined by the generalized structure IHADCB. The structure is qualified with a parameter stating that a BDAM DCB is required. There is also a DSECT of the same name.

The DCB is assembled as part of the FCT. (Note that there is no RDO for BDAM files.) DFHFCRP acquires storage for the DCB below the 16MB line and copies the DCB into it (only on cold start). The DCB is cataloged and restored across a warm and emergency start. Thus, unlike an ACB, a DCB is only built once.
Data set name block (DSNB)

The DSNB represents a physical VSAM or BDAM data set that is being accessed through one or more CICS files. It is used by file control to hold information relevant to the data set and not only to the CICS file. Also, it provides a single “anchor block” to control many requests accessing this data set through many different CICS files.

After it has been created, a DSNB survives the lifetime of a CICS run unless the user deletes it by means of an EXEC CICS SET DSNAME REMOVE command or its CEMT equivalent.

The DSNB is addressable through pointers in an FCTE entry, or through DFHTMP using the 44-character name as a key, or using the DSNB number as a key.

A DSNB is created, if it does not exist already, when an FCTE attempts to connect itself to a DSNB. This happens at file open time, or when an EXEC CICS SET FILE DSNAME command (or its CEMT equivalent) is executed.

A DSNB that represents a VSAM base data set has a base cluster block embedded in it, which has information specific to the base data set. Note that a BDAM data set has a small amount of information held in the base cluster block.

A DSNB representing a VSAM path has a blank base cluster block embedded in it.

Information about the base data set is obtained from the VSAM catalog when a CICS file (path or base) referencing that data set is opened. The information is stored in the base cluster block.

DSNBs are cataloged in the CICS global catalog and are restored across warm and emergency starts.

DSNBs reside above the 16MB line.

The layout of the DSNB is defined by the DFHDSNPS structure, and by the DFHDSNDS DSECT (using the DFHDSND macro).

The DFHFCDN module handles DSNAME blocks (creation, deletion, FCTE-DSNB connections). DFHFCDN also provides an interface for the EXEC layer to process DSNAME blocks through the use of EXEC CICS INQUIRE or SET DSNAME, and CEMT INQUIRE or SET DSNAME. Modules within the file control component can access the DSNBs directly through pointers in the FCTE.

File browse work area (FBWA)

The FBWA maintains the state of a browse to a data table. It is used for browsing coupling facility data tables, CICS-maintained data tables, and user-maintained data tables.

An FBWA is created when the browse is started (via a STARTBR request), and is addressed by the FRT_FBWA_ADDRESS field in the FRTE. It is stored in a file control IO buffer of the appropriate size to hold the key information.

Some of the fields are specific to CICS-maintained data tables, because the source data set will sometimes be accessed during a browse of a CICS-maintained data table.
File control

There is a variable-length portion at the end of the FBWA which contains keys, which are pointed to by fields in the fixed hang on part:

- CURRENT_KEY points to the first of the key fields, which is used to hold the key returned by the most recent request.
- REQUEST_KEY points to the second of the key fields, which is used to contain the key specified at the start of a browse segment (STARTBR or RESETBR).
- NEXT_KEY points to the third of the key fields, which is used for CICS-maintained data tables to handle "gaps".

File control static storage (FC static)

File control static storage is used by file control to store information for use throughout the lifetime of a CICS run; for example, SHRCTL vectors and entry points of file control modules. It is used by file control modules and by modules outside the file control component, and lasts for the lifetime of a CICS run. It is addressed by a field in the CSA named CSAFCSBA; it is created by DFHFCIN during CICS initialization before DFHFCRP gets control, and resides above the 16MB line.

FC static storage is defined by the DFHFCSPS structure and by the DFHFCSDS DSECT.

File control quiesce receive element (FCQRE)

File control uses quiesce receive elements to communicate details of quiesce requests received from SMSVSAM. There is also a permanent error FCQRE used for communicating errors. The FCQRE contains information about the data set to which the quiesce applies (or the cache for quiesce type QUICA), the type of quiesce, and (for the error FCQRE) the type of error and error data.

Each quiesce request received from SMSVSAM via the quiesce exit results in DFHFCQX, the quiesce exit module, creating an FCQRE which is passed to DFHFCQR, the quiesce receive system task module.

Storage for FCQREs is obtained from storage MVS getmained above the 16MB line.

FCQREs are chained in a one-way linked list anchored from file control static storage. The permanent error FCQRE is also anchored from file control static storage, and is added to the FCQRE chain when an error occurs.

The layout of the FCQRE is defined by the DFHFCQRE structure and the DFHFCQRE DSECT.

File control quiesce send element (FCQSE)

File control uses quiesce send elements to communicate the details of quiesce requests that are to be sent to SMSVSAM. They contain information about the task initiating the request, the data set to be quiesced, the type of quiesce requested, and the address of an ECB which is posted by SMSVSAM when the request is completed.

Each quiesce request initiated by CICS results in DFHFCQI, the quiesce initiate module, creating an FCQSE which is passed to DFHFCQS, the quiesce send module.
Storage for FCQSEs is obtained from the FC_ABOVE subpool, which resides above the 16MB line.

FCQSEs are chained in a two-way linked list anchored from fields in file control static storage.

The layout of the FCQSE is defined by the DFHFCQSE structure and the DFHFCQSE DSECT.

**File control coupling facility data table pool element (FCPE)**

A file control CFDT pool element represents one connection to a Coupling Facility Data Table Pool. For each CFDT pool which can be accessed by a given MVS image, there is a CFDT server running in that image which manages access to the pool.

An FCPE is created and chained to FC static when a file definition that refers to the pool is installed and there is not already a pool element for that CFDT pool. The creation of an FCPE can occur:
- when files are installed at CICS startup,
- when files are installed using CEDA,
- when a SET FILE is issued which names a CFDT pool for which there is not already a pool element.

FCPEs are getmained from the FCPE subpool which is created by DFHFCRP during File Control Initialization, and chained to the FCPE chain in FC static. The head of the FCPE chain is the field FC_FCPE_CHAIN.

FCPEs are catalogued when they are created, so that they can be restored at emergency restart.

**File control coupling facility data table pool wait element (FCPW)**

The file control CFDT pool wait element (FCPW) represents a task which has tried to issue a request to a coupling facility data table that resides in a particular pool, but which has to wait because there are no available request slots. Depending on the kind of request, the FCPW will represent either a ‘Locking request slot’ (LRS) waiter or a ‘MaxReqs’ waiter. A flag in the FCPW indicates what kind of wait it is.

The FCPW is created when a task goes into a MaxReqs or LRS wait. It is getmained from the pool wait element subpool, and appended to a chain of wait elements for the pool. The wait chains are anchored in the pool element (FCPE), with one FCPW for each task that is waiting. The FCPE contains head and tail fields for the chains of LRS and MaxReqs FCPWs (FCPE_FIRST_LRS_WAITER, FCPE_LAST_LRS_WAITER, FCPE_FIRST_WAITER and FCPE_LAST_WAITER). The chains are manipulated using logic which does not require any special case code for the ends of the chains, but which does mean that when the chains are empty, the head and tail fields contain a special initial value, rather than zero.

The FCPW includes:
- A pointer to the next FCPW in the chain (if no next FCPW, this contains the special initial value).
- A pointer to previous FCPW in the chain (if no previous FCPW, this contains the special initial value).
File control

- The suspend token for the wait.
- The task token of the waiting task.
- The suspend start time.

File control table entry (FCTE)

Each entry in the file control table defines a CICS file that is defined to be the CICS view of a VSAM or BDAM data set or a data table. The FCTE is used by all modules in the file control component (but never outside), and lasts for the lifetime of a CICS run, or from when it is created by RDO to the end of the CICS run.

The FCTE is addressable through a TMP index; its layout is defined by the DFHFCTPS structure and by the DFHFCNTDS DSECT; and it resides above the 16MB line.

The FCTE contains information that can be split into three broad groups:
- CICS information about the file, including statistics
- Information that is used as input to build the VSAM ACB or BDAM DCB
- Information that is returned by VSAM, both from the ACB and direct from the VSAM catalog, when the file is opened.

An FCTE can be created in two ways:
- By defining the file using the DFHFCCT TYPE=FILE macro (BDAM only).
- By defining the file online using RDO while CICS is running (VSAM only).

File control table entry (FCPW)

File control coupling facility data tables UOW pool block (FCUP)

The File Control CFDT UOW Pool Block (FCUP) represents recoverable updates made within a unit of work to one or more CF data tables residing in a coupling facility data table pool. An FCUP block is created when a unit of work makes its first recoverable request to a CFDT in a given pool, at the same time as an RMC link is added to represent the recoverable update.

There is one FCUP block per UOW per recoverably-updated CFDT pool. The FCUP is getmained and freemained from the FCUP subpool using the storage manager quickcell mechanism. The FCUP blocks for a unit of work are chained from the FRAB for that unit of work, addressed by FRAB_FCUP_CHAIN_ADDRESS.

An FCUP block contains:
- Forward and back pointers for the chain of FCUP blocks relating to this unit of work.
- The name of the CFDT pool.
- The CFDT RMC link token.
- A pointer to the pool element for the CFDT pool.
- A pointer back to the owning FRAB.
File input/output area (FIOA)
The FIOA is analogous to the VSWA for VSAM, in that it represents the request to BDAM. Embedded in the FIOA is what is known as the data event control block (DECB), which is passed to BDAM to initiate the request.

The FIOA is used by DFHFCBD when processing browse requests against BDAM files. It holds position in a browse when browsing a BDAM file.

An FIOA survives as long as the DECB needs to survive to complete the BDAM request; for example, it survives from READ UPDATE to the REWRITE request.

The address of the FIOA is held in the file request thread element (FRTE) in the FRT_WORK_AREA_ADDRESS field.

Storage for the FIOA is acquired from below the 16MB line.

The layout of the FIOA is defined by the DFHFIOA DSECT.

File lasting access block (FLAB)
The FLAB serves as an anchor for the set of file request thread elements (FRTEs) belonging to a particular file within a given transaction and a given environment. If a transaction accesses several files from within the same environment, there will be one FLAB for each file. If a transaction accesses the same file from more than one environment, there will be one FLAB for each environment.

The FLAB contains pointers to the FCTE for the file, to the owning FRAB, to the chain of FRTEs owned by the FLAB, and to the next FLAB in the chain of FLABs for the unit of work.

The FLAB is used by file control to
- anchor the FRTEs for the file within the unit of work and environment,
- ensure that a file cannot be closed if there are any FRTEs associated with it, or if there have been recoverable updates made by units of work which have not yet reached syncpoint phase 2,
- ensure that the corresponding file entry cannot be reallocated to a different data set, even if the file is closed and disabled, when there is uncommitted recoverable work associated with the file,
- hold READ SET storage control information across intermediate syncpoints,
- ensure that units of work which have updated the file reach syncpoint before a copy or BWO copy for a file opened in RLS mode is allowed to proceed,
- record the reason for a failure during syncpoint, and keep track of the fact that the file has uncommitted updates within a unit of work as a result of the failure.

The file lasting access block is built by DFHFCFR as part of processing of the first file control request for a particular file within a given transaction and environment. FLABs for recoverable files are also rebuilt by DFHFCIR at warm and emergency restart.

The storage for the FLAB is obtained from a FLAB storage subpool above the 16MB line.

The FLAB is deleted after all the FRTEs have been processed during syncpoint terminate processing, providing that there have been no syncpoint failures for the
File control

File within the unit of work. The FLAB storage is not returned to the FLAB storage subpool, but is instead added to a chain of free FLABs, anchored from file control static storage. Subsequent requests to build a FLAB are, if possible, satisfied by a quick cell mechanism from this chain.

If a unit of work is shunted as a result of a syncpoint failure, the FLABs for any files which suffered the syncpoint failure are also shunted.

The chain of FLABs for a unit of work is anchored from field FRAB_FLAB_CHAIN_ADDRESS in the FLAB.

The layout of the FLAB is defined by the DFHFLAB structure and the DFHFLAB DSECT.

File control locks locator blocks (FLLBs)
The file control locks locator block records the fact that a unit of work held locks against a file which were protecting uncommitted changes to the file, and that it is now uncertain whether the locks are valid. This can occur, for example, if the data set against which the locks were held is now in the lost locks state, or if a non-RLS open for update has taken place despite the presence of retained locks and has overridden the locks (in this case the locks are intact, but the data may not be). It is used by file control to keep track of outstanding recovery work, because whilst the data set still has FLLBs associated with it, special processing rules apply (the actual rules vary with the type of lock condition that has occurred).

FLLBs are created by DFHFCRR (for the lost locks condition, or for an OFFSITE=YES CICS restart), or by DFHFCRO (after a file open which has returned the ‘non-RLS override’ reason code).

FLLBs are chained from both the associated DSNB and the associated FRAB. There is one FLLB per file that held locks per unit of work. Since the FLLB records information about a data set and a unit of work, it contains the DSNB address and the local unit of work ID. It also contains an indicator of the type of lock failure condition that it represents.

FLLBs are getmained from an FLLB subpool above the 16MB line.

File control locks locator blocks are freemained by DFHFCRC at commit time when there are no longer any retained FLABs for the file.

The layout of the FLLB is defined by the DFHFLLB structure and the DFHFLLB DSECT.

File request anchor block (FRAB)
The file request anchor block serves as an anchor for the set of file lasting access blocks (FLABs) belonging to a particular transaction. The file request thread elements (FRTEs) are chained from the FLABs. The FRAB identifies the transaction to which a given file control request belongs.

The FRAB contains pointers to: the next FRAB in the chain from the FC static, the chain of FLABs for this transaction, the chain of FLLBs for the transaction, and any VSWA that has suffered exclusive control conflict for the transaction. The FRAB also contains some indicators related to recovery, such as whether or not the transaction holds RLS locks, whether the unit of work has been through phase 2 of syncpoint, and whether the unit of work has ever been shunted. There is also some
information related to RLS access, including the local unit of work id, a timeout value to be specified on RLS requests, and some problem determination information returned by VSAM RLS when deadlocks occur.

The FRAB is built by DFHFCFR as part of processing of the first File Control request in a transaction. The storage for the FRAB is obtained from a FRAB storage subpool above the 16MB line. The address of the FRAB is then used as the Recovery Manager token associated with the client name ‘FC’. FRABs are rebuilt by DFHFCIR at warm or emergency restart, for units of work which had not completed when CICS terminated. A FRAB is also built if a failure occurs during phase 2 of an intermediate syncpoint. The original FRAB for the transaction is shunted along with the failed parts of the unit of work, and the newly built FRAB is passed on to the next unit of work in the transaction.

If a unit of work is shunted, the FRAB is shunted with it, unless there was no recoverable file control work in the unit of work.

The FRAB is deleted after all the FLABs have been processed during syncpoint at transaction termination. At the same time, the Recovery Manager token is set to zero. At this point, the FRAB storage is not returned to the FRAB storage subpool, but is instead added to a chain of free FRABs, anchored from file control static storage. Subsequent requests to build a FRAB are, if possible, satisfied by a quick cell mechanism from this chain.

Issuing an INQUIRE_WORK_TOKEN call to the recovery manager with client name ‘FC’ returns the address of the file request anchor block for a transaction. There is a chain of all the FRABs in a CICS system, anchored from field FC_FRAB_CHAIN in file control static storage.

The layout of the FRAB is defined by the DFHFRAB structure and the DFHFRAB DSECT.

**File request thread elements (FRTEs)**

FRTEs are used by file control to:
- Represent active file control requests
- Link related requests together as a file thread, for example, the request sequence STARTBR, READNEXT, ..., ENDBR, or READ UPDATE, REWRITE
- Anchor SET storage used for READ SET UPDATE requests and browse requests with the set option, the lifetime of which is that of the request thread.

FRTEs are created by the main file control module, DFHFCFR, and are freed either by DFHFCFR at the end of a request or thread of requests or by the file control recovery control program, DFHFCRC, at syncpoint if this occurs before a thread of requests has completed.

FRTEs for a particular file within a particular task and environment are chained together, and anchored from the FLAB for that file, task and environment.

Storage for FRTEs is acquired from above the 16MB line.

The layout of FRTEs is defined by the DFHFRTE structure and by the DFHFRTE DSECT.
File control

**Keypoint list element (KPLE)**

The keypoint list forms part of file control’s implementation of backup while open (BWO) copy for data sets accessed in non-RLS mode. One KPLE exists for each keypoint and records the start and end times at which tie up records are written.

The KPLE chain is anchored from FC_KPLE_CHAIN in file control static storage.

The keypoint list elements are created, processed and deleted (when they become redundant) by DFHFCRC following RMKP take keypoint calls from the recovery manager. These calls are made whenever a CICS keypoint is taken. KPLEs are getmained from above the 16MB line.

The layout of the KPLE is defined by the KPLE structure.

**Shared resources control (SHRCTL) block**

The SHRCTL block represents the CICS region’s requirements of, and the use made of, a local shared resources pool (LSRPOOL). It is used by DFHFCL when calling VSAM to build an LSRPOOL. It is also used by DFHFCL and statistics programs to hold and update file control statistics. It lasts for the lifetime of a CICS run, and is addressable through a pointer in file control static storage. There are eight pointers collectively named the SHRCTL vector.

A SHRCTL block holds information such as how many virtual and hyperspace buffers of a particular size are needed, how many strings are needed, the maximum key length allowed. CICS passes this information to VSAM when the pool is built. It also holds statistics about the pool which are sent to the statistics domain when requested or when the pool is deleted.

Each SHRCTL block represents one LSRPOOL, and there are eight SHRCTL blocks. The layout of each SHRCTL block is defined by the DFHFCTLs structure and by the DFHFCTSR DSECT, and they reside above the 16MB line.

On a CICS cold start, DFHFCRP performs the following:
- Unconditionally builds eight SHRCTL blocks above the 16MB line (from a SHRCTL block subpool)
- Fills in default settings in the block, or inserts user-specified information
- Catalogs each SHRCTL block in the CICS global catalog (GCD).

On a CICS warm or emergency start:
- DFHFCRP restores all eight SHRCTL blocks from the global catalog.

The contents of a SHRCTL block are decided in one of three ways:
- User defines the contents in the FCT by means of the DFHFCT TYPE=SHRCTL,LSRPOOL=n macro call. This assembled information is used by DFHFCRP on a COLD start only (as per FCT entries).
- User defines the contents online through a CEDA DEFINE LSRPOOL command.
- If neither of the above two methods is used, DFHFCL calculates the contents before calling VSAM to build the LSRPOOL.
**VSAM work area (VSWA)**

The VSWA represents a VSAM request to CICS. Embedded in the VSWA is the request parameter list (RPL) which is passed to VSAM to perform the request. In addition to the RPL, the VSWA contains other CICS information related to the request.

The VSWA is used by DFHFCVS and DFHFCRS when processing VSAM files.

A VSWA survives as long as the RPL needs to survive to complete the VSAM request; for example, it survives from READ UPDATE to the REWRITE request.

The address of the VSWA is held in the file request thread element (FRTE) in the FRT_WORK_AREA_ADDRESS field.

Storage for the VSWA is acquired from above the 16MB line.

The layout of the VSWA is defined by the DFHVSWAS structure and by the DFHVSWA DSECT.

### Modules

This § describes the following modules. Unless otherwise stated, addressing mode and residency mode are AMODE 31 and RMODE ANY respectively.

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
<th>See page</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHEIFC</td>
<td>File control EXEC interface module</td>
<td>565</td>
</tr>
<tr>
<td>DFHEIQCF</td>
<td>Exec INQUIRE CFDTPOOL module</td>
<td>-</td>
</tr>
<tr>
<td>DFHFCAT</td>
<td>File control catalog manager</td>
<td>566</td>
</tr>
<tr>
<td>DFHFCBD</td>
<td>File control BDAM request processor</td>
<td>567</td>
</tr>
<tr>
<td>DFHFCCA</td>
<td>File control RLS control ACB manager</td>
<td>568</td>
</tr>
<tr>
<td>DFHFCDL</td>
<td>File control coupling facility data table load program</td>
<td>569</td>
</tr>
<tr>
<td>DFHFCDN</td>
<td>File control DSNAMEx block manager</td>
<td>569</td>
</tr>
<tr>
<td>DFHFCDO</td>
<td>File control coupling facility data table open/close program</td>
<td>571</td>
</tr>
<tr>
<td>DFHFCDR</td>
<td>File control coupling facility data table request processor</td>
<td>571</td>
</tr>
<tr>
<td>DFHFCDT</td>
<td>File control shared data table request processor</td>
<td>571</td>
</tr>
<tr>
<td>DFHFCDS</td>
<td>File control shared data table function ship program</td>
<td>571</td>
</tr>
<tr>
<td>DFHFCDU</td>
<td>File control coupling facility data table UOW calls program</td>
<td>572</td>
</tr>
<tr>
<td>DFHFCDW</td>
<td>File control coupling facility data table RMC program</td>
<td>572</td>
</tr>
<tr>
<td>DFHFCDY</td>
<td>File control coupling facility data table resynchronization program</td>
<td>572</td>
</tr>
<tr>
<td>DFHFCES</td>
<td>File control ENF servicer</td>
<td>572</td>
</tr>
<tr>
<td>DFHFCFL</td>
<td>File control FRAB and FLAB processor</td>
<td>572</td>
</tr>
<tr>
<td>DFHFCFR</td>
<td>File control file request handler</td>
<td>572</td>
</tr>
<tr>
<td>DFHFCFS</td>
<td>File control file state program</td>
<td>572</td>
</tr>
<tr>
<td>DFHFCIN1</td>
<td>File control initialization program 1</td>
<td>572</td>
</tr>
<tr>
<td>DFHFCIN2</td>
<td>File control initialization program 2</td>
<td>572</td>
</tr>
<tr>
<td>DFHFCIR</td>
<td>File control initialize recovery</td>
<td>572</td>
</tr>
<tr>
<td>DFHFCCL</td>
<td>File control shared resources pool processor</td>
<td>572</td>
</tr>
<tr>
<td>DFHFCCLF</td>
<td>File control log failure handler</td>
<td>572</td>
</tr>
<tr>
<td>DFHFCJ</td>
<td>File control logging and journaling program</td>
<td>572</td>
</tr>
<tr>
<td>DFHFCMT</td>
<td>File control table manager</td>
<td>580</td>
</tr>
<tr>
<td>DFHFCN</td>
<td>File control open/close program</td>
<td>580</td>
</tr>
<tr>
<td>DFHFCNQ</td>
<td>File control non-RLS lock handler</td>
<td>587</td>
</tr>
<tr>
<td>DFHFCOR</td>
<td>File control offsite recovery completion</td>
<td>587</td>
</tr>
<tr>
<td>DFHFCQI</td>
<td>File control RLS quiesce initiation</td>
<td>588</td>
</tr>
</tbody>
</table>
### File control

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
<th>See page</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHFCQR</td>
<td>File control RLS quiesce receive transaction</td>
<td>588</td>
</tr>
<tr>
<td>DFHFCQS</td>
<td>File control RLS quiesce send transaction</td>
<td>588</td>
</tr>
<tr>
<td>DFHFCQT</td>
<td>File control RLS quiesce common system transaction</td>
<td>588</td>
</tr>
<tr>
<td>DFHFCQU</td>
<td>File control RLS quiesce processor</td>
<td>588</td>
</tr>
<tr>
<td>DFHFCQX</td>
<td>File control RLS quiesce exit</td>
<td>588</td>
</tr>
<tr>
<td>DFHFCRC</td>
<td>File control recovery control program</td>
<td>589</td>
</tr>
<tr>
<td>DFHFCRD</td>
<td>File control RLS cleanup transaction</td>
<td>591</td>
</tr>
<tr>
<td>DFHFCRF</td>
<td>File control function shipping interface module</td>
<td>591</td>
</tr>
<tr>
<td>DFHFCRL</td>
<td>File control share control block manager</td>
<td>591</td>
</tr>
<tr>
<td>DFHFCRO</td>
<td>File control RLS open/close program</td>
<td>591</td>
</tr>
<tr>
<td>DFHFCRP</td>
<td>File control restart program</td>
<td>591</td>
</tr>
<tr>
<td>DFHFCRR</td>
<td>File control RLS restart</td>
<td>591</td>
</tr>
<tr>
<td>DFHFCRS</td>
<td>File control RLS record management processor</td>
<td>591</td>
</tr>
<tr>
<td>DFHFCRV</td>
<td>File control RLS VSAM interface processor</td>
<td>591</td>
</tr>
<tr>
<td>DFHFCSD</td>
<td>File control shutdown program</td>
<td>591</td>
</tr>
<tr>
<td>DFHFCST</td>
<td>File control statistics program</td>
<td>591</td>
</tr>
<tr>
<td>DFHFCVR</td>
<td>File control VSAM interface program</td>
<td>591</td>
</tr>
<tr>
<td>DFHFCVS</td>
<td>File control VSAM request processor</td>
<td>591</td>
</tr>
</tbody>
</table>

There are also a number of modules which make up the coupling facility data tables server. These all have names of the form DFHCFxx.

Figure 61 on page 565 shows the main file control modules and their interfaces.
DFHEIFC (file control EXEC interface module)

**Call mechanism**
Kernel subroutine call. Automatic stack storage acquired as part of the call.

**Entry address**
DFHEIFC. Stored in the CSA in a field named CSAEIFC.
File control

Purpose
DFHEIFC is DFHEIP’s file control interface. It routes requests to the file control file request handler, DFHFCFR.

Called by
DFHEIP exclusively.

Inputs
The EIEI parameter list, as defined by the DFHEIEIA DSECT.

Outputs
Updated EIEI parameter list, with completed EIB.

Operation
• Call user exit XFCREQ.
• Call file control request handler DFHFCFR.
• Call user exit XFCREQC.

How loaded
At CICS startup, as part of the building of the CICS nucleus. The nucleus is built by DFHSIB1, which uses its nucleus build list to determine the content and characteristics of the CICS nucleus.

DFHFCAT (file control catalog manager)

Call mechanism
Kernel subroutine call. Automatic stack storage acquired as part of the call.

Entry address
DFHFCAT. The entry point address is held in FC static storage in a field named FC_FCAT_ADDRESS, which is set by DFHFCRP when it loads DFHFCAT.

Purpose
The file control catalog manager is part of the file control component. This program processes inquire and update requests on the state of the backup while open (BWO) attributes in the ICF catalog for VSAM data sets and inquire on the quiesce state in the ICF catalog. The DFSMS Callable Services interface is used for these operations.

Called by
DFHFCDN
Get the base data set name for a DSNB that has not yet been validated, update the recovery point, or to set the BWO attributes to a ‘forward recovered’ state

DFHFCN
Inquire on the current state of, and to update, BWO attributes during file open processing; and to reset these attributes during file close processing.

DFHFCQI
Inquire on the quiesce state of a data set.

Inputs
The FCAT parameter list, as defined by the DFHFCATA DSECT, is created as part of the subroutine call.

The input parameters are:
Data set name
Recovery point

**Outputs**
Returned in the FCAT parameter list:

- Quiesce state
- Base data set name
- State (fuzzy, sharp)
- Response
- Reason

**Operation**
DFHFCAT provides the following functions:

**INQ_BASEDSNAME**
Gets the base data set name for a specified data set name from the ICF catalog. This function is used when there is not a validated DSN block for the data set.

**INQ_CATALOGQUIESCESTATE**
If the level of DFSMS is 1.3 or higher, issues an IGWARLS call to determine the quiesce state of the data set (quiesced or unquiesced).

**INQ_DATASET_STATE**
Determines the current state of a VSAM data set's BWO attributes in the ICF catalog. If the BWO attributes indicate that the data set is "back level", that is, a backup copy has been restored but not forward recovered, an exception response is returned; otherwise, a state of 'fuzzy' or 'sharp' is returned, indicating whether or not the data set is defined in the ICF catalog as eligible for BWO.

**SET_CATALOG_RECOVERED**
Updates a VSAM data set's BWO attributes in the ICF catalog to a 'forward recovered' state to indicate that the data set has been forward recovered.

**SET_CATALOG_RECOV_POINT**
Updates a VSAM data set's BWO attributes in the ICF catalog with the new recovery point.

**SET_BWO_BITS_DISABLED**
Updates a VSAM data set's BWO attributes in the ICF catalog to show that the data set is no longer eligible for BWO support, and updates the recovery point.

**SET_BWO_BITS_ENABLED**
Updates a VSAM data set's BWO attributes in the ICF catalog to show that the data set is eligible for BWO support, and updates the recovery point.

**How loaded**
By DFHFCRP as part of file control initialization.

**DFHFCBD (file control BDAM request processor)**

**Call mechanism**
Kernel subroutine call. Automatic stack storage acquired as part of the call.
Entry address
DFHFCBD. The entry point address is held in FC static storage in a field named FC_BDAM_ENTRY_ADDRESS.

Addressing mode
AMODE 31.

Residency mode
RMODE 24.

Purpose
The BDAM request processor is part of the file control component. It processes access requests to BDAM files.

Called by
DFHFCFR, after having determined that the request is for a BDAM file.

Inputs
The FCFR parameter list, as defined by the DFHFCFRA DSECT. Also, the file control environment, including FC static storage and the FCT.

Outputs
Updated FCFR parameter list.

Operation
Acquires and releases FIOA storage as necessary. Implements BDAM exclusive control requests. Performs record-length and key-length checking. Calls BDAM to perform the I/O request.

Acquires storage, in the correct key subpool, for requests that specify SET.

How loaded
By DFHFCFS, by means of a loader domain call. DFHFCBD is not loaded unless DFHFCFS is called to open a BDAM file and, in doing so, it discovers that DFHFCBD is not yet in storage.

DFHFCCA (file control RLS control ACB manager)
DFHFCCA is the file control RLS control ACB manager. The RLS control ACB is a special ACB required when a commit protocol application such as CICS uses VSAM RLS. FCCA processes requests to register and unregister the control ACB, and all other file control requests to SMSVSAM that have to be made via the control ACB. These requests are:

- IDAREGP (register)
- IDAUNRP (unregister)
- IDARECOV (clear recovery status)
- IDAINQRC (inquire on recovery)
- IDAQUIES (quiesce)
- IDALKREL (release locks, and retain locks marked for retention)
- IDARETLK (mark locks for retention)

DFHFCCA also includes the code for the RLSWAIT exit used by control ACB requests. Whenever CICS issues such a request, VSAM drives the RLSWAIT exit as soon as it is about to transfer control to the SMSVSAM address space. CICS is then able to drive the dispatcher and schedule other CICS tasks whilst the SMSVSAM address space is busy processing the request.
**DFHFCDL (file control CFDT load program)**

DFHFCDL is attached by DFHFCDO to load a load-capable coupling facility data table with records from a source data set.

**DFHFCDN (file control DSN block manager)**

**Call mechanism**
Kernel subroutine call. Automatic stack storage acquired as part of the call.

**Entry address**
DFHFCDN. The entry point address is held in FC static storage in a field named FC_FCDN_ADDRESS, which is set by DFHFCRP when it loads DFHFCDN.

**Purpose**
The DSNAME block manager is part of the file control component. This program is called to perform various operations on data set name blocks. These operations include connecting and disconnecting DSN blocks and FCT entries, setting their attributes, and deleting them when no longer required. The program also allows the caller to inspect a particular DSN block or browse a set of blocks. It can also be called to update the backup while open (BWO) attributes in the ICF catalog for VSAM data sets, and to set the quiesce state to normal in all DSN blocks. Finally it can be called to catalog the information in a DSN block to the CICS global catalog.

**Called by**

**DFHAMFC**
Connect a DSN block to a newly created FCT entry

**DFHAMPFI**
Connect the DSN block for the CSD to the associated FCT entry

**DFHEIQDN**
Connect, disconnect, delete, set attributes, browse, and inquire against DSN blocks in response to external requests; and to update the BWO attributes in the ICF catalog for a VSAM data set to a ‘forward recovered’ state

**DFHEIQDS**
Connect or disconnect DSN blocks and FCT entries in response to external requests

**DFHFCLF**
Set the availability attribute to unavailable after a forward recovery log stream failure

**DFHFCMT**
Disconnect the DSN block when deleting an FCT entry

**DFHFCN**
Connect or disconnect and to catalog a DSN block

**DFHFCRC**
Update the recovery point in the ICF catalog for all VSAM data sets that are open for update in non-RLS mode and defined as eligible for BWO support at keypoint time

**DFHFCRD**
To reset all quiesce states to normal after an SMSVSAM server failure

**DFHFCRO**
Connect or disconnect and to catalog a DSN block

**DFHFCRP**
Connect or reconnect DSN blocks during file control initialization or restart.
**File control**

**Inputs**
The FCDN parameter list, as defined by the DFHFCDNA DSECT, is created as part of the subroutine call.

The input parameters include:

- Request identifier
- Address of FCTE or FCTE token
- Data set name
- Browse token
- Availability status
- Type of pointer
- Recovery point

**Outputs**
Output parameters, as part of the FCDN parameter list. Apart from the response, all these are returned on the inquire or browse requests. The parameters include:

- Access method
- Base data set name
- Availability status
- DSNB type
- File count
- DSNB valid status
- Lost locks status
- Forward-recovery log stream name
- Forward-recovery log ID
- Recovery status
- Response
- Reason

**Operation**

- **Connect:**
  
  The inputs are a data set name and an FCTE pointer or an FCTE token, with an indication of whether the entity to be connected is a base or an object.

  If the FCT entry is already connected, the connection is broken before connecting it to a DSN block representing the new object. The DSN block that is connected can exist already, or DFHFCDN creates a new block before connecting it.

  The request is rejected if it requires an existing connection to be broken, and there are uncommitted updates to the file; that is, there are retained locks.

- **Disconnect:**

  The connection between the FCT entry and the DSN block is broken. The DSN block remains even if there are no other FCT entries connected to it. The request is rejected if there are uncommitted updates to the file: that is, there are retained locks.

- **Delete:**

  Checks are made to ensure that the DSN block is allowed to be deleted. If the deletion can proceed, the table manager is called to delete the DSN from the DSN index, and the storage domain is called to free the storage.

- **Inquire:**

  The attributes stored in the DSN block are returned to the caller in the FCDN parameter list.

- **Set:**
The availability status is set in the DSN block. The catalog domain is called to catalog the change.

- **Start browse, get next, end browse:**
  
The DSN blocks are browsed in order. For each, the attributes are returned to the caller.

- **Catalog:**
  
The information in a DSN block is cataloged to the CICS global catalog.

- **SET_CATALOG_RECOVERED:**
  
  This function is used by DFHEIQDN. DFHFCDN in turn issues a SET_CATALOG_RECOVERED call to DFHFCAT to update the BWO attributes in the ICF catalog for a given VSAM data set to a 'forward recovered' state.

- **UPDATE_RECOVERY_POINTS:**
  
  This function is used by DFHFCRC. DFHFCDN in turn issues a SET_CATALOG_RECOV_POINT call to DFHFCAT to update the recovery point in the BWO attributes in the ICF catalog for every data set that is open for update in non-RLS mode and defined as eligible for BWO support.
  
The recovery point is the time from which a forward-recovery utility should start applying log records. It is always before the time the last backup was taken. For further information about recovery points and backup while open in general, see the CICS Recovery and Restart Guide.

- **RESET_ALL_QUIESCE_STATUS:**
  
  This function is used by DFHFCRD. The DSNB table is scanned, and the quiesce status is reset to normal in each DSNB.

**How loaded**

By DFHFCRP as part of file control initialization.

**DFHFCDO (file control CFDT open/close program)**

When called using the FCFS parameter list, DFHFCDO performs an equivalent function for coupling facility data table opens and closes as is performed by DFHFCN for non-RLS VSAM files.

When called using the FCDS parameter list, DFHFCDO performs statistics collection for coupling facility data tables, and disconnects from CFDT pools at shutdown.

**DFHFCDR (file control CFDT request processor)**

DFHFCDR performs an equivalent function for coupling facility data tables as is performed by DFHFCVS for non-RLS VSAM files, and uses the same interface.

**DFHFCDTDS (file control shared data table request program)**

DFHFCDTDS performs an equivalent function for CICS-maintained and user-maintained data tables as is performed by DFHFCVS for non-RLS VSAM files and uses the same interface.

**DFHFCDTX (file control shared data table function ship program)**

DFHFCDTX receives file requests from DFHFCDTS in FCFRR format, converts them into command level interface form and then calls ISP to function ship the request.
The response returned by ISP in the EIB is translated back into an FCFRR response and reason code.

**DFHFCDU (file control CFDT UOW calls program)**

DFHFCDU encapsulates the processing required to call the coupling facility data tables server for unit of work related operations, such as commit, backout, inquire. It is called via the FCDU parameter list by DFHFCDW and DFHFCDY.

**DFHFCDW (file control CFDT RMC program)**

DFHFCDW provides a recovery manager connector (RMC) between file control and the coupling facility data tables server, to support 2-phase commit and recovery for recoverable coupling facility data tables. It is called by the CICS Recovery Manager using the RMLK parameter list.

**DFHFCDY (file control CFDT resynchronization program)**

DFHFCDY performs resynchronization of coupling facility data table pools and links. It is called using the FCDY parameter list by DFHFCDO, DFHFCDR and DFHFCDU.

**DFHFCES (file control ENF servicer)**

DFHFCES is the file control ENF servicer. It is used to prompt dynamic restart of RLS file control when the SMSVSAM Server becomes available again after an earlier failure. DFHFCES is invoked whenever the MVS Event Notification Facility notifies CICS (via the CICS domain manager ENF support) that SMSVSAM is available.

DFHFCES establishes a transaction environment, and calls DFHFCRR to dynamically restart RLS.

**DFHFCFL (file control FRAB and FLAB processor)**

DFHFCFL is the File Control FRAB/FLAB processor. It contains a number of functions to process FLAB control blocks belonging to a particular base data set. It processes the functions of the FCFL interface.

The DSNB of the data set is not locked during the processing of the commands. As a FLAB exists, and hence an FCTE, the DSNB cannot be deleted, therefore there is no need to lock the DSNB.

**DFHFCFR (file control file request handler)**

**Call mechanism**

Kernel subroutine call. Automatic stack storage acquired as part of the call.

**Entry address**

DFHFCFR. Stored in the CSA in a field named CSAFCEP.

**Purpose**

The central module in the file control component.

Processes file control requests issued by DFHEIFC (requests from application programs), or from other CICS modules (internal CICS file control requests).

Receives and routes file control access-method dependent requests to one of the following:
• DFHFCRS for VSAM RLS files
• DFHFCVS for VSAM non-RLS files
• DFHFCBD for BDAM files
• DFHFCDR for coupling facility data tables
• DFHFCSTS for user-maintained data tables
• DFHFCDTS for non-update requests to CICS maintained data table
• DFHFCVS for update requests to CICS-maintained data tables
• DFHFCRF for requests to remote files

Implements TEST_FILE_USER requests.

Routes RESTART_FILE_CONTROL requests to DFHFCVS and DFHFCRS during the file control initialization.

Frees buffers at the request of DFHAPSM when ‘short on storage’ has been detected.

Performs a CLEAR_ENVIRONMENT when requested by DFHERM, DFHAPLI or DFHUEH. This cleans up file control storage at the completion of a task-related user exit, a URM, or a global user exit:
• The FLAB and FRTE chain are scanned to find all FRTEs for the specified environment.
• An ENDBR request is issued to terminate any active browse operation.
• An UNLOCK request is issued for any active READ UPDATE or WRITE MASSINSERT.

Called by

DFHAPLI
  AP language interface program
DFHAPSM
  AP domain storage notify gate
DFHDMPCA
  CSD manager adapter
DFHDTLX
  Shared data tables load program
DFHEIFC
  File control EXEC interface module
DFHERM
  Resource manager interface (RMI) module
DFHFCDL
  Coupling facility data tables load program
DFHFCDTS
  File control shared data table request processor
DFHFCFR
  File control file request handler (a recursive call)
DFHFCRC
  File control recovery control program
DFHFCRP
  File control restart program
DFHUEH
  AP user exit handler.
Inputs
The FCFR parameter list, as defined by the DFHFCFRA DSECT. Also the file control environment, including FC static storage and the FCT.

Outputs
Updated FCFR parameter list.

Operation
Selects on the request type, and passes control to the routine specific to that request.

Performs monitoring.

Obtains a FLAB and FRTE to represent this request, or scans the FLAB and FRTE chains to associate this request with a previous FRTE if required. Some checking for error situations is performed during the scan.

Performs file state checking to determine whether or not a (VSAM or BDAM) request to a file is able to proceed. If file is enabled but closed and is not a request to a remote file, opens it before carrying out the request.

Checks for “privileged” requests.

If the request is not remote, checks the “service request” attributes for the file to determine whether the request can proceed.

Checks the file’s access method (VSAM or BDAM as defined in the FCT). If BDAM, calls DFHFCBD to process the request. If VSAM and non-RLS, calls DFHFCVS to process the request. If VSAM and RLS, calls DFHFCRS to process the request. If a data table, calls DFHFCDTS for read requests against a CICS-maintained data table or any request against a user-maintained table, and calls DFHFCVS otherwise (that is, for update and browse requests against a CICS-maintained data table). If the file is remote, calls DFHFCRF to process the request.

On return, performs cleanup if required.

How loaded
By DFHSIB1 as part of the CICS nucleus.

DFHFCFS (file control file state program)

Call mechanism
Kernel subroutine call. Automatic stack storage acquired as part of the call.

Entry address
DFHFCFS. The entry point address is held in FC static storage in a field named FC_FCFS_ADDRESS, which is set by DFHFCRP when it loads DFHFCFS.

Purpose
The file control file state program is part of the file control component.

The program processes requests to enable, disable, open, and close files. Such requests can originate from explicit requests (either CEMT or EXEC CICS SET), from implicit requests (such as implicit open), or from requests made from CICS internal processing.
Close and disable requests are processed in different ways, depending on whether the request has been issued with the WAIT or the NOWAIT option. A request with the WAIT option is treated as a synchronous request, that is, control returns to the requesting program only after all users of the file have completed their use.

A request with the NOWAIT option is treated as an asynchronous request. In this case, the file is marked with the intended state and control is returned immediately.

**Called by**

- **DFHAMFC**
  - Enable a newly installed file
- **DFHDMPCA**
  - Change the state of the CSD
- **DFHDMRM**
  - Close CSD after an error
- **DFHDTLX**
  - Close the data set associated with a shared data table
- **DFHEIQDS**
  - Implement CEMT and EXEC CICS requests
- **DFHFCDL**
  - Close the data set associated with a coupling facility data table
- **DFHFCDTS**
  - Close shared data table if remote connection disabled or invalidated
- **DFHFCFR**
  - Implicit open
- **DFHFCQU**
  - Close files for quiesce, cancel close for unquiesce, enable files
- **DFHFRCR**
  - Open files which need backout, and close files at syncpoint
- **DFHFRCR**
  - Immediate close of RLS files
- **DFHFCRD**
  - Close files for pending immediate close requests
- **DFHFCSD**
  - Close files on a normal CICS shutdown
- **DFHFSC**
  - Open all files with FILSTAT=OPEN coded
- **DFHFCVS**
  - Open the base, and during empty file or I/O error processing.

**Inputs**

The FCFS parameter list, as defined by the DFHFCFSA DSECT, is created as part of the subroutine call.

The input parameters are:

- Request identifier (open, close, enable, disable, cancel close)
- FCTE address
- FCTE token
- Open options (open base, open for backout)
- Close qualifier (close pending, shutdown, immediate close, quiesce, and so on)
- Action (wait, do not wait, force)

**Outputs**

Returned in the FCFS parameter list:
**DFHFCN return code**
**Register 15 return code**
**VSAM return code**

**Operation**
Before any processing to change the state of a file is carried out, its FCT entry is locked by means of a DFHKC ENQ call. At the conclusion of file state change processing, the FCT entry is unlocked before returning to the caller.

- **Enable file.**
  DFHFCFS marks the FCT entry as ‘enabled’, and catalogs the change.

- **Disable file.**
  If the WAIT option is specified, DFHFCFS tests whether the transaction issuing the request is a current user of the file. If it is, DFHFCFS returns an exception response.
  DFHFCFS next marks the FCT entry entry as ‘disabled’ and catalogs the change. If the disable request stems from a close request (see later), DFHFCFS also sets the implicit indicator, thereby marking the state as ‘unenabled’. However, if this close request originated from DFHFCSD as part of CICS shutdown processing, DFHFCFS does *not* mark the state as ‘unenabled’.
  Finally, if the WAIT option is specified, the FCT entry is unlocked before waiting for the ‘disabled’ ECB in the FCT entry to be posted by the transaction that reduces the use count to zero.

- **Open file.**
  If the file is unenabled (due to a previous close), DFHFCFS enables it and catalogs the new state, unless the open option is open for backout.
  If the file refers to a BDAM data set, DFHFCFS tests whether DFHFCBD is already loaded; if not, it calls loader domain to do so.
  If the file is a data table, DFHFCFS loads and initializes data table services, if this has not been done already on a previous open request.
  DFHFCFS next calls DFHFCN (for non-RLS) or DFHFCRO (for RLS) to perform the physical open. After the file has been successfully opened, its FCT entry is marked accordingly.
  For a data table, DFHFCFS issues OPEN and LOAD requests to data table services.

- **Close file.**
  If there is no close qualifier, the file is first implicitly disabled (as described above), taking into account the WAIT or NOWAIT option. The new state is cataloged.
  If the file use count is zero, DFHFCFS calls DFHFCN or DFHFCRO to perform the physical close. After the file has been successfully closed, its FCT entry is marked accordingly.
  An immediate close is issued if the SMSVSAM RLS server fails. The close must wait until there are no requests active in the RLS record management processor. The enablement state of the file is not changed. A close with close qualifier of quiesce is issued to process an RLS quiesce request. The file is unenabled, and the state cataloged.
  For a data table, DFHFCFS issues a CLOSE request to data table services, except in the case of a special type of CLOSE request issued by DFHFCVS for a user-maintained data table, when loading is complete and the source data set is to be closed, but not the table itself.
  For a remote data table, DFHFCFS issues a DISCONNECT request to data table services.
If the file use count is nonzero, DFHFCFS sets the ‘close requested’ indicator in the FCT and returns to the caller. Any subsequent transaction that reduces the use count to zero tests the ‘close requested’ indicator and, if set, performs the actual close.

When called by DFHFCSD during CICS shutdown, DFHFCFS ensures that files are closed, marks the file as ‘closed unenabled’ in the FCT, but does not record this change in the global catalog. This allows implicit file opens on a subsequent restart.

- Cancel close.
  
  An in-progress close is cancelled if a data set is unquiesced. The close_in_progress flag is reset, any tasks waiting for the file to close are resumed, and the file is re-enabled.

**How loaded**

By DFHFCRP as part of file control initialization.

### DFHFCIN1 (file control initialization program 1)

**Call mechanism**

Kernel subroutine call. Automatic stack storage acquired as part of the call.

**Entry address**

DFHFCIN1. Stored in the CSA in a field named CSAFCXAD.

**Purpose**

The file control initialization program is part of the file control component. This program initializes file control and starts the file control restart task. It also waits for the restart task to complete, and returns the status of the completion to the caller.

**Called by**

DFHSII1, as part of CICS initialization.

**Inputs**

The FCIN parameter list, as defined by the DFHFCINA DSECT.

**Outputs**

Updated FCIN parameter list.

**Operation**

**Initialize:**

- Calls storage manager domain to add a subpool for file control static storage.
- Calls storage manager domain to create the storage for file control static storage.
- Initializes file control static storage.
- Attaches the file control restart task by means of a DFHKC request, with entry point address DFHFCIN2.

**WAITINIT:**

- Issues a dispatcher domain call to wait on the CICS ECB which indicates that the file control restart task has finished (FC_RECOV_ALLOWED_ACB) in file control static storage.
- On completion of the wait, tests the response and returns to DFHSII1.

**How loaded**

Link-edited with DFHFCIN2 to form the DFHFCIN module, which is loaded by DFHSIB1 as part of the CICS nucleus.
DFHFCIN2 (file control initialization program 2)

Call mechanism
Attached by DFHFCIN1 as a separate CICS task. Given control by means of the DFHKC TYPE=ATTACH mechanism.

Entry address
DFHFCIN2. Because DFHFCIN2 is link-edited with DFHFCIN1, the entry address is known to DFHFCIN1 at the time the DFHKC TYPE=ATTACH is issued.

Purpose
The file control initialization program is part of the file control component. This program loads and calls the file control restart program (DFHFCRP), to perform file control restart as a separate task.

Called by
CICS task control, after being attached by DFHFCIN1.

Inputs
None.

Outputs
The initialized file control component. Addresses and indicators completed in file control static storage.

Operation
Calls loader domain to acquire (that is, to load) the DFHFCRP program. Stores the entry point address of the loaded module (which is also the load point) in DFHFCIN2’s automatic storage in a field named FCRP_ENTRY_ADDRESS.

If the ACQUIRE request failed, calls loader domain to define program and then retries the ACQUIRE request.

Calls DFHFCRP by means of a subroutine call via the kernel.

On successful completion, calls loader domain to release DFHFCRP. On both successful and unsuccessful completion, posts the ECBs FC_NON_RECOV_ALLOWED_ECB and FC_RECOV_ALLOWED_ECB. The success or otherwise of File Control restart is indicated by the flag FCSCMPLT in file control static storage.

On unsuccessful completion, posts the Restart Task ECB complete and returns.

How loaded
By DFHSIB1 as part of the CICS nucleus.

DFHFCIR (file control initialize recovery)

DFHFCIR is the File Control Initialize Recovery Module. It initializes the File Control environment in which recovery after a CICS failure is carried out.

DFHFCIR handles the delivery of recovery data by the CICS Recovery Manager during its scan of the system log at warm or emergency restart, and rebuilds the file control structures that represent units of work that were in-flight or shunted when CICS terminated.

During its log scan, Recovery Manager calls File Control’s recovery gate, which invokes the module DFHFCRC. DFHFCRC passes the calls through to DFHFCIR.
via a kernel subroutine call. The calls are the RMDE functions START_DELIVERY,
DELIVER_RECOVERY, DELIVER_FORGET and END_DELIVERY.

DFHFCL (file control shared resources pool processor)

**Call mechanism**
BALR, obtaining LIFO storage on entry.

**Entry address**
DFHFCLNA. DFHFCL is, together with DFHFCN and DFHFCM, link-edited with
DFHFCFS. All calls to DFHFCL are made from DFHFCN; the entry point address
is known to DFHFCN from the link edit.

**Purpose**
The shared resources pool processor is part of the file control component.

This program is called at file open time to create a specific local shared resources
pool if it does not exist. It is also called to delete a specific pool when the last file
to use the pool is being closed.

The size and characteristics of the pool being built are obtained either from
information in the SHRCTL definition in the FCT or, if that information has not
been provided, from the best information available to DFHFCL at the time of the
open.

**Called by**
DFHFCL is called exclusively by DFHFCN.

**Inputs**
The FCLPARAM parameter list, created in DFHFCN’s automatic storage and
addressed by register 1 on the call.

The input parameters are:

- Request identifier (build, delete)
- LSR pool number

**Outputs**
Returned in the FCLPARAM parameter list:

- DFHFCL return code
- BLDVRP/DLVRP return code
- VSAM return code

**Operation**
If the request is for LSR pool creation, DFHFCL first checks whether the SHRCTL
block includes specifications for the number of strings, maximum key length, and
the number of virtual and hyperspace buffers of each of the eleven sizes in the
pool. If these values are known, DFHFCL sets up the BLDVRP parameter list and
creates the pool by issuing the BLDVRP macro.

If some or all of the pool characteristics are not specified in the SHRCTL definition,
DFHFCL calculates the pool requirements from the information in the FCT and the
VSAM catalog.

Each FCT entry is inspected to find whether it is to be included in the pool being
built. If so, its DSNAME is determined and this is used to obtain data set
File control

characteristics from the VSAM catalog. The information required for the BLDVRP macro is accumulated in the SHRCTL block and the pool is built from these values.

If the request is for LSR pool deletion, DFHFCL first obtains the VSAM statistics for the pool and saves them in the SHRCTL block. These statistics are unobtainable after the pool has been deleted.

DFHFCL next deletes the specified pool by issuing a DLVRP macro.

Finally, DFHFCL sends pool statistics to the statistics domain as unsolicited data.

How loaded
As a constituent part of DFHFCFS, which is loaded by DFHFCRP as part of file control initialization.

DFHFCLF (file control log failures handler)
DFHFCLF provides control of long term logger failures for File Control. It is called in the event of a failure of a general log stream, which will be either the forward recovery log for a data set or the autojournal for a file.

The CICS Log Manager invokes DFHFCLF when an MVS log stream being used for forward recovery or file autojournalling suffers a long term failure. The call is made using the LGGL ERROR function.

When file control opens a forward recovery log stream or an autojournal, it will register this call back gate to the Log Manager by specifying FCLF as the file control error gate.

When called, DFHFCLF takes action to ensure that the log stream failure causes minimum damage. For a forward recovery log failure it closes all files open against the data set using that forward recovery log (across the sysplex for a data set accessed in RLS mode) and issues a message advising that a new backup copy should be taken. For an autojournal it closes the file using that autojournal and issues a warning message.

DFHFCLJ (file control logging and journaling program)
DFHFCLJ is the file control logging and journaling program. It is called to perform logging for transaction backout and forward recovery, to write to journals for autojournal requests and to write to the log of logs.

Records are written to the system log using the RMRE APPEND function, and optionally forced using the RMRE FORCE function. Records are written to forward recovery logs and autojournals using the LGGL WRITE function, and to the log of logs using the LGGL WRITE_JNL function.

DFHFCMT (file control table manager)

Call mechanism
Kernel subroutine call. Automatic stack storage acquired as part of the call.

Entry address
DFHFCMT. The entry point address is held in FC static storage in a field named FC_FCMT_ADDRESS, which is set by DFHFCRP when it loads DFHFCMT.
Purpose
The file control table manager is part of the file control component. This program is called to add, delete, and set FCT entries, and to return attributes of an FCT entry (inquire).

Called by
DFHAMFC
Inquire on, add, or delete a newly created FCT entry to the system
DFHAMPFI
Add the entry in the FCT for the CSD to the system
DFHDMPCA
Inquire on and set the attributes of the FCT entry for the CSD
DFHEDFX
Inquire on the attributes of an FCT entry
DFHEIQDS
Inquire on or set the attributes of FCT entries, or delete an FCT entry.

Inputs
The FCMT parameter list, as defined by the DFHFCMTA assembler DSECT, is created as part of the subroutine call.

The input parameters are:

Common parameters:
- File name
- String number
- Journal ID
- Recovery characteristics
- Journaling characteristics
- Enablement status
- Open time
- Data set disposition
- Service request attributes
- Record format
- Number of data buffers
- Number of index buffers
- Whether to catalog the FCT entry

VSAM-specific parameters:
- VSAM password
- Empty status
- Data set name sharing
- LSR pool ID
- Base name
- Forward recovery log ID
- BWO eligibility
- RLS access mode
- Read integrity

BDAM-specific parameters:
- Exclusive control

Outputs
Output parameters, as part of the FCMT parameter list. Apart from the response, all these are returned on the inquire or browse requests. The output parameters are:
File control

Common parameters:
- File type
- String number
- Record size
- Key length
- Key position
- Recovery characteristics
- Journaling characteristics
- Enablement status
- Open status
- Open time
- Data set type
- Data set disposition
- Data set name
- Base data set name
- Service request attributes
- Record format
- Block format
- Access method
- Remote name
- Remote system

VSAM-specific parameters:
- VSAM password
- Empty status
- Object type
- Data set name sharing
- Number of data buffers
- Number of index buffers
- Number of active strings
- LSR pool ID
- Whether using shared resources
- Forward-recovery log ID
- RLS access mode
- Read integrity

BDAM-specific parameters:
- Block size
- Block key length
- Relative address form
- Exclusive control
- Response
- Reason

Data Table specific parameters:
- Table type
- Table size

**Operation**

- **Add:**
  Storage for the new FCT entry is obtained out of the VSAM FCT storage subpool (BDAM FCT entries cannot be created).
  The new FCT entry is completed by filling in the information from the caller’s parameter list.
  The name of the new FCT entry is added to the TMP index.
Finally the information in the new entry is written to the CICS global catalog if required.

- **Delete:**
  - The request is rejected if there are uncommitted updates for the file; that is, there are retained locks. DFHTMP is called to locate and quiesce the FCT entry.
  - Any DSN block that is connected to the FCT entry is disconnected.
  - The FCT entry name is deleted from the TMP index.
  - The storage for the FCT entry is freed. In the case of a BDAM FCT entry, its DCB storage is also freed.
  - Any catalog entries for the FCT entry are deleted.

- **Set:**
  - DFHTMP is called to locate the FCT entry.
  - The request is rejected if there are uncommitted updates for the file; that is, there are retained locks.
  - If the FCT entry is not marked ‘closed’ and ‘disabled’ (or ‘unenabled’), the request is rejected.
  - Changes are made to the information in the FCT according to the caller’s parameter list.
  - Finally the changes are recorded by writing them to the CICS global catalog.

- **Inquire:**
  - DFHTMP is called to locate the FCT entry.
  - The attributes are returned in the FCMT parameter list.

- **Connect:**
  - DFHTMP is called to locate the FCT entry.
  - The connect count is incremented. The FCT token is returned to the caller.

- **Disconnect:**
  - DFHTMP is called to quiesce the FCT entry.
  - A check is made to ensure that the file is closed and disabled (or unenabled). If the check fails, an error is returned to the caller.
  - The connect count in the FCT is cleared and a call is again made to DFHTMP to release the quiesce.

**How loaded**

By DFHFCRP as part of file control initialization.

**DFHFCN (file control open/close program)**

**Call mechanism**

BALR, obtaining LIFO storage on entry.

**Entry address**

DFHFCNNA. DFHFCN is link-edited with DFHFCFS. All calls to DFHFCN are made from DFHFCFS; the entry point address is known to DFHFCFS from the link-edit.

**Purpose**

The file control open/close program is part of the file control component.
File control

This program performs the physical opening and closing of files by making the corresponding requests to VSAM or BDAM. Associated with these operations are a number of further activities that must be completed before control is returned to DFHFCFS.

These activities include:
- Dynamic allocation of the file
- Empty file checking
- Dynamically setting up ACB fields in advance of the VSAM open
- Copying into file-control control blocks VSAM information about the file which is available after the open
- Inquiring on, and updating, the VSAM data set’s backup while open (BWO) attributes in the ICF catalog for a file that is defined in the FCT as eligible for BWO support if the appropriate prerequisite software levels have been installed
- On close, deallocating the file if necessary and clearing the file control information related to the file
- Resetting a VSAM data set’s BWO attributes in the ICF catalog during close processing.

Called by
DFHFCFS, exclusively.

Inputs
The FCSPARMS parameter list, created in DFHFCFS’s automatic storage and addressed by register 1 on the call.

The input parameters are:
- FCTE address
- Request identifier

Outputs
Returned in the FCSPARMS parameter list:
- DFHFCN return code
- Register 15 return code
- VSAM return code
- Base data set name
- Recovery attributes of base

Operation
Execution of the DFHFCN code is serialized. This is done by DFHFCFS issuing a DFHKC ENQ before calling DFHFCN, and a DFHKC DEQ after calling DFHFCN. As a consequence, only a single open or close request to any file can be in progress at any time, and multiple concurrent requests are single-threaded.

The main actions when processing an open request:
1. If the file is being opened for update and any type of autojournaling is specified on the file definition, then the autojournal log stream is opened, via a call to DFHLGGL.
2. The file is tested to determine if it is allocated to the job by means of a JCL statement or is to be allocated dynamically.
If the file is already allocated, any existing DSN block to which it may be
connected is disconnected and a new block with the actual DSNAME is
connected. Connecting and disconnecting of DSNAME blocks is always
performed by calling DFHFCDN.

If the file is not already allocated, it is at this point dynamically allocated to
the DSNAME in the DSNAME block to which it is connected.

In the case of a VSAM file, the file’s data set name is used to issue
appropriate SHOWCAT and LOCATE instructions to determine relevant
information from the VSAM catalog about the data set that the file represents.
In particular, the following are obtained:

- Base/path indicator
- Base data set name
- Attributes of the data set
- Key length of the base
- Relative key position of base key
- Maximum record length
- Control interval size
- Share options
- High RBA

3. The data set is checked to determine if it is empty (high RBA is zero) or is to
be emptied.

   The ‘load’ mode indicator is set on.

4. DFHFCDN is now called to connect the FCT entry to a DSNAME block for
the base cluster (which may be the existing allocation DSNAME block, or may
need to be newly created, or may already exist and need only be pointed to
from the FCT). The base cluster’s attributes, as obtained from the VSAM
catalog, are stored in the base cluster block.

   The file’s recovery characteristics are checked against any that may already
have been stored in the base cluster block and, if they have not yet been set
up, are saved there. Any conflict with the stored values is handled. In some
cases the new value overrides the old one, in others an error is returned.

   During this processing, if this is the first open for update for a file associated
with this particular data set:

   a. a call is made to the VSAM callable interface IGWARLS, in order to get
      any recovery attributes that may be defined in the VSAM catalog. If they
      are present, then they override any values in the FCT entry.

   b. if forward recovery logging is specified, the forward recovery log stream is
      opened, using either the log stream name from the VSAM catalog, or a log
      stream name derived from the id specified in the file definition.

In the case of an entry sequenced data set or a path to an ESDS, the next
available RBA in the data set is determined and stored in the base cluster
block.

5. If the file uses a shared resources (LSR) pool, and if the pool is not currently
in existence, DFHFCL is called to determine the pool’s characteristics and to
build it.

6. Before opening a VSAM file, any STRNO, BUFND, or BUFNI parameters that
may have been specified in the JCL DD statement are copied to the FCT entry
(for LSR opens, these are ignored). The ACB is now created and its various
options and parameters filled in from information in the FCT entry. The
OPEN is finally completed by a call to VSAM.
File control

7. If the file refers to a BDAM data set, the assembled DCB is used for the open request and no dynamic setting of DCB options is carried out.

8. After the VSAM file has been successfully opened, certain file attributes are obtained from VSAM and are stored in the FCT entry. These include:

   - Key length
   - Relative key position
   - Base/path/AIX indicator
   - KSDS/ESDS/RRDS/VRRDS indicator
   - Number of strings required for an update operation.

9. For a file opened for update against a VSAM base data set when the update use count in the DSNB for this data set is zero, the BWO attributes in the ICF catalog are validated to find their current state. This is done by making an INQ_DATASET_STATE call to DFHFCAT, regardless of whether the file is defined in the FCT as eligible for BWO support.

   The file open request is rejected if one of the following is true:
   a. The BWO attributes in the ICF catalog show either that the data set is “back level”, that is, a backup copy has been restored but not forward recovered, or that either the catalog or the data set has been corrupted.
   b. The BWO attributes in the FCT entry conflict with those defined in the DSNB, that is, the file has already been opened with different attributes since the DSNB was created.

   If the file is defined in the FCT as eligible for BWO support, the BWO attributes in the ICF catalog are updated by making a SET_BWO_BITS_ENABLED call to DFHFCAT.

   However, if the file is not defined in the FCT as eligible for BWO support, but the BWO attributes in the ICF catalog currently show that the VSAM base data set is eligible for BWO support, the BWO attributes in the ICF catalog are disabled by making a SET_BWO_BITS_DISABLED call to DFHFCAT, and CICS issues a warning message.

   **Note:** The ICF BWO attributes are a property of a VSAM sphere; therefore, the VSAM base data set and alternate index path definitions should be consistent. For a general description of the CICS backup while open (BWO) facility, see the [CICS Recovery and Restart Guide](#).

10. The base DSNB, and path DSNB if this is a path, are marked as validated and catalogued.

The main actions when processing a close request:

1. If the close request is for the last file that was opened for update against a VSAM base data set and the file is defined in the FCT as eligible for BWO support, the BWO attributes in the ICF catalog are reset so that BWO support is no longer enabled. This is done by making a SET_BWO_BITS_DISABLED call to DFHFCAT.

2. Before performing the access method close for a VSAM file, the number of accumulated EXCPs is obtained by making a call to VSAM and is saved in the FCT entry ready to be sent to the statistics domain as part of the file statistics.

3. A CLOSE request is then made by issuing the appropriate (VSAM or BDAM) macro.

4. The ACB storage is freed, and certain fields in the FCT entry which are no longer valid are cleared.
5. File statistics and data table statistics, if any, are sent to the statistics domain as unsolicited data.

6. If the file being closed uses shared resources, and if it is the last to have been closed in its LSR pool, DFHFCL is called to delete the pool.

7. If the file was dynamically allocated at open time, it is deallocated, leaving a pointer to the DSNAME block in the FCT entry.

8. If the file had an autojournal, then the autojournal log stream is closed.

9. If the base data set was forward recoverable, and its use count is non-zero, then the forward recovery log stream is closed.

How loaded
As a constituent part of DFHFCFS, which is loaded by DFHFCRP as part of file control initialization.

DFHFCNQ (file control non-RLS lock handler)
DFHFCNQ is the file control non-RLS lock handler. It is called using the FCCA RETAIN_DATASET_LOCKS interface to retain locks in cases of backend failure. It is called using the NQNQ INTERPRET_ENQUEUE interface to interpret File Control locks for presentation purposes.

Lock retention
When DFHFCRC encounters a failure during an attempt to backout a unit of work it must retain all record locks held by that UOW for the failing data set. It issues an FCCA RETAIN_DATASET_LOCKS request to DFHFCCA for RLS access data sets and to this DFHFCNQ for non-RLS access data sets.

Lock name interpretation
Non-RLS locks include record locks for all file types, and for VSAM files, mass-insert range locks, load mode locks and ESDS WRITE locks. Each lock belongs to one of some half dozen or so pools created by DFHFCRP during CICS initialization. DFHFCNQ is called using the NQNQ INTERPRET_ENQUEUE interface and is passed the enqueue pool name and the lock identifier. The name of pool to which a lock belongs is sufficient information to allow the identifier to be parsed and its constituents returned to the caller.

The pool names and lock constituents are:
- FCDSREC - Data set name and record identifier - for VSAM and CICS-maintained data tables
- FCFLREC - File name and record identifier - for BDAM and user-maintained data tables
- FCDSRNGE - Data set name and record identifier - VSAM range locks
- FCDSLMD - Data set name - VSAM load mode locks
- FCDSESWR - Data set name - VSAM ESDS WRITE locks
- FCFLUMTL - File name - UMT load locks

DFHFCOR (file control offsite recovery completion)
DFHFCOR is the file control RLS offsite recovery completion transaction.

Transaction CFOR is attached when CICS detects that it has completed its RLS offsite recovery processing. RLS offsite recovery is only performed when OFFSITE=YES is specified as a system initialization override. CFOR may be attached either during RLS warm or emergency restart (if there is no RLS offsite
recovery work to be performed) or during file control commit processing (if the commit was for the last remaining item of RLS offsite recovery work).

DFHFCOR issues message DFHFC0575 and awaits an operator reply. When the reply is received, it enables RLS access for new transactions.

**DFHFCQI (file control RLS quiesce initiation)**

DFHFCQI is the RLS Quiesce Initiation module. It provides code to initiate a quiesce request against a base data set. It also provides code to inquire on the quiesce state of a base data set, and to complete a quiesce request against a base data set. Quiesce initiations are issued by the CICS API, or by CICS internally, or by CICS internally cancelling certain in-progress quiesce operations. Quiesce inquiries are issued via the CICS API. Quiesce completions are issued by CICS internally.

**DFHFCQR (file control quiesce receive transaction)**

DFHFCQR is the VSAM RLS Quiesce Receive module, running under a dedicated CFQR system transaction. It provides code to take quiesce requests from the CICS VSAM RLS quiesce exit and pass them to DFHFCQU for processing. As DFHFCQR runs under a system transaction, it has full transaction environment which enables it to invoke API-capable global user exits, or to call parts of file control that reference the TCA.

**DFHFCQS (file control RLS quiesce send transaction)**

DFHFCQS is the VSAM RLS Quiesce Send module, running under a dedicated CFQS system transaction. It provides code to take quiesce requests from another task and pass them to SMSVSAM. As DFHFCQS runs under a system transaction, it has full transaction environment which enables it to invoke API-capable global user exits, or to call parts of file control that reference the TCA. DFHFCQS is called from DFHFCQT, the quiesce system transaction module, if the transaction id under which DFHFCQT was started is ‘CFQS’.

**DFHFCQT (file control RLS quiesce common system transaction)**

DFHFCQT is the file control RLS quiesce common system transaction.

There are two file control system transactions dedicated to RLS quiesce processing: CFQS and CFQR. CFQS sends quiesce requests to SMSVSAM in order to initiate the quiesce or unquiesce of a data set throughout the sysplex. CFQR receives quiesce requests from VSAM RLS and performs the quiesce processing required for the CICS region concerned. These transactions share a common top-level program, DFHFCQT.

There is no DFHFCQT parameter list. The action DFHFCQT takes depends on the transid of the transaction it is running under. If it is CFQS then DFHFCQS SEND_QUIESCES is called. If it is CFQR then DFHFCQR RECEIVE_QUIESCES is called. If DFHFCQS or DFHFCQR subsequently fail with a disastrous error, control is returned to DFHFCQT and a transaction abend is issued, having first re-attached the transaction concerned to ensure that RLS Quiesce support is not lost for ever.

**DFHFCQU (file control RLS quiesce processor)**

DFHFCQU is the RLS Quiesce Process module. It processes quiesce requests received from SMSVSAM via the quiesce exit mechanism.
DFHFCQX (file control RLS quiesce exit)

DFHFCQX is the RLS Quiesce Exit module. It is called by SMSVSAM whenever the CICS region concerned is required to perform processing for a quiesce request.

The quiesce exit is specified on the RLS control ACB EXLST. The exit simply initiates processing and returns to VSAM. It must not issue any VSAM requests. It is scheduled as an IRB on the TCB that registered the RLS control ACB. Because of the environment DFHFCQX cannot issue CICS requests. GTF tracing is used to trace entry, exit and any errors.

On entry to DFHFCQX, register 1 contains the address of a VSAM structure mapped by IFGQUIES which defines the quiesce request. The processing of the quiesce request is performed by the CFQR long-running system transaction (DFHFCQR). To communicate the quiesce to CFQR, DFHFCQX creates an FC Quiesce Receive Element (FCQRE) to describe the request, and adds it to a chain in file control static storage, posting an ECB associated with the chain also in FC static.

DFHFCRC (file control recovery control program)

DFHFCRC provides recovery control for file control. All calls from the Recovery Manager domain to file control come through DFHFCRC.

DFHFCRC is called by the Recovery Manager domain to participate in syncpoint and in warm and emergency restart.

Early on during startup File Control registers as a client of the CICS Recovery Manager. During File Control initialization, File Control will add its recovery gate to the kernel, specifying DFHFCRC as the entry point, and then declares the recovery gate to the CICS Recovery Manager via an RMCD SET_GATE call.

At syncpoint, a resource owner such as File Control may be called either
1. to prepare, optionally followed by shunt-unshunt pairs, followed either by calls to backout (as in 2 below) or a call to commit.
2. to backout, which involves start_backout, optional delivery of backout data, and end_backout, followed by prepare and commit, optionally followed by backout retries (which consist of shunt-unshunt pairs followed by the start_backout - delivery of backout data - end_backout - prepare - commit sequence).

At warm or emergency restart, a resource owner such as File Control will be called with start_delivery, optional deliver_recovery and deliver_forget calls, followed by end_deliver.

The Recovery Manager functions processed by DFHFCRC are:
- RMRO PERFORM_PREPARE
- RMRO PERFORM_COMMIT
- RMRO START_BACKOUT
- RMRO DELIVER_BACKOUT_DATA
- RMRO END_BACKOUT
- RMRO PERFORM_SHUNT
- RMRO PERFORM_UNSHUNT
- RMKP TAKE_KEYPOINT
- RMDE START_DELIVERY
- RMDE DELIVER_RECOVERY
DFHFCRC performs different processing depending on the function with which it has been called:

**PERFORM PREPARE**
Any active VSAM requests are terminated, and a vote of READ_ONLY is returned if the unit of work did not make any recoverable file control updates, a vote of YES if the prepare was successful, or a vote of NO otherwise.

**PERFORM COMMIT**
For a forwards syncpoint, any changes made by the unit of work to recoverable user-maintained data tables are committed. For a backwards syncpoint, locks for any backout-failed data sets are retained. All other locks are released.

On transaction termination, the FLABs and FRAB are freed unless there are FLABs marked for retention. On an intermediate syncpoint, various flags in the FLABs and FRAB are reset to indicate that a commit has been performed.

**START BACKOUT**
Any active VSAM requests are terminated, and any changes made by the unit of work to recoverable user-maintained data tables are backed out.

**DELIVER BACKOUT DATA**
The recoverable file control change represented by the log record delivered to DFHFCRC is backed out via calls to DFHFCFR which reverse the update. The change is not backed out if the unit of work has already suffered a backout failure for the data set, or if the data set is in a ‘non-RLS update permitted’ state, or if this call is being made as part of a CEMT or EXEC CICS SET DSNAME RESETLOCKS request.

If a failure occurs during the backout, then backout failure processing is carried out.

**END BACKOUT**
Under normal conditions there should be no processing required at END BACKOUT, but it is conceivable that there might be outstanding active VSAM requests to be terminated.

**PERFORM SHUNT**
The failed parts of the unit of work’s file control structures are put into a condition to survive without an executable transaction environment. This involves retaining any FLABs that are marked for retention, which will allow files to be closed, but not to be reallocated to a different data set.

If this is an intermediate syncpoint, and the shunt is due to a failure in phase 2 of syncpoint, the transactional parts of the unit of work are copied into a new control structure to be passed to the follow-on unit of work. A new FRAB is acquired to anchor this control structure. If this is transaction termination, or the shunt is due to a failure in phase 1 of syncpoint, the transactional parts are cleaned up.

**PERFORM UNSHUNT**
The file control structures are converted back into a condition suitable for a unit of work that is in an executable state. Retained FLABs for the unit of work are restored.
TAKE_KEYPOINT
DFHFCRC is called when CICS takes a keypoint, to perform processing required by BWO backup on non-RLS data sets. This involves the writing of a set of ‘tie up records’ and the calculation of a new BWO recovery time.

START_DELIVERY
DFHFCIR is called to process the call.

DELIVER_RECOVERY
DFHFCIR is called to process the call.

DELIVER_FORGET
DFHFCIR is called to process the call.

END_DELIVERY
DFHFCIR is called to process the call.

DFHFCRD (file control RLS cleanup transaction)
As soon as CICS detects an SMSVSAM server failure, it runs program DFHFCRD under transaction CSFR to perform cleanup.

Following the server failure all current RLS ACBs become unusable. DFHFCRD scans a chain of files open in RLS mode, which is anchored from file control static storage and call DFHFCFS to perform an IMMEDIATE_CLOSE for each open file.

DFHFCRD then waits:
1. for the last file to close,
2. once the last file has closed, for SMSVSAM to complete any residual requests against the RLS control ACB.

When both these events have occurred, DFHFCRD calls DFHFCCA to perform UNREGISTER_CONTROL_ACB processing in order to clean up the CICS and VSAM state with respect to the control ACB.

DFHFCRD finally posts an ECB which allows dynamic RLS restart to go ahead. Dynamic RLS restart cannot start until DFHFCRD has completed clean up and posted this ECB.

DFHFCRF (file control function shipping interface module)

Call mechanism
Kernel subroutine call. Automatic stack storage acquired as part of the call.

Entry address
FC_FCRF_ADDRESS stored in FC Static Storage.

Purpose
DFHFCRF is the function shipping interface module. It is called by the access method independent module DFHFCFR for record management requests (e.g. reads, writes, rewrites, etc.) that are to be directed to files that are defined as remote.

DFHFCRF is called with the FCFR parameter list. From this it constructs an FCRF parameter list, which is subsequently passed to DFHISP and, in turn, either to DFHXFX (the MRO transformer) or to DFHXFFP (the ISC transformer).

DFHFCRF executes the following requests from the DFHFCFRR parameter list:
File control

• Simple read requests
  – READ_INTO and READ_SET

• The read update family
  – READ_UPDATE_INTO and READ_UPDATE_SET
  – REWRITE
  – REWRITE_DELETE
  – UNLOCK

• The browse family
  – START_BROWSE
  – RESET_BROWSE
  – READ_NEXT_SET, READ_NEXT_INTO, READ_PREVIOUS_SET,
    READ_NEXT_UPDATE_SET, READ_NEXT_UPDATE_INTO,
    READ_PREVIOUS_UPDATE_SET, and READ_PREVIOUS_UPDATE_INTO
  – END_BROWSE

• Write requests
  – WRITE

• Delete requests
  – DELETE

Called by
DFHFCFRR, the File Control file request handler.

Inputs
The FCFR parameter list, as defined by the DFHFCFRA DSECT.

Outputs
The FCRF parameter list, as defined by the DFHFCRFA DSECT.

Operation
Traces module entry.

Checks for an explicit SYSID specified on the request and sets the remote system
and remote file name in the DFHFCRF parameter list ready for function shipping.

Increments statistics for the type of request.

Checks request specific parameters

Ships the request.

Handles return codes.

Finally, traces the module exit.

How loaded
By FCRP at file control initialization.

DFHFCRL (file control share control block manager)

Call mechanism
Kernel subroutine call. Automatic stack storage acquired as part of the call.
Entry address
DFHFCRL. The entry point address is held in FC static storage in a field named FC_FCRL_ADDRESS, which is set by DFHFCRP when it loads DFHFCRL.

Purpose
The file control share control block manager is part of the file control component.

This program modifies the CICS specification of a shared resources pool. The changes are allowed to be made only when the actual pool is deleted.

Called by
DFHAMFC, when installing an LSR pool defined by RDO.

Inputs
The FCRL parameter list, as defined by the DFHFCRLA DSECT, is created as part of the subroutine call.

The input parameters are:
- Request identifier
- Pool identifier
- Number of strings
- Maximum key length
- Share limit
- Buffer characteristics

Outputs
The response and reason codes only. These are returned in the FCRL parameter list.

Operation
The SHRCTL block for the specified pool is addressed. A test is made to determine whether or not the pool is currently built; if it is built, the request is rejected with an error response.

The pool characteristics specified in the input parameter list are included in the SHRCTL block.

Finally the information in the SHRCTL block is written to the CICS global catalog.

How loaded
By DFHFCRP as part of file control initialization.

DFHFCRO (file control RLS open/close program)
DFHFCRO performs an equivalent function for RLS opens and closes as is performed by DFHFCN for non-RLS access mode.

DFHFCRP (file control restart program)

Call mechanism
Kernel subroutine call. Automatic stack storage acquired as part of the call.

Entry address
DFHFCRP. This address is needed only by DFHFCIN2 during initialization; it is therefore not saved in FC static storage.
File control

**Purpose**
The file control restart program is part of the file control component. This program creates a file control component on a cold or initial start of CICS, or re-creates it after a warm or emergency start. For a warm or emergency start, the intention is to reconstruct the identical file control environment which was in effect at the time of the previous CICS termination.

**Called by**
DFHFCIN2, during file control initialization.

**Inputs**
None.

**Outputs**
The restarted file control component. File control static addresses and indicators are set up. DFHFCRP’s response and reason codes are set in the parameter list defined by DFHFCRPA DSECT.

**Operation**
Calls loader domain to define (if necessary) and acquire (load) the following file control programs: DFHDTIN5, DFHFPCAT, DFHFCCA, DFHFCDN, DFHFCD2, DFHFCE5, DFHFCCFL, DFHFCCFS, DFHFCSR, DFHFCLF, DFHFCLJ, DFHFCMT, DFHFCCNQ, DFHFCCQI, DFHFCCQU, DFHFCCQX, DFHFCCRC, DFHFCRL, DFHFCRO, DFHFCCRR, DFHFCSR, DFHFCRV, DFHFCSD, DFHFCST, and DFHFCVS.

Adds gates to the kernel for recovery control, ENF services, and log stream failure notification.

Calls storage manager domain to add (create) the following storage subpools: file control general below 16MB, VSAM FCTE, BDAM FCTE, ACB, DCB, SHRCTL, DSN, FFLE, FRAB, FRTE, FLLB, FLAB, RPL, IFGLUWID, file control fixed-length buffer storage. Calls the NQ domain to add (create) enqueue subpools for: dataset record NQs, file record NQs, range NQs, load mode NQs, ESDS write NQs, and UMT loading NQs.

Calls DFHTMP to create TMP primary indexes for the FCT, AFCT, and DSN tables, and a TMP secondary index for the DSN table.

If RLS is supported (correct level of DFSMS, and RLS=YES SIT parameter) initializes the CSFR, CFQS, CFQR and CFOR tasks, registers file control’s interest in the SMSVSAM ENF signal by a LISTEN call to DFHDMEN, and calls DFHFCRR to restart RLS.

On a warm or emergency start:
- Determines installation levels of the MVS/Data Facility Product (MVS/DFP) (or DFSMS), the Data Facility Hierarchical Storage Manager (DFHSM), and the Data Facility Data Set Services (DFDSS) for VSAM backup while open (BWO) support.
- Restores DSNAME blocks from the CICS global catalog, recreating a DSN control block in the DSN subpool storage. For each block, adds its DS name to the TMP primary index, and adds its DS number to the TMP secondary index.
- Restores VSAM file entries from the CICS global catalog. For each entry, adds its file name to the TMP FCT index.
- Restores BDAM file entries from the CICS global catalog. For each entry, adds its file name to the TMP FCT index. Further, for each entry, restores the BDAM DCB from the catalog and copies it to an entry in the DCB storage subpool.
• Restores DSNAME references from the CICS global catalog. For each entry, locates its FCTE and invokes DFHFCDN to connect the FCTE to its DSN block.
• Restores SHRCTL blocks from the CICS global catalog.

On a cold start:
• As for a warm or emergency start, determines installation levels of MVS/DFP, DFHSM, and DFDSS for VSAM backup while open (BWO) support.
• Purges the CICS global catalog of all FCTEs, SHRCTL blocks, DSNAME references, AFCTEs, and BDAM DCBs.
• Calls the loader domain to load the FCT specified by the FCT system initialization parameter.
• Builds all eight SHRCTL blocks, using any information that may have been specified in the loaded FCT. Writes the blocks to the CICS global catalog.
• For each file control table entry in the loaded FCT, creates an FCT entry in the FCT storage subpool, copies the information to it, adds the file name to the TMP index, and writes the table entry to the CICS global catalog.
• Calls the loader domain to delete the previously loaded FCT.

Indicates file control restart complete for non-recoverable business by setting FC_NON_REV_ALLOWED_ECB on.

Sends message to inform that file control restart is complete.

If all was successful, turns on the FCSCMPLT flag in FC static.

Finally, posts the FC_RECOV_ALLOWED_ECB in FC static.

How loaded
By the file control initialization module 2, DFHFCIN2, and deleted after it has completed.

DFHFCRR (file control RLS restart)

DFHFCRR is used to restart the RLS component of File Control. It is called whenever CICS is restarted and after any total RLS failure. DFHFCRR is also called whenever a resource can be made available again after earlier failures have been rectified, and after recovery from Lost Locks.

DFHFCRR is invoked whenever CICS is restarted (COLD, WARM or EMERGENCY) by DFHFCRP, and following any total RLS failure (DYNAMIC restart) by DFHFCES.

DFHFCRR is also called to retry work which has been shunted because a resource (a data set, and RLS cache, or the VSAM RLS server) was not available. For this purpose, it is called by DFHFCQU when CICS is notified that a data set has been unquiesced, has completed a non-BWO copy or has completed forward recovery, and when CICS is notified that a previously failed cache is now available; by DFHFCFL when the API interface is used to retry all shunted work for a given data set; and by DFHFCRO when an override condition is detected, in order to drive any shunted work. DFHFCRR is also called by DFHFCQU when CICS is notified that all systems have completed lost locks recovery for a data set.
**DFHFCRS (file control RLS record management processor)**

DFHFCRS performs an equivalent function for RLS access mode record management requests as is performed by DFHFCVS for non-RLS access mode requests.

**DFHFCRV (file control RLS VSAM interface processor)**

DFHFCRV performs an equivalent function for RLS access mode record management requests as is performed by DFHFCVR for non-RLS access mode requests.

**DFHFCSD (file control shutdown program)**

**Call mechanism**
Kernel subroutine call. Automatic stack storage acquired as part of the call.

**Entry address**
DFHFCSD. The entry point address is held in FC static storage in a field named FC_FCSD_ADDRESS, which is set by DFHFCRP when it loads DFHFCSD.

**Purpose**
The file control shutdown program is part of the file control component. Its purpose is to close all CICS files that are still open during phase 2 of a normal controlled CICS termination. This processing is bypassed for immediate termination.

**Called by**
DFHSTP, to close all open files managed by CICS file control.

**Inputs**
The FCSD parameter list, as defined by the DFHFCSDA DSECT, is created as part of the subroutine call.

The input parameters are:

Type of shutdown (immediate, warm)

**Outputs**
The response and reason codes only, which are returned in the FCSD parameter list.

**Operation**
DFHFCSD has only one function: TERMINATE.

On a ‘warm’ shutdown (that is, a not-immediate shutdown), DFHFCSD calls DFHTMP to scan all FCT entries. For each file, it calls DFHFCFS to close the file. A special CLOSE qualifier (shutdown) is specified on the call to DFHFCFS so as not to catalog the FCT entry as in an ‘unenabled’ state. DFHFCSD also calls DFHFCDO to disconnect coupling facility data table pools.

If RLS is supported, the quiesce system tasks CFQS and CFQR are terminated.

**How loaded**
By DFHFCRP as part of file control initialization.
**DFHFCST (file control statistics program)**

**Call mechanism**
Kernel subroutine call. Automatic stack storage acquired as part of the call.

**Entry address**
DFHFCST. The entry point address is held in FC static storage in a field named FC_FCST_ADDRESS, which is set by DFHFCRP when it loads DFHFCST.

**Purpose**
The file control statistics program is part of the file control component.

This program is called to collect statistics for a single file, together with any data table statistics, or to collect statistics for the activity in a shared resources pool.

It is also called to return file statistics associated with a file’s use of a shared resources pool.

**Called by**
DFHSTFC
Collect file statistics

DFHSTLS
Collect pool statistics and also file-in-pool statistics.

**Inputs**
The FCST parameter list, as defined by the DFHFCSTA DSECT, is created as part of the subroutine call.

The input parameters are:

- Request identifier
- File name
- FCTE token
- Statistics record
- Pool identifier
- Browse token
- Reset indicator

**Outputs**
Returned in the FCST parameter list:

- Browse token
- Response
- Reason

**Operation**
- Collect file statistics:
  - The FCT entry token is validated if supplied; otherwise, the file name is used to locate the FCT entry.
  - The file statistics, and any data table statistics, are collected from the FCTE and copied into the statistics record. The statistics in the FCTE are optionally reset according to the reset indicator.
  - For data tables, a STATISTICS data table service request is issued to retrieve and reset those statistics that are maintained by data table services. These statistics are appended to the file statistics record.
  - The FCT entry is unlocked and the statistics record returned to the caller.
File control

- Collect pool statistics:
  The SHRCTL block for the specified pool is addressed. The pool statistics are
  copied into the statistics record and are returned to the caller.
- Start browse of files in pool:
  Storage is obtained from the general file control pool for the browse cursor. The
  browse token is returned to the caller.
- Get statistics for next file in pool:
  DFHTMP is invoked to locate the FCT entry identified by the browse cursor. If
  the file uses the specified pool, the shared pool statistics for this file are retrieved
  and returned in the statistics record.
  The statistics contain the data and index buffer sizes, and the number of times
  buffer waits occurred.
  The browse cursor is updated before returning to the caller.
- End browse of files in pool:
  The browse cursor storage is freed before returning to the caller.

How loaded
By DFHFCRP as part of file control initialization.

DFHFCVR (file control VSAM interface program)

Call mechanism
BALR, obtaining LIFO storage on entry.

Entry address
DFHFCVR. DFHFCVR is link-edited with DFHFCVS. For calls to DFHFCVR from
DFHFCVS, the entry point address is known to DFHFCVS from the link-edit. This
address is also stored in FC static storage in a field named FC_FCVR_ENTRY. In
addition, there is a further “entry address”, UPADEXIT, which is the entry code for
the UPAD exit code.

Purpose
The VSAM request interface program is part of the file control component.

This module contains code that issues the VSAM requests, and performs UPAD
exit processing in the case of synchronous requests to LSR files, or performs the
IOEVENT wait (‘FCIOWAIT’) in the case of asynchronous requests to NSR files.

The module also contains a number of further routines that implement functions
required by DFHFCVS.

Called by
DFHFCBD
  To issue a message
DFHFCFR
  To wait on a CICS ECB
DFHFCVR
  Recursively, in order to issue an ENDREQ request to free a deadlock
DFHFCVS
  When issuing VSAM requests
DFHFCVS
  To execute one of the constituent functions
VSAM
  To invoke the UPAD exit.
Inputs
The FCWSV parameter list, as defined by the DFHFCWS macro, is created in the
caller’s automatic storage and addressed by register 1 on the call. The input
parameters are:

- Request identifier
- FCTE address
- VSWA address
- ECB address
- Wait resource type
- Message number
- Dump code

In addition, DFHFCVR requires access to the TCA for certain of its operations.

Outputs
FCVR_RESPONSE parameter (only), defined as part of the FCWSV parameter list.

Operation
Initialize: Copies the VSAM exit list to FC static storage. This action is performed
as part of file control initialization.

VSAM_Request: Issues the request to VSAM. Performs the IOEVENT wait.
Handles LSR ‘no buffers’ logical error. Issues change mode request to perform the
request under the concurrent TCB if possible.

Get_Strings and Free_Strings: Acquires and frees the required number of shared
strings from the LSR pool.

Get_TRANID and Free_TRANID: Allocates and releases a VSAM tranid required
during sequential update operations to an LSR file.

Wait_CICSECB: Issues a function request to wait for a CICS ECB to be posted.

Wait_String: Issues a function request to wait for a private string to become
available.

Send_Message: Issues a function request to send a message.

How loaded
Link-edited with DFHFCVS to form the DFHFCVS load module, which is loaded
by DFHFCRP as part of file control initialization.

DFHFCVS (file control VSAM request processor)

Call mechanism
Kernel subroutine call. Automatic stack storage acquired as part of the call.

Entry address
DFHFCVS. The entry point address is held in FC static storage in a field named
FC_FCVS_ADDRESS, which is set by DFHFCRP when it loads DFHFCVS.

Purpose
Processes file control requests to VSAM files.

Also initializes certain FC static storage fields during file control initialization.
Called by
DFHFCFR

To access the VSAM source data set to satisfy requests that cannot be satisfied by the table itself.

After having determined that the request is for a VSAM file.

Inputs
The FCFR parameter list, as defined by the DFHFCFRA DSECT. Also the file control environment, including FC static storage and the FCT.

Outputs
Updated FCFR parameter list.

Operation
Selects on the request type, and passes control to the routine specific to that request.

Acquires and releases the VSWA as necessary.

Logs and journals the request if required.

Performs record-length and key-length checking.

Acquires storage, in the correct key subpool, for requests that specify SET.

Calls DFHFCVR to perform the VSAM request.

Resolves conflicts of exclusive control.

Performs record locking and resolves locking conflicts, including the detection of deadlocks caused either by single tasks that deadlock themselves or by multiple tasks that deadlock each other.

Performs initialization of FC static storage during file control initialization.

For CICS-maintained data tables, calls data table services to update the table to keep it in step with the VSAM source data set.

How loaded
By DFHFCRP as part of file control initialization.

Parameter lists

File control provides the following functions in OCO modules:

FCCA CHECK function

CHECK is issued to get the results of a previous, asynchronous, operation.

Input parameters

CHECK_TOKEN

is a token that was returned on the previous request for which the results are being checked.
Output parameters

ACCMETH_RETURN_CODE
is a two-byte code returned by SMSVSAM.

CONFLICTING_QUIESCE
indicates the type of quiesce which conflicts with this request, and can have any of these values:
QUIESCE|UNQUIESCE|NONBWO_END|BWO_END|NONBWO_START|BWO.START

RESPONSE
is DFHFCCA’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>VSAM_REQUEST_ERROR, RLS_FAILURE</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
</tbody>
</table>

FCCA COLD_START_RLS function

This request is issued as part of CICS cold start processing. CICS issues an IDARECOV TYPE=COLDSTART call to SMSVSAM to release all RLS locks owned by this CICS, and to clear the lost locks status and ‘non RLS update permitted’ state of all data sets with respect to this CICS.

Input parameters

SUBSYSNM
is a pointer to an IFGSYSNM structure.

Output parameters

ACCMETH_RETURN_CODE
is a two-byte code returned by SMSVSAM.

RESPONSE
is DFHFCCA’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>VSAM_REQUEST_ERROR, RLS_FAILURE</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
</tbody>
</table>

FCCA DRAIN_CONTROL_ACB function

The Control ACB must be drained when file control detects that an instance of the SMSVSAM server has become failed. DFHFCCA will set an indicator in FC static storage so that no other RLS activity may proceed, and then drain all existing RLS
access. This involves incrementing the server sequence number in FC static storage, closing all RLS ACBs and unregistering the Control ACB.

**Input parameters**
None

**Output parameters**

RESPONSE
is DFHFCCA’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>DISASTER_PERCOLATION, ABEND</td>
</tr>
</tbody>
</table>

**FCCA INQUIRE_RECOVERY function**

This request is issued as part of CICS start up processing. CICS makes an IDAINQRC request to VSAM to obtain the information necessary to determine what RLS recovery actions are required by CICS.

**Input parameters**

AREA_PTR
is the address of an area in which the IFGINQRC information is to be returned.

AREA_LENGTH
is the length of the supplied area.

**Output parameters**

ACCMETH_RETURN_CODE
is a two-byte code returned by SMSVSAM.

REQUIRED_LENGTH
is the length of the IFGINQRC area to be returned, if it exceeds the length of the supplied area.

RESPONSE
is DFHFCCA’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>AREA_TOO_SMALL, VSAM_REQUEST_ERROR, RLS_FAILURE</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
</tbody>
</table>

**FCCA LOST_LOCKS_COMPLETE function**

CICS issues an IDARECOV TYPE=LL request to SMSVSAM when it has completed recovery processing for a data set that is in lost locks status. SMSVSAM resets the
state of the data set in the sharing control data set to indicate that the data set is no longer lost locks with respect to this CICS.

**Input parameters**

DATASET  
is the 44-character name of the base data set for which CICS has completed lost locks recovery.

[RESTART]  
is an optional parameter which indicates whether the call was issued by file control restart. It can have either of these values:

YES | NO

**Output parameters**

ACCMETH_RETURN_CODE  
is a two-byte code returned by SMSVSAM.

RESPONSE  
is DFHFCCA’s response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON]  
is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>VSAM_REQUEST_ERROR, RLS_FAILURE</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
</tbody>
</table>

**FCCA QUIESCE_COMPLETE function**

When CICS has completed the processing required by it for a quiesce request from SMSVSAM, it issues an IDAQUIES call to SMSVSAM with a quiesce type of QUICMP.

**Input parameters**

DATASET  
is the 44-character name of the base data set for which quiesce processing is complete.

VSAM_QUIESCE_TOKEN  
is a token used to relate quiesce completion to the quiesce request which has been completed, and which is supplied by SMSVSAM when the quiesce request is received by CICS.

**Output parameters**

ACCMETH_RETURN_CODE  
is a two-byte code returned by SMSVSAM.

RESPONSE  
is DFHFCCA’s response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON]  
is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:
**FCCA QUIESCE_REQUEST function**

DFHFCCA issues quiesce requests to SMSVSAM on behalf of the quiesce component of CICS. It issues IDAQUIES calls of the following types:

- **QUICLOSE** to request SMSVSAM to notify all CICS systems that have ACBs open against this data set that these ACBs are to be closed. In addition the data set is marked in the VSAM catalog as being quiesced once these ACBs have been closed.
- **QUIOPEN** to request SMSVSAM to mark the data set as no longer quiesced, i.e. unquiesced. In addition QUIOPEN will cancel an in-progress QUICLOSE.
- **QUIBEND** to request SMSVSAM to cancel an in-progress BWO backup of a data set.
- **QUICEND** to request SMSVSAM to cancel an in-progress non-BWO backup of a data set.

**Input parameters**

**DATASET**

is the 44-character name of the base data set to be quiesced.

**QUIESCE_TYPE**

is the type of quiesce, and can have any of these values:

QUIESCE|UNQUIESCE|NONBWO_END|BWO_END

**[IMMEDIATE]**

applies only when the quiesce type is quiesce, and indicates whether or not the quiesce will force files to close immediately, or will allow in-flight units of work to reach syncpoint. It can have either of these values:

YES|NO

**Output parameters**

**ACCMETH_RETURN_CODE**

is a two-byte code returned by SMSVSAM.

**CHECK_TOKEN**

is a token which will be used on the CHECK request.

**CONFLICTING_QUIESCE**

indicates the type of quiesce which conflicts with this request, and can have any of these values:

QUIESCE|UNQUIESCE|NONBWO_END|BWO_END|NONBWO_START|BWO_START

**RESPONSE**

is DFHFCCA’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**[REASON]**

is returned when RESPONSE is EXCEPTION or DISASTER. Possible values
RESPONSE Possible REASON values

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>VSAM_REQUEST_ERROR, RLS_FAILURE</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
</tbody>
</table>

**FCCA REGISTER_CONTROL_ACB function**

The Control ACB is ‘opened’ using an IDAREGP request to SMSVSAM. The Control ACB must be registered before CICS can open any ‘ordinary’ ACB for RLS access.

**Input parameters**

None

**Output parameters**

- **VSAM_RETURN_CODE**
  
  is a fullword return code from VSAM.

- **VSAM_REASON_CODE**
  
  is a fullword return code from VSAM.

- **VSAM_ERROR_DATA**
  
  is an 8-byte field containing error data returned by VSAM.

**RESPONSE**

is DFHFCCA’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**[REASON]**

is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>RLS_FAILURE, VSAM_REQUEST_ERROR</td>
</tr>
<tr>
<td>DISASTER</td>
<td>DISASTER_PERCOLATION, ABEND</td>
</tr>
</tbody>
</table>

**FCCA RELEASE_LOCKS function**

CICS issues an IDALKREL request to SMSVSAM as part of commit processing at the end of every unit of work. It requests VSAM to release all locks owned by the unit of work.

**Input parameters**

- **LUWID**
  
  is a pointer to an IFGLUWID structure containing the id for the unit of work.

**[RESTART]**

is an optional parameter which indicates whether the call was issued by file control restart. It can have either of these values:

YES|NO

**Output parameters**

- **ACCMETH_RETURN_CODE**
  
  is a two-byte code returned by SMSVSAM.
RESPONSE is DFHFCCA’s response to the call. It can have any of these values: OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON] is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>VSAM_REQUEST_ERROR, RLS_FAILURE</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
</tbody>
</table>

**FCCA RESET_NONRLS_BATCH function**

CICS issues an IDARECOV TYPE=NONRLS request to VSAM when it has completed processing the NSR batch override response from an RLS file open. SMSVSAM resets the state of the data set in the sharing control data set to indicate that the batch override (or ‘non RLS update permitted’) state no longer needs to be reported to this CICS when it opens the data set.

**Input parameters**

**DATASET** is the 44-character name of the base data set for which the state is to be cleared.

**Output parameters**

**ACCMETH_RETURN_CODE** is a two-byte code returned by SMSVSAM.

**RESPONSE** is DFHFCCA’s response to the call. It can have any of these values: OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON] is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>VSAM_REQUEST_ERROR, RLS_FAILURE</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
</tbody>
</table>

**FCCA RETAIN_DATASET_LOCKS function**

CICS issues an IDARETLK TYPE=SS call to SMSVSAM when a unit of work has suffered a backout failure on a data set. This requests SMSVSAM to mark all locks against the data set owned by the unit of work for conversion into retained locks on a subsequent IDALKREL call.

**Input parameters**

**LUWID** is a pointer to an IFGLUWID structure containing the id for the unit of work.
DATASET
is the 44-character name of the base data set which has suffered a backout failure.

Output parameters

ACCMETH_RETURN_CODE
is a two-byte code returned by SMSVSAM.

RESPONSE
is DFHFCGA’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>VSAM_REQUEST_ERROR, RLS_FAILURE</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
</tbody>
</table>

FCCA RETAIN_UOW_LOCKS function

CICS issues an IDARETLK TYPE=IND call to SMSVSAM when a unit of work has encountered an in-doubt failure. This requests VSAM to mark all locks owned by the unit of work for conversion into retained locks on a subsequent IDALKREL call.

Input parameters

LUWID
is a pointer to an IFGLUWID structure containing the id for the unit of work.

Output parameters

ACCMETH_RETURN_CODE
is a two-byte code returned by SMSVSAM.

RESPONSE
is DFHFCGA’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>VSAM_REQUEST_ERROR, RLS_FAILURE</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
</tbody>
</table>

FCCA UNREGISTER_CONTROL_ACB function

The RLS Control ACB is ‘closed’ using an IDAUNRP request to SMSVSAM. The Control ACB cannot be unregistered while there are any ‘ordinary’ ACBs open for RLS access.
**File control**

**Input parameters**
None

**Output parameters**

VSAM_RETURN_CODE
is a fullword return code from VSAM.

VSAM_REASON_CODE
is a fullword reason code from VSAM.

RESPONSE
is DFHFCCA’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>RLS_FAILURE, VSAM_REQUEST_ERROR</td>
</tr>
<tr>
<td>DISASTER</td>
<td>DISASTER_PERCOLATION, ABEND</td>
</tr>
</tbody>
</table>

**FCCI INQUIRE function**

FCCI is the parameter list used by File Control to communicate with the Coupling Facility Data Table cross-memory server, DFHCFMN, for the table inquire function. It is not used by CICS.

**FCCR POINT function**

FCCR is the parameter list used by File Control to communicate with the Coupling Facility Data Table cross-memory server, DFHCFMN, for data access requests.

The POINT function locates a record in a coupling facility data table.

**Input parameters**

TABLE_NAME
is the 16-character name of the CFDT (8 characters padded with trailing spaces).

TABLE_TOKEN
is the token returned on OPEN which must be passed on all subsequent requests against that open table.

KEY
is the 16-byte key of the record to be accessed. For approximate key operations, this specifies the start key and is updated on successful completion to contain the key of the record actually accessed.

KEY_COMPARISON
is the comparison condition, and can take the values
LT|LTEQ|EQ|GTEQ|GT

KEY_MATCH_LENGTH
is the key match length for generic key operations.

UOW_ID
is the unit of work identification, which is required when updating using the locking model (non-recoverable or recoverable).
TRANSACTION_NUMBER
identifies the requesting task within the debug trace, if used.

Output parameters
KEY returns the 16-byte key of the located record.
RESPONSE
is DFHFCRR’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>SERVER_CONNECTION_FAILED, RECORD_NOT_FOUND, TABLE_LOADING, TABLE_TOKEN_INVALID,</td>
</tr>
<tr>
<td></td>
<td>TABLE_Destroyed, POOL_STATE_ERROR, CF_ACCESS_ERROR</td>
</tr>
</tbody>
</table>

FCCR HIGHEST function

FCCR is the parameter list used by File Control to communicate with the Coupling Facility Data Table cross-memory server, DFHCFMN, for data access requests.

The HIGHEST function returns the highest key in a coupling facility data table, if any.

Input parameters
TABLE_NAME
is the 16-character name of the CFDT (8 characters padded with trailing spaces).

TABLE_TOKEN
is the token returned on OPEN which must be passed on all subsequent requests against that open table.

TRANSACTION_NUMBER
identifies the requesting task within the debug trace, if used.

Output parameters
KEY returns the 16-byte key of the highest record.
RESPONSE
is DFHFCRR’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>SERVER_CONNECTION_FAILED, RECORD_NOT_FOUND, TABLE_LOADING, TABLE_TOKEN_INVALID,</td>
</tr>
<tr>
<td></td>
<td>TABLE_Destroyed, POOL_STATE_ERROR, CF_ACCESS_ERROR</td>
</tr>
</tbody>
</table>

FCCR READ function

FCCR is the parameter list used by File Control to communicate with the Coupling Facility Data Table cross-memory server, DFHCFMN, for data access requests.
The READ function reads a record from a coupling facility data table, optionally for update.

**Input parameters**

**TABLE_NAME**
- is the 16-character name of the CFDT (8 characters padded with trailing spaces).

**TABLE_TOKEN**
- is the token returned on OPEN which must be passed on all subsequent requests against that open table.

**KEY_COMPARISON**
- is the comparison condition, and can take the values
  - LT
  - LTEQ
  - EQ
  - GTEQ
  - GT

**KEY_MATCH_LENGTH**
- is the key match length for generic key operations.

**KEY**
- is the 16-byte key of the record to be accessed. For approximate key operations, this specifies the start key and is updated on successful completion to contain the key of the record actually accessed.

**BUFFER**
- is the input buffer for read requests.

**UOW_ID**
- is the unit of work identification, which is required when updating using the locking model (non-recoverable or recoverable).

**SUSPEND**
- specifies whether to wait if the requested record is locked by an active lock, and can take the values
  - YES
  - NO

**TRANSACTION_NUMBER**
- identifies the requesting task within the debug trace, if used.

**Output parameters**

**UPDATE_TOKEN**
- returns a token on a read for update.

**KEY**
- returns the 16-byte key of the highest record.

**LOCK_OWNER_SYSTEM**
- identifies the MVS system from which the record lock was acquired for a record_busy or record_locked condition. Also set when the wait exit is taken for a lock wait.

**LOCK_OWNER_APPLID**
- identifies the applid of the region which owns the record lock for a record_busy or record_locked condition. Also set when the wait exit is taken for a lock wait.

**LOCK_OWNER_UOW_ID**
- identifies the unit of work which owns the record lock for a record_busy or record_locked condition. Also set when the wait exit is taken for a lock wait.

**RESPONSE**
- is DFHFCRR’s response to the call. It can have any of these values:
[REASON]

is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>SERVER_CONNECTION_FAILED, RECORD_NOT_FOUND, RECORD_BUSY, RECORD_LOCKED, TABLE_LOADING, INVALID_REQUEST, INCOMPLETE_UPDATE, TABLE_TOKEN_INVALID, TABLE_DESTROYED, UOW_FAILED, UOW_NOT_IN_FLIGHT, UOW_TOO_LARGE, POOL_STATE_ERROR, CF_ACCESS_ERROR</td>
</tr>
</tbody>
</table>

**FCCR READ_DELETE function**

FCCR is the parameter list used by File Control to communicate with the Coupling Facility Data Table cross-memory server, DFHCFMN, for data access requests.

The READ_DELETE function reads and deletes a record from a coupling facility data table. It is not used by CICS.

**FCCR UNLOCK function**

FCCR is the parameter list used by File Control to communicate with the Coupling Facility Data Table cross-memory server, DFHCFMN, for data access requests.

The UNLOCK function unlocks a record previously read for update in a coupling facility data table.

**Input parameters**

- **TABLE_NAME**
  - is the 16-character name of the CFDT (8 characters padded with trailing spaces).

- **TABLE_TOKEN**
  - is the token returned on OPEN which must be passed on all subsequent requests against that open table.

- **KEY**
  - is the 16-byte key of the record to be unlocked.

- **BUFFER**
  - is the input buffer for read requests.

- **UPDATE_TOKEN**
  - is the token returned on the preceding read for update.

- **UOW_ID**
  - is the unit of work identification, which is required for the locking model (non-recoverable or recoverable).

- **TRANSACTION_NUMBER**
  - identifies the requesting task within the debug trace, if used.

**Output parameters**

- **RESPONSE**
  - is DFHFCR’s response to the call. It can have any of these values:
    - OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]

is returned when RESPONSE is EXCEPTION. Possible values are:
**FCCR LOAD function**

FCCR is the parameter list used by File Control to communicate with the Coupling Facility Data Table cross-memory server, DFHCFMN, for data access requests.

The LOAD function adds a record to a coupling facility data table during loading.

**Input parameters**

| TABLE_NAME | is the 16-character name of the CFDT (8 characters padded with trailing spaces). |
| TABLE_TOKEN | is the token returned on OPEN which must be passed on all subsequent requests against that open table. |
| KEY | is the 16-byte key of the record to be loaded. |
| DATA | is the address and length of the record data to be loaded. |

**Output parameters**

| RESPONSE | is DFHFCRCR's response to the call. It can have any of these values: OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED |
| [REASON] | is returned when RESPONSE is EXCEPTION. Possible values are: |

| RESPONSE | Possible REASON values |
| EXCEPTION | SERVER_CONNECTION_FAILED, DUPLICATE_RECORD, MAXIMUM_RECORDS_REACHED, NO_SPACE_IN_POOL, INVALID_REQUEST, INVALID_LENGTH, TABLE_TOKEN_INVALID, TABLE_DESTROYED, POOL_STATE_ERROR, CF_ACCESS_ERROR |

**FCCR WRITE function**

FCCR is the parameter list used by File Control to communicate with the Coupling Facility Data Table cross-memory server, DFHCFMN, for data access requests.

The WRITE function writes a new record to a coupling facility data table.

**Input parameters**

| TABLE_NAME | is the 16-character name of the CFDT (8 characters padded with trailing spaces). |
TABLE_TOKEN
is the token returned on OPEN which must be passed on all subsequent requests against that open table.

KEY is the 16-byte key of the record to be added.

DATA is the address and length of the record data to be added.

UOW_ID
is the unit of work identification, which is required when updating using the locking model (non-recoverable or recoverable).

SUSPEND
specifies whether to wait if the requested record is locked by an active lock, and can take the values YES|NO

TRANSACTION_NUMBER
identifies the requesting task within the debug trace, if used.

Output parameters

LOCK_OWNER_SYSTEM
identifies the MVS system from which the record lock was acquired for a record_busy or record_locked condition. Also set when the wait exit is taken for a lock wait.

LOCK_OWNER_APPLID
identifies the applid of the region which owns the record lock for a record_busy or record_locked condition. Also set when the wait exit is taken for a lock wait.

LOCK_OWNER_UOW_ID
identifies the unit of work which owns the record lock for a record_busy or record_locked condition. Also set when the wait exit is taken for a lock wait.

RESPONSE
is DFHFCCR’s response to the call. It can have any of these values: OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>SERVER_CONNECTION_FAILED, DUPLICATE_RECORD, RECORD_BUSY, RECORD_LOCKED,</td>
</tr>
<tr>
<td></td>
<td>MAXIMUM_RECORDS_REACHED, NO_SPACE_IN_POOL, TABLE_LOADING, INVALID_REQUEST, INVALID_LENGTH,</td>
</tr>
<tr>
<td></td>
<td>UPDATE_TOKEN_INVALID, INCOMPLETE_UPDATE, TABLE_TOKEN_INVALID, TABLE_DESTROYED, UOW_FAILED,</td>
</tr>
<tr>
<td></td>
<td>UOW_NOT_IN_FLIGHT, UOW_TOO_LARGE, POOL_STATE_ERROR, CF_ACCESS_ERROR</td>
</tr>
</tbody>
</table>

FCCR REWRITE function

FCCR is the parameter list used by File Control to communicate with the Coupling Facility Data Table cross-memory server, DFHCFMN, for data access requests.

The REWRITE function rewrites an existing record in a coupling facility data table, following a read for update.
Input parameters

**TABLE_NAME**
is the 16-character name of the CFDT (8 characters padded with trailing spaces).

**TABLE_TOKEN**
is the token returned on OPEN which must be passed on all subsequent requests against that open table.

**KEY**
is the 16-byte key of the record to be rewritten.

**DATA**
is the address and length of the record data to be rewritten.

**UPDATE_TOKEN**
is the token returned on the preceding read for update.

**UOW_ID**
is the unit of work identification, which is required when updating using the locking model (non-recoverable or recoverable).

**SUSPEND**
specifies whether to wait if the requested record is locked by an active lock, and can take the values

YES|NO

**TRANSACTION_NUMBER**
identifies the requesting task within the debug trace, if used.

Output parameters

**LOCK_OWNER_SYSTEM**
identifies the MVS system from which the record lock was acquired for a record_busy or record_locked condition. Also set when the wait exit is taken for a lock wait.

**LOCK_OWNER_APPLID**
identifies the applid of the region which owns the record lock for a record_busy or record_locked condition. Also set when the wait exit is taken for a lock wait.

**LOCK_OWNER_UOW_ID**
identifies the unit of work which owns the record lock for a record_busy or record_locked condition. Also set when the wait exit is taken for a lock wait.

**RESPONSE**
is DFHFCRR’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**[REASON]**
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>SERVER_CONNECTION_FAILED, RECORD_NOT_FOUND, RECORD_CHANGED, RECORD_BUSY, RECORD_LOCKED,</td>
</tr>
<tr>
<td></td>
<td>MAXIMUM_RECORDS_REACHED, NO_SPACE_IN_POOL, TABLE&gt;Loading, INVALID_REQUEST, INVALID_LENGTH,</td>
</tr>
<tr>
<td></td>
<td>UPDATE_TOKEN_INVALID, INCOMPLETE_UPDATE, TABLE_TOKEN_INVALID, TABLE_DESTROYED, UOW_FAILED,</td>
</tr>
<tr>
<td></td>
<td>UOW_NOT_IN_FLIGHT, UOW_TOO_LARGE, POOL_STATE_ERROR, CF_ACCESS_ERROR</td>
</tr>
</tbody>
</table>
**FCCR DELETE function**

FCCR is the parameter list used by File Control to communicate with the Coupling Facility Data Table cross-memory server, DFHCFMN, for data access requests.

The DELETE function deletes a record from a coupling facility data table, following a read for update.

**Input parameters**

**TABLE_NAME**
- is the 16-character name of the CFDT (8 characters padded with trailing spaces).

**TABLE_TOKEN**
- is the token returned on OPEN which must be passed on all subsequent requests against that open table.

**KEY_COMPARISON**
- is the comparison condition, and can take the values
  - LT
  - LTEQ
  - EQ
  - GTEQ
  - GT

**KEY_MATCH_LENGTH**
- is the key match length for generic key operations.

**KEY**
- is the 16-byte key of the record to be deleted.

**UPDATE_TOKEN**
- is the token returned on the preceding read for update.

**UOW_ID**
- is the unit of work identification, which is required when updating using the locking model (non-recoverable or recoverable).

**SUSPEND**
- specifies whether to wait if the requested record is locked by an active lock, and can take the values
  - YES
  - NO

**TRANSACTION_NUMBER**
- identifies the requesting task within the debug trace, if used.

**Output parameters**

**KEY**
- is the 16-byte key of the record actually deleted.

**LOCK_OWNER_SYSTEM**
- identifies the MVS system from which the record lock was acquired for a record_busy or record_locked condition. Also set when the wait exit is taken for a lock wait.

**LOCK_OWNER_APPLID**
- identifies the applid of the region which owns the record lock for a record_busy or record_locked condition. Also set when the wait exit is taken for a lock wait.

**LOCK_OWNER_UOW_ID**
- identifies the unit of work which owns the record lock for a record_busy or record_locked condition. Also set when the wait exit is taken for a lock wait.

**RESPONSE**
- is DFHFCRCR’s response to the call. It can have any of these values:
[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>SERVER_CONNECTION_FAILED, RECORD_NOT_FOUND, RECORD_CHANGED,</td>
</tr>
<tr>
<td></td>
<td>RECORD_BUSY, RECORD_LOCKED, TABLE_LOADING, INVALID_REQUEST,</td>
</tr>
<tr>
<td></td>
<td>UPDATE_TOKEN_INVALID, INCOMPLETE_UPDATE, TABLE_TOKEN_INVALID,</td>
</tr>
<tr>
<td></td>
<td>TABLE_DESTROYED, UOW_FAILED, UOW_NOT_IN_FLIGHT, UOW_TOO_LARGE,</td>
</tr>
<tr>
<td></td>
<td>POOL_STATE_ERROR, CF_ACCESS_ERROR</td>
</tr>
</tbody>
</table>

**FCCR DELETE_MULTIPLE function**

FCCR is the parameter list used by File Control to communicate with the Coupling Facility Data Table cross-memory server, DFHCFMN, for data access requests.

The DELETE_MULTIPLE function deletes records from a coupling facility data table, subject to key match conditions, until no more records match or an exception occurs.

**Input parameters**

**TABLE_NAME**
is the 16-character name of the CFDT (8 characters padded with trailing spaces).

**TABLE_TOKEN**
is the token returned on OPEN which must be passed on all subsequent requests against that open table.

**KEY_COMPARISON**
is the comparison condition, and can take the values

<table>
<thead>
<tr>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>LT</td>
</tr>
<tr>
<td>LTEQ</td>
</tr>
<tr>
<td>EQ</td>
</tr>
<tr>
<td>GTEQ</td>
</tr>
<tr>
<td>GT</td>
</tr>
</tbody>
</table>

**KEY_MATCH_LENGTH**
is the key match length for generic key operations.

**KEY**
is the 16-byte key of the record(s) to be deleted.

**UOW_ID**
is the unit of work identification, which is required when updating using the locking model (non-recoverable or recoverable).

**SUSPEND**
specifies whether to wait if the requested record is locked by an active lock, and can take the values

<table>
<thead>
<tr>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>YES</td>
</tr>
<tr>
<td>NO</td>
</tr>
</tbody>
</table>

**TRANSACTION_NUMBER**
identifies the requesting task within the debug trace, if used.

**Output parameters**

**DELETED_RECORD_COUNT**
is the number of records successfully deleted by the delete_multiple request.

**KEY**
is the 16-byte key of the last record deleted.
LOCK_OWNER_SYSTEM
identifies the MVS system from which the record lock was acquired for a record_busy or record_locked condition. Also set when the wait exit is taken for a lock wait.

LOCK_OWNER_APPLID
identifies the applid of the region which owns the record lock for a record_busy or record_locked condition. Also set when the wait exit is taken for a lock wait.

LOCK_OWNER_UOW_ID
identifies the unit of work which owns the record lock for a record_busy or record_locked condition. Also set when the wait exit is taken for a lock wait.

RESPONSE
is DFHFCCR’s response to the call. It can have any of these values:
OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>SERVER_CONNECTION_FAILED, RECORD_NOT_FOUND, RECORD_CHANGED, RECORD_BUSY, RECORD_LOCKED,</td>
</tr>
<tr>
<td></td>
<td>TABLE_LOADING, INVALID_REQUEST, UPDATE_TOKEN_INVALID, INCOMPLETE_UPDATE, TABLE_TOKEN_</td>
</tr>
<tr>
<td></td>
<td>INVALID, TABLE_DESTROYED, UOW_FAILED, UOW_NOT_IN_FLIGHT, UOW_TOO_LARGE, POOL_STATE_</td>
</tr>
<tr>
<td></td>
<td>ERROR, CF_ACCESS_ERROR</td>
</tr>
</tbody>
</table>

FCCT OPEN function
FCCT is the parameter list used by File Control to communicate with the Coupling Facility Data Table cross-memory server, DFHCFMN, for table status functions (Open, Close etc.).

The OPEN function defines a coupling facility data table table and establishes a connection between it and a CICS file. A security check is performed for access to the table name. If the table does not exist, it is implicitly created. If the table requires loading, it can only be opened if the access mode specifies exclusive access (or prefer_shared, allowing exclusive access if necessary).

Input parameters
TABLE_NAME
is the 16-character name of the CFDT (8 characters padded with trailing spaces).

RECORD_LENGTH
specifies the maximum record length, in the range 1 to 32767.

KEY_LENGTH
specifies the key length, in the range 1 to 16.

MAXIMUM_RECORDS
specifies the maximum number of records which can be stored in the table.

UPDATE_MODEL
specifies the method to be used for updating. It can take any of the values:
Contention means version compare and swap. Locking means normal update locking. Recoverable includes backout support in addition to the basic locking model.

**INITIAL_LOAD**
specifies whether initial load is required. It can take the values:
YES|NO

**OPEN_MODE**
specifies a read_only or read_write open. It can take the values
READ_ONLY|READ_WRITE

**ACCESS_MODE**
specifies whether the table is being opened for exclusive or shared use. It can take the values:
EXCLUSIVE|SHARED|PREFER_SHARED

Only one user at a time can have an exclusive open active. If the table requires loading and is not yet being loaded, it can only be opened in exclusive mode. If PREFER_SHARED is specified, the table will be opened in exclusive mode if loading is required, otherwise it will be open in shared mode.

**SHARED_ACCESS**
specifies for an exclusive mode open whether other users will be allowed shared access to the file at the same time. It can take the values:
NONE|READ_ONLY|READ_WRITE

**TRANSACTION_NUMBER**
identifies the requesting task within the debug trace, if used.

**Output parameters**

**TABLE_TOKEN**
is a unique token representing the connection to this table. It must be passed on all subsequent requests against that open table, including close and set.

**RECORD_LENGTH**
returns the maximum record length of the table.

**KEY_LENGTH**
returns the key length of the table.

**MAXIMUM_RECORDS**
returns the maximum number of records limit for the table.

**UPDATE_MODEL**
returns the update model for the data table. It can take any of the values:
CONTENTION|LOCKING|RECOVERABLE

Contestation means version compare and swap. Locking means normal update locking. Recoverable includes backout support in addition to the basic locking model.

**INITIAL_LOAD**
returns whether or not the data table requires initial loading. It can take the values:
YES|NO
ACCESS_MODE
returns whether the table was opened for exclusive or shared use. It can take the values:
EXCLUSIVE|SHARED

LOADED
returns an indication of whether the table has been loaded. If the table was created as empty this is set to yes as if loading were already done. It can take the values:
YES|NO

CURRENT_USERS
returns the number of explicit opens which are currently active against the table (not including internal recoverable opens issued by the server).

CURRENT_RECORDS
returns the number of records in the data table.

CURRENT_HIGH_KEY
returns the key of the last record in the table at the time of the request, or low values if the table does not contain any records.

RESPONSE
is DFHFCCT’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>SERVER_CONNECTION_FAILED, ACCESS_NOT_ALLOWED, TABLE_NOT_AVAILABLE, NOT_YET_LOADED,</td>
</tr>
<tr>
<td></td>
<td>SHARED_ACCESS_CONFLICT, EXCLUSIVE_ACCESS_CONFLICT, INCOMPATIBLE_ATTRIBUTES,</td>
</tr>
<tr>
<td></td>
<td>INCOMPLETE_ATTRIBUTES, INCORRECT_STATE, RECOVERY_NOT_ENABLED,</td>
</tr>
<tr>
<td></td>
<td>OPTION_NOT_SUPPORTED, NO_SPACE_IN_POOL, MAXIMUM TABLES_REACHED, TOO_MANY_USERS,</td>
</tr>
<tr>
<td></td>
<td>TABLE_DESTROYED, POOL_STATE_ERROR, CF_ACCESS_ERROR</td>
</tr>
</tbody>
</table>

FCCT CLOSE function
FCCT is the parameter list used by File Control to communicate with the Coupling Facility Data Table cross-memory server, DFHCFMN, for table status functions (Open, Close etc.).

The CLOSE function terminates the connection to the specified table.

Input parameters

TABLE_NAME
is the 16-character name of the CFDT (8 characters padded with trailing spaces).

TABLE_TOKEN
is the token which was returned by the open.

_TRANSACTION_NUMBER
identifies the requesting task within the debug trace, if used.
Output parameters

**RESPONSE**

is DFHCCT’s response to the call. It can have any of these values:

OK| EXCEPTION| DISASTER| INVALID| KERNERROR| PURGED

**[REASON]**

is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>SERVER_CONNECTION_FAILED, TABLE_TOKEN_INVALID,</td>
</tr>
<tr>
<td></td>
<td>TABLE_DESTROYED, POOL_STATE_ERROR, CF_ACCESS_ERROR</td>
</tr>
</tbody>
</table>

**FCCT DELETE function**

FCCT is the parameter list used by File Control to communicate with the Coupling Facility Data Table cross-memory server, DFHCFMN, for table status functions (Open, Close etc.).

The DELETE function deletes a coupling facility data table, provided that it is not currently open. A security check for table access is performed.

**Input parameters**

**TABLE_NAME**

is the 16-character name of the CFDT (8 characters padded with trailing spaces).

**TRANSACTION_NUMBER**

identifies the requesting task within the debug trace, if used.

**Output parameters**

**RESPONSE**

is DFHCCT’s response to the call. It can have any of these values:

OK| EXCEPTION| DISASTER| INVALID| KERNERROR| PURGED

**[REASON]**

is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>SERVER_CONNECTION_FAILED, ACCESS_NOT_ALLOWED,</td>
</tr>
<tr>
<td></td>
<td>TABLE_NOT_FOUND, EXCLUSIVE_ACCESS_CONFLICT,</td>
</tr>
<tr>
<td></td>
<td>TABLE_DESTROYED, POOL_STATE_ERROR, CF_ACCESS_ERROR</td>
</tr>
</tbody>
</table>

**FCCT SET function**

FCCT is the parameter list used by File Control to communicate with the Coupling Facility Data Table cross-memory server, DFHCFMN, for table status functions (Open, Close etc.).

The SET function is used to change the attributes of a table. The maximum number of records can be changed, the open mode can be changed to indicate no longer loading, and the access mode can be changed from exclusive to shared.
Input parameters

TABLE_NAME
is the 16-character name of the CFDT (8 characters padded with trailing spaces).

MAXIMUM_RECORDS
specifies the maximum number of records which can be stored in the table.

AVAILABLE
indicates whether new open requests are to be allowed for this table. It can take the values:
YES|NO

LOADED
indicates whether the table is to be marked as loaded. It can take the values:
YES|NO

ACCESS_MODE
specifies the access mode which is to be set for the table. It can take the values:
EXCLUSIVE|SHARED

The access mode is normally set to shared when a data table load has completed.

SHARED_ACCESS
specifies the shared access which is to be allowed by other users when the access mode is shared.
NONE|READ_ONLY|READ_WRITE

TRANSACTION_NUMBER
identifies the requesting task within the debug trace, if used.

Output parameters

RESPONSE
is DFHFCCT’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>SERVER_CONNECTION_FAILED, ACCESS_NOT_ALLOWED, TABLE_NOT_FOUND, SHARED_ACCESS_CONFLICT,</td>
</tr>
<tr>
<td></td>
<td>EXCLUSIVE_ACCESS_CONFLICT, ALREADY_SET, INCORRECT_STATE, OPTION_NOT_SUPPORTED,</td>
</tr>
<tr>
<td></td>
<td>TABLE_TOKEN_INVALID, TABLE_DESTROYED, POOL_STATE_ERROR, CF_ACCESS_ERROR</td>
</tr>
</tbody>
</table>

FCCT EXTRACT_STATISTICS function

FCCT is the parameter list used by File Control to communicate with the Coupling Facility Data Table cross-memory server, DFHFCHMN, for table status functions (Open, Close etc.).

The EXTRACT_STATISTICS function returns information about a table which is currently open, with optional reset.
Input parameters

TABLE_NAME
is the 16-character name of the CFDT (8 characters padded with trailing spaces).

TABLE_TOKEN
is the token which was returned by the open.

RESET_STATISTICS
is an optional parameter which specifies whether or not statistics are to be reset. It can take the values YES|NO

TRANSACTION_NUMBER
identifies the requesting task within the debug trace, if used.

Output parameters

CURRENT_USERS
is the number of explicit opens which are currently active against the table (not including internal recoverable opens issued by the server).

CURRENT_RECORDS
is the number of records currently in the data table.

HIGHEST_RECORDS
is the highest number of records in the table as seen by the current server at any time since the last statistics reset.

CONTENTION_COUNT
is the number of times a rewrite or delete failed because of a mismatched version (for the contention model) or the number of times that a lock was found to be unavailable (for the locking or recoverable models) since the last statistics reset.

RESPONSE
is DFHFCCT’s response to the call. It can have any of these values: OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>SERVER_CONNECTION_FAILED, TABLE_TOKEN_INVALID</td>
</tr>
</tbody>
</table>

FCCU PREPARE function

FCCU is the parameter list used by File Control to communicate with the Coupling Facility Data Table cross-memory server, DFHCFMN, for unit of work related functions.

The PREPARE function prepares to commit a unit of work.

Input parameters

UOW_ID
is the CICS unit of work identification, which is prefixed by the CFDT server with the subsystem name to form the fully qualified unit of work identifier.
TRANSACTION_NUMBER
  is used for debug trace purposes.

Output parameters

RESPONSE
  is DFHFCU’s response to the call. It can have any of these values:

  OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
  is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>SERVER_CONNECTION_FAILED, RECOVERY_NOT_ENABLED, UOW_NOT_FOUND, UOW_MADE_NO_CHANGES, UOW_FAILED, NO_SPACE_IN_POOL, POOL_STATE_ERROR, CF_ACCESS_ERROR</td>
</tr>
</tbody>
</table>

FCCU RETAIN function

FCCU is the parameter list used by File Control to communicate with the Coupling Facility Data Table cross-memory server, DFHCFMN, for unit of work related functions.

The RETAIN function marks a unit of work as retained.

Input parameters

UOW_ID
  is the CICS unit of work identification, which is prefixed by the CFDT server with the subsystem name to form the fully qualified unit of work identifier.

TRANSACTION_NUMBER
  is used for debug trace purposes.

Output parameters

RESPONSE
  is DFHFCU’s response to the call. It can have any of these values:

  OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
  is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>SERVER_CONNECTION_FAILED, RECOVERY_NOT_ENABLED, UOW_NOT_FOUND, UOW_MADE_NO_CHANGES, UOW_FAILED, NO_SPACE_IN_POOL, POOL_STATE_ERROR, CF_ACCESS_ERROR</td>
</tr>
</tbody>
</table>

FCCU COMMIT function

FCCU is the parameter list used by File Control to communicate with the Coupling Facility Data Table cross-memory server, DFHCFMN, for unit of work related functions.

The COMMIT function commits a unit of work.
Input parameters

UOW_ID
is the CICS unit of work identification, which is prefixed by the CFDT server with the subsystem name to form the fully qualified unit of work identifier.

TRANSACTION_NUMBER
is used for debug trace purposes.

Output parameters

RESPONSE
is DFHFCCU’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>SERVER_CONNECTION_FAILED, RECOVERY_NOT_ENABLED,</td>
</tr>
<tr>
<td></td>
<td>UOW_NOT_FOUND, UOW_MADE_NO_CHANGES,</td>
</tr>
<tr>
<td></td>
<td>UOW_FAILED, NO_SPACE_IN_POOL, POOL_STATE_ERROR,</td>
</tr>
<tr>
<td></td>
<td>CF_ACCESS_ERROR</td>
</tr>
</tbody>
</table>

FCCU BACKOUT function

FCCU is the parameter list used by File Control to communicate with the Coupling Facility Data Table cross-memory server, DFHCFMN, for unit of work related functions.

The BACKOUT function backs out a unit of work.

Input parameters

UOW_ID
is the CICS unit of work identification, which is prefixed by the CFDT server with the subsystem name to form the fully qualified unit of work identifier.

TRANSACTION_NUMBER
is used for debug trace purposes.

Output parameters

RESPONSE
is DFHFCCU’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>SERVER_CONNECTION_FAILED, RECOVERY_NOT_ENABLED,</td>
</tr>
<tr>
<td></td>
<td>UOW_NOT_FOUND, UOW_MADE_NO_CHANGES,</td>
</tr>
<tr>
<td></td>
<td>UOW_FAILED, NO_SPACE_IN_POOL, POOL_STATE_ERROR,</td>
</tr>
<tr>
<td></td>
<td>CF_ACCESS_ERROR</td>
</tr>
</tbody>
</table>
FCCU INQUIRE function

FCCU is the parameter list used by File Control to communicate with the Coupling Facility Data Table cross-memory server, DFHCFMN, for unit of work related functions.

The INQUIRE function inquires about the status of a unit of work.

Input parameters

UOW_ID

is the CICS unit of work identification, which is prefixed by the CFDT server with the subsystem name to form the fully qualified unit of work identifier.

UOW_RESTARTED

is an optional parameter which indicates whether the inquire should select only units of work which have been through restart processing, and can take the values:

NO|YES

TRANSACTION_NUMBER

is used for debug trace purposes.

BROWSE

specifies whether the inquire is for a single unit of work or for the first or next UOW in a browse. If omitted, a single UOW inquire is performed. If specified, it can take the values

FIRST|NEXT

FIRST indicates a search for a UOWID greater than or equal to the specified UOWID, and NEXT indicates a search for a UOWID greater than the specified UOWID.

Output parameters

UOW_STATE IN_FLIGHT, IN_DOUBT, IN_COMMIT, IN_BACKOUT

indicates the state of an active unit of work, and can have any of the values:

IN_FLIGHT|IN_DOUBT|IN_COMMIT|IN_BACKOUT

In_flight means that the unit of work has made some changes but has not yet reached the stage of prepare to commit. In_doubt means that it has been prepared but not committed or backed out. In_commit means that commit processing has been started. In_backout means that backout processing has been started. (When commit or backout processing completes, the unit of work is deleted).

UOW_ID

is the CICS unit of work id of the UOW for which inquire data is being returned.

UOW_RESTARTED

indicates whether the unit of work has been through restart processing, and can take the values:

NO|YES
**File control**

**UOW_RETAINED**
indicates whether the locks for the unit of work have been marked as
retained, either explicitly within the current connection or implicitly by a
restart. It can take the values:
NO|YES

**RESPONSE**
is DFHFCCU’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**[REASON]**
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>SERVER_CONNECTION_FAILED, RECOVERY_NOT_ENABLED,</td>
</tr>
<tr>
<td></td>
<td>UOW_NOT_FOUND</td>
</tr>
</tbody>
</table>

**FCCU RESTART function**
FCCU is the parameter list used by File Control to communicate with the Coupling
Facility Data Table cross-memory server, DFHCFMN, for unit of work related
functions.

The RESTART function establishes recovery status on connecting to a CFDT server.

**Input parameters**

**UOW_SUBSYSTEM_NAME**
is not specified by CICS (the CICS applid is used by default).

**TRANSACTION_NUMBER**
is used for debug trace purposes.

**Output parameters**

**RESPONSE**
is DFHFCCU’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**[REASON]**
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>SERVER_CONNECTION_FAILED, SUBSYSTEM_ALREADY_ACTIVE,</td>
</tr>
<tr>
<td></td>
<td>RESTART_ALREADY_ACTIVE, TABLE_OPEN_FAILED, NO_SPACE_IN_POOL,</td>
</tr>
<tr>
<td></td>
<td>POOL_STATE_ERROR, CF_ACCESS_ERROR</td>
</tr>
</tbody>
</table>

**FCDS EXTRACT_CFDT_STATS function**
This function causes statistics relating to coupling facility data table usage to be
extracted from the coupling facility data tables server.

**Input parameters**

**FCTE_POINTER**
is the address of the FCTE entry of the file for which CFDT statistics are to be extracted.
RESET_STATISTICS
indicates whether the statistics fields are to be reset to zero or not. It takes the values YES|NO

TRANSACTION_NUMBER
is an optional parameter which allows the transaction number to be passed to the CFDT server for inclusion in trace messages.

Output parameters
CURRENT_USERS
is an optional fullword parameter which returns the current number of users of the coupling facility data table (that is, the number of opens issued against it).

MAXIMUM_RECORDS
is an optional fullword parameter which returns the current value of the MAXNUMRECS limit for the data table.

CURRENT_RECORDS
is an optional fullword parameter which returns the current number of records in the coupling facility data table.

HIGHEST_RECORDS
is an optional fullword parameter which returns the highest number of records which have ever been in this coupling facility data table since it was last created.

CONTENTION_COUNT
is an optional fullword parameter which returns the number of contentions which have been detected, for a coupling facility data table which uses the contention update model.

RESPONSE
is DFHFCDS’s response to the call. It can have any of these values: OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION, INVALID or DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>CFDT_CONNECT_ERROR, CFDT_DISCONNECT_ERROR, CFDT_REOPEN_ERROR, CFDT_SERVER_NOT_AVAILABLE, CFDT_SERVER_NOT_FOUND, CFDT_STATS_ERROR, CFDT_SYSIDERR, CFDT_TABLE_GONE</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
<tr>
<td>DISASTER</td>
<td>POOL_ELEMENT_NOT_FOUND, ABEND, DISASTER_PERCOLATION</td>
</tr>
</tbody>
</table>

FCDS DISCONNECT_CFDT_POOLS function
This function causes CICS to disconnect from any coupling facility data table pools to which it is connected.

Input parameters
None
Output parameters

RESPONSE

is DFHFCDS’s response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON]

is returned when RESPONSE is EXCEPTION, INVALID or DISASTER.

Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>CFDT_DISCONNECT_ERROR</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND, DISASTER_PERCOLATION</td>
</tr>
</tbody>
</table>

FCDU PREPARE function

This function causes the coupling facility data table server to be called to prepare a unit of work which has made recoverable updates to one or more coupling facility data tables.

Input parameters

POOL_ELEM_ADDR

is the address of the pool element which identifies the coupling facility data table pool for which the prepare is to be issued. One or more of the coupling facility data tables updated by the unit of work reside in this pool. The prepare call will be issued to the CFDT server for this pool.

POOL_NAME

is the name of the coupling facility data table pool. The pool name is included for diagnostic purposes.

UOW_ID

is the identifier for the unit of work which is to be prepared.

Output parameters

RESPONSE

is DFHFCDU’s response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON]

is returned when RESPONSE is EXCEPTION, INVALID or DISASTER.

Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>SERVER_CONNECTION_FAILED, RECOVERY_NOT_ENABLED, UOW_NOT_FOUND, UOW_MADE_NO_CHANGES, UOW_FAILED, NO_SPACE_IN_POOL, POOL_STATE_ERROR, CF_ACCESS_ERROR, CFDT_SYSIDERR, CFDT_SERVER_NOT_AVAILABLE, CFDT_SERVER_NOT_FOUND, CFDT_CONNECT_ERROR, CFDT_DISCONNECT_ERROR, RESYNC_RETRY_FAILED</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND, DISASTER_PERCOLATION</td>
</tr>
</tbody>
</table>
**FCDU RETAIN function**

This function causes the coupling facility data table server to be called to convert locks held by the unit of work against recoverable coupling facility data tables into retained locks.

**Input parameters**

**POOL_ELEM_ADDR**

is the address of the pool element which identifies the coupling facility data table pool for which the retain is to be issued. One or more of the coupling facility data tables updated by the unit of work reside in this pool. The retain call will be issued to the CFDT server for this pool.

**POOL_NAME**

is the name of the coupling facility data table pool. The pool name is included for diagnostic purposes.

**UOW_ID**

is the identifier for the unit of work for which locks are to be retained.

**Output parameters**

**RESPONSE**

is DFHFCDU’s response to the call. It can have any of these values:

OK, EXCEPTION, DISASTER, INVALID, KERNERROR, PURGED

**[REASON]**

is returned when RESPONSE is EXCEPTION, INVALID or DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>SERVER_CONNECTION_FAILED, RECOVERY_NOT_ENABLED, UOW_NOT_FOUND, UOW_MADE_NO_CHANGES,</td>
</tr>
<tr>
<td></td>
<td>UOW_FAILED, NO_SPACE_IN_POOL, POOL_STATE_ERROR, CF_ACCESS_ERROR, CFDT_SYSERR,</td>
</tr>
<tr>
<td></td>
<td>CFDT_SERVER_NOT_AVAILABLE, CFDT_SERVER_NOT_FOUND, CFDT_CONNECT_ERROR, CFDT_DISCONNECT_ERROR,</td>
</tr>
<tr>
<td></td>
<td>RESYNC_RETRY_FAILED</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND, DISASTER_PERCOLATION</td>
</tr>
</tbody>
</table>

**FCDU COMMIT function**

This function causes the coupling facility data table server to be called to commit a unit of work which has made recoverable updates to one or more coupling facility data tables.

**Input parameters**

**POOL_ELEM_ADDR**

is the address of the pool element which identifies the coupling facility data table pool for which the commit is to be issued. One or more of the coupling facility data tables updated by the unit of work reside in this pool. The commit call will be issued to the CFDT server for this pool.

**POOL_NAME**

is the name of the coupling facility data table pool. The pool name is included for diagnostic purposes.
**File control**

**UOW_ID**

is the identifier for the unit of work which is to be committed.

**Output parameters**

**RESPONSE**

is DFHFCDU’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**[REASON]**

is returned when RESPONSE is EXCEPTION, INVALID or DISASTER.

Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>SERVER_CONNECTION_FAILED, RECOVERY_NOT_ENABLED, UOW_NOT_FOUND, UOW_MADE_NO_CHANGES, UOW_FAILED, NO_SPACE_IN_POOL, POOL_STATE_ERROR, CF_ACCESS_ERROR, CFDT_SYSIDERR, CFDT_SERVER_NOT_AVAILABLE, CFDT_SERVER_NOT_FOUND, CFDT_CONNECT_ERROR, CFDT_DISCONNECT_ERROR, RESYNC_RETRY_FAILED</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND, DISASTER_PERCOLATION</td>
</tr>
</tbody>
</table>

**FCDU BACKOUT function**

This function causes the coupling facility data table server to be called to backout a unit of work which has made recoverable updates to one or more coupling facility data tables.

**Input parameters**

**POOL_ELEM_ADDR**

is the address of the pool element which identifies the coupling facility data table pool for which the backout is to be issued. One or more of the coupling facility data tables updated by the unit of work reside in this pool. The backout call will be issued to the CFDT server for this pool.

**POOL_NAME**

is the name of the coupling facility data table pool. The pool name is included for diagnostic purposes.

**UOW_ID**

is the identifier for the unit of work which is to be backed out.

**Output parameters**

**RESPONSE**

is DFHFCDU’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**[REASON]**

is returned when RESPONSE is EXCEPTION, INVALID or DISASTER.

Possible values are:
FCDU INQUIRE function

This function causes an INQUIRE to be issued to the coupling facility data table in order to obtain information about the status of an active unit of work. If the BROWSE parameter is specified, then the function will return the status of the next unit of work in the browse.

Input parameters

POOL_ELEM_ADDR
is the address of the pool element which identifies the coupling facility data table pool for which the INQUIRE is to be issued. The inquire call will be issued to the CFDT server for this pool.

POOL_NAME
is the name of the coupling facility data table pool. The pool name is included for diagnostic purposes.

UOW_ID
identifies the unit of work for which status information is to be returned, or gives the previous unit of work in the browse.

UOW_RESTARTED
is an optional input parameter which indicates whether or not the inquire should select only units of work which have been through restart processing. It can take the values
YES|NO

BROWSE
is an optional parameter which specified whether the inquire is for a single unit of work or for the first or next UOW in a browse, and which can take the values
FIRST|NEXT

If the BROWSE parameter is omitted, the request is a single UOW inquire. The FIRST option indicates a search for a UOW id greater than or equal to the specified UOW_ID, and next indicates a search for a UOW id greater than the specified UOW_ID.

Output parameters

RETURNED_UOW_ID
Is the unit of work for which the browse is returning status information.

UOW_STATE
indicates the state of the unit of work, and can have the values:
IN_FLIGHT|IN_DOUBT|IN_COMMIT|IN_BACKOUT
**UOW_RESTART_STATE**
indicates whether the unit of work has been through restart processing.

**UOW RETAINED**
indicates whether the locks for the unit of work have been retained.

**RESPONSE**
is DFHFCDU’s response to the call. It can have any of these values:
OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

**[REASON]**
is returned when RESPONSE is EXCEPTION, INVALID or DISASTER.
Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>SERVER_CONNECTION_FAILED, RECOVERY_NOT_ENABLED, UOW_NOT_FOUND, CF_ACCESS_ERROR, CFDT_SYSIDERR, CFDT_SERVER_NOT_AVAILABLE, CFDT_SERVER_NOT_FOUND, CFDT_CONNECT_ERROR, CFDT_DISCONNECT_ERROR, RESYNC_RETRY_FAILED</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND, DISASTER_PERCOLATION</td>
</tr>
</tbody>
</table>

**FCDU RESTART function**
This function establishes recovery status for a coupling facility data table pool when a CICS region has successfully connected to it.

**Input parameters**

**POOL_ELEM_ADDR**
is the address of the pool element which identifies the coupling facility data table pool for recovery status is to be established. The RESTART call will be issued to the CFDT server for this pool.

**POOL_NAME**
is the name of the coupling facility data table pool. The pool name is included for diagnostic purposes.

**Output parameters**

**RETURNED_UOW_ID**
Is the unit of work for which the browse is returning status information.

**UOW_STATE**
indicates the state of the unit of work, and can have the values:
IN_FLIGHT | IN_DOUBT | IN_COMMIT | IN_BACKOUT

**UOW_RESTART_STATE**
indicates whether the unit of work has been through restart processing.

**UOW RETAINED**
indicates whether the locks for the unit of work have been retained.

**RESPONSE**
is DFHFCDU’s response to the call. It can have any of these values:
OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

**[REASON]**
is returned when RESPONSE is EXCEPTION, INVALID or DISASTER.
Possible values are:
**FCDY RESYNC_CFDT_POOL function**

This function causes a coupling facility data table pool to be resynchronized.

**Input parameters**

POOL_NAME

is the name of the coupling facility data table pool which is to be resynchronized.

**Output parameters**

RESPONSE

is DFHFCDY’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]

is returned when RESPONSE is EXCEPTION, INVALID or DISASTER.

Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>INITIATE_RECOVERY_FAILED, TERMINATE_RECOVERY_FAILED, CFDT_SERVER_CALL_FAILED</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND, DISASTER_PERCOLATION</td>
</tr>
</tbody>
</table>

**FCDY RESYNC_CFDT_LINK function**

This function causes a link between a unit of work and a coupling facility data table pool to be resynchronized.

**Input parameters**

POOL_NAME

is the name of the coupling facility data table pool for which the link is to be resynchronized.

UOW_ID

is the unit of work ID which identifies the link to be resynchronized.

**Output parameters**

RESPONSE

is DFHFCDY’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>INITIATE_RECOVERY_FAILED, TERMINATE_RECOVERY_FAILED, CFDT_SERVER_CALL_FAILED</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND, DISASTER_PERCOLATION</td>
</tr>
</tbody>
</table>
File control

[REASON]
is returned when RESPONSE is EXCEPTION, INVALID or DISASTER.
Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>INITIATE_RECOVERY_FAILED, TERMINATE_RECOVERY_FAILED, CFDT_SERVER_CALL_FAILED</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND, DISASTER_PERCOLATION</td>
</tr>
</tbody>
</table>

**FCDY RETURN_CFDT_ENTRY_POINTS function**

This function causes module DFHFCDY to return the entry point addresses of the other modules with which it is link-edited.

**Input parameters**
None

**Output parameters**

<table>
<thead>
<tr>
<th>CFDT_EP_DFHFCDW</th>
<th>is the entry point address of module DFHFCDW.</th>
</tr>
</thead>
<tbody>
<tr>
<td>CFDT_EP_DFHFCDU</td>
<td>is the entry point address of module DFHFCDU.</td>
</tr>
</tbody>
</table>

**RESPONSE**
is DFHFCDY’s response to the call. It can have any of these values:
OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON]
is returned when RESPONSE is INVALID or DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND, DISASTER_PERCOLATION</td>
</tr>
</tbody>
</table>

**FCFL END_UOWDSN_BROWSE function**

After a browse of all the data set failures within a unit of work, the END_UOWDSN_BROWSE function releases the storage that was used for a snapshot of the failures.

**Input parameters**

| BROWSE_TOKEN | is the token which was used for the browse. |

**Output parameters**

| RESPONSE | is DFHFCFL’s response to the call. It can have any of these values: OK | INVALID | DISASTER | PURGED |
|----------|-----------------------------------------------------------------------|

[REASON]
is returned when RESPONSE is INVALID or DISASTER. Possible values
are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>INVALID_BROWSE_TOKEN</td>
</tr>
<tr>
<td>DISASTER</td>
<td>DISASTER_PERCOLATION, ABEND</td>
</tr>
</tbody>
</table>

**FCFL FIND RETAINED function**

This function looks for any FLAB associated with the specified data set which is flagged as retained, indicating that there are retained locks associated with the data set.

**Input parameters**

**DSNAME**

is the 44-character name of the data set for which associated retained locks are to be found.

**Output parameters**

**RETLocks**

indicates whether or not there are retained locks associated with the data set, and can have either of these values:

RETAI NED|NORETAI NED

**RESPONSE**

is DFHFCFL’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]

is returned when RESPONSE is DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>DISASTER_PERCOLATION, ABEND</td>
</tr>
</tbody>
</table>

**FCFL FORCE_INDOUBTS function**

This function is used by the CEMT or EXEC CICS SET DSNAME() UOWACTION(COMMIT|BACKOUT|FORCE) command. Shunted in-doubt units of work are forced to complete in the specified direction. FORCE means that the direction is obtained from the ACTION specified on the transaction definition.

**Input parameters**

**DSNAME**

is the 44-character name of the data set for which shunted in-doubt units of work are to be forced to complete.

**DIRECTION**

is the direction in which the units of work are to complete: forwards (commit), backwards (backout), or heuristic (from the action specified on the transaction definition). It can have any of these values:

FORWARD|BACKWARD|HEURISTIC

**Output parameters**

**RESPONSE**

is DFHFCFL’s response to the call. It can have any of these values:
File control

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>DISASTER_PERCOLATION, ABEND</td>
</tr>
</tbody>
</table>

FCFL GET_NEXT_UOWDSN function

This function returns the failure information for the next data set that has a failure within the unit of work being browsed.

Input parameters

BROWSE_TOKEN
is the token for the browse, which was returned by a START_UOWDSN_BROWSE call.

Output parameters

DSNAME
is the 44-character name of the data set for which failure information is returned.

RLSACCESS
indicates whether the data set was last open in RLS or non-RLS access mode, and can have either of these values:
RLS|NOTRLS

CAUSE
indicates the cause of the failure, and can have any of these values:
CACHE|RLSSERVER|CONNECTION|DATASET|UNDEFINED

RETAINT_REASON
indicates the reason for the failure, and can have any of these values:
RLSGONE|COMMITFAIL|IOERROR|DATASETFULL|INDEXRECFULL|
OPENERROR|DELETEERROR|DEADLOCK|BACKUPNONBWO|LOCKSTRUCFULL|
FAILEDGBKOUT|NOTAPPLIC|RR_COMMITFAIL|RR_INDOUBT|

RESPONSE
is DFHFCFL’s response to the call. It can have any of these values:
OK|INVALID|EXCEPTION|DISASTER

REASON
is returned when RESPONSE is EXCEPTION, INVALID, or DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>END_OF_LIST</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_BROWSE_TOKEN</td>
</tr>
<tr>
<td>DISASTER</td>
<td>DISASTER_PERCOLATION, ABEND</td>
</tr>
</tbody>
</table>

FCFL RESET_BFAILS function

This function is used by the CEMT and EXEC CICS SET DSNAME() ACTION(RESETLOCKS) command. It purges shunted unit of work log records which hold backout-failure or commit-failure locks on the specified data set, and releases the locks.
**Input parameters**

**DSNAME**

is the 44-character name of the data set for which backout and commit failures are to be reset.

**Output parameters**

**RESPONSE**

is DFHFCFL’s response to the call. It can have any of these values:

- OK
- EXCEPTION
- DISASTER
- INVALID
- KERNERROR
- PURGED

[REASON]

is returned when RESPONSE is DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>DISASTER_PERCOLATION, ABEND, REMOVE_FAILURE</td>
</tr>
</tbody>
</table>

**FCFL RETRY function**

This function is used by the CEMT and EXEC CICS SET DSNAME() UOWACTION(RETRY) command. It drives retry of any failed backouts and commits for the specified data set, by informing DFHFCRR that the failed resource (that is, the data set) is now available.

**Input parameters**

**DSNAME**

is the 44-character name of the data set for which backout and/or commits are to be retried.

**Output parameters**

**RESPONSE**

is DFHFCFL’s response to the call. It can have any of these values:

- OK
- EXCEPTION
- DISASTER
- INVALID
- KERNERROR
- PURGED

[REASON]

is returned when RESPONSE is DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>DISASTER_PERCOLATION, ABEND, RESOURCE_NOT_FOUND</td>
</tr>
</tbody>
</table>

**FCFL START_UOWDSN_BROWSE function**

This function starts a browse of the data set failures within a unit of work. A snapshot of the failed data sets for the unit of work and the reasons for the failures are collected in an in-storage table to be browsed by the GET_NEXT_UOWDSN function.

**Input parameters**

**UOW**

is the 8-byte local unit of work identifier.

**Output parameters**

**BROWSE_TOKEN**

is a token which is used during the browse.
**File control**

**FCFL TEST_USER function**

This function is used to test if the task has updated a record, and therefore established itself as a file user, either for any data set or for a specified data set. It can be used either as a domain subroutine call or as an inline macro.

**Input parameters**

- **[ENVIRONMENT]**
  - is an optional parameter which is a fullword environment identifier. If specified, then the function will test whether the task is a user of any files within that environment.

- **[DSNAME]**
  - is an optional parameter which specifies that a particular data set is to be tested.

**Output parameters**

- **FLAB_PTR**
  - is the address of a FLAB which was found by the test. If a non-zero value is returned, then this means that the user is a task.

- **RESPONSE**
  - is DFHFCFL’s response to the call. It can have any of these values:
    - OK | EXCEPTION | DISASTER | PURGED

- **[REASON]**
  - is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>UOW_NOT_FOUND, NO_FLABS_FOUND</td>
</tr>
<tr>
<td>DISASTER</td>
<td>DISASTER_PERCOLATION, ABEND</td>
</tr>
</tbody>
</table>

**FCLJ FILE_OPEN function**

This function is called when a file is opened, and causes a ‘tie up record’ record to be written to the log of logs if either the file (or associated data set) is forward recoverable or if autojournalling is specified for the file, to the forward recovery log if the file (or associated data set) is forward recoverable, and to the autojournal if autojournalling is specified for the file.

**Input parameters**

- **FCTE_ADDRESS**
  - is the address of the file control table entry for the file being opened.
File control

Output parameters
RESPONSE is DFHFCLJ’s response to the call. It can have any of these values:
OK | INVALID | PURGED | DISASTER

[REASON] is returned when RESPONSE is DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LG_RETURNED_ERROR</td>
</tr>
</tbody>
</table>

FCLJ FILE_CLOSE Function
This function is called when a file is closed, and causes a file close log record to be written to the log of logs if either the file (or associated data set) is forward recoverable or if autojournaling is specified for the file, to the forward recovery log if the file (or associated data set) is forward recoverable, and to the autojournal if autojournaling is specified for the file.

Input parameters
FCTE_ADDRESS is the address of the file control table entry for the file being closed.

Output parameters
RESPONSE is DFHFCLJ’s response to the call. It can have any of these values:
OK | INVALID | PURGED | DISASTER

[REASON] is returned when RESPONSE is DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LG_RETURNED_ERROR</td>
</tr>
</tbody>
</table>

FCLJ READ_ONLY Function
This function causes a read_only log record to be written to an autojournal, if read-only autojournaling is specified on the file definition. The log record is built using the input parameters.

Input parameters
BASE_ESDS_RBA is the RBA of the record being read, if the file is an ESDS.
FCTE_ADDRESS is the address of the file control table entry for the file being read.
KEY_ADDRESS is the address of the key of the record being read.
KEY_LENGTH is the key length of the record being read.
RECORD_ADDRESS is the address of the record being read.
RECORD_LENGTH
is the length of the record being read.

SHUNTED
indicates whether or not the unit of work has ever been shunted (due to some failure during syncpoint). It can have either of these values:
YES | NO

Output parameters
RESPONSE
is DFHFCJ’s response to the call. It can have any of these values:
OK | INVALID | PURGED | DISASTER

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LG_RETURNED_ERROR, RM_RETURNED_ERROR</td>
</tr>
</tbody>
</table>

FCLJ READ_UPDATE Function
This function causes a read_update log record to be written to the system log, if the file is recoverable, and if the destination parameter specifies either LOG or BOTH. It causes a read_update log record to be written to the autojournal if journaling of read updates is specified on the file definition, and if the destination parameter specifies either JOURNAL or BOTH. The log record is built using the input parameters.

Input parameters
BASE_ESDS_RBA
is the RBA of the record being read for update, if the file is an ESDS.

FCTE_ADDRESS
is the address of the file control table entry for the file being read for update.

KEY_ADDRESS
is the address of the key of the record being read for update.

KEY_LENGTH
is the key length of the record being read for update.

RECORD_ADDRESS
is the address of the record being read for update.

RECORD_LENGTH
is the length of the record being read for update.

DESTINATION
specifies whether the log record is to be written to the autojournal, the system log, or both. It is used to suppress writing records that would otherwise be requested by the file definition. It can have any of these values:
JOURNAL | LOG | BOTH

SYNCHRONIZE_LOG
indicates whether or not the system log is to be synchronized (forced) when the log record is written. It can have either of these values:
YES | NO
SHUNTED
denotes whether or not the unit of work has ever been shunted (due to
some failure during syncpoint). It can have either of these values:
YES|NO

Output parameters

[LOG_TOKEN]
is an optional parameter which is returned if SYNCHRONIZE(NO) was
specified, and which contains a token to be used when subsequently
synchronizing (forcing) the system log.

RESPONSE
is DFHFCCLJ’s response to the call. It can have any of these values:
OK|INVALID|PURGED|DISASTER

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LG_RETURNED_ERROR, RM_RETURNED_ERROR</td>
</tr>
</tbody>
</table>

FCLJ WRITE_UPDATE Function

This function causes a write_update log record to be written to the forward
recovery log, if the file (or associated data set) is forward recoverable, and to the
autojournal, if journaling of write updates is specified on the file definition. A
write_update log record represents the completion of a file REWRITE request. The
log record is built using the input parameters.

Input parameters

BASE_ESDS_RBA
is the RBA of the record being rewritten, if the file is an ESDS.

FCTE_ADDRESS
is the address of the file control table entry for the file being rewritten to.

KEY_ADDRESS
is the address of the key of the record being rewritten.

KEY_LENGTH
is the key length of the record being rewritten to.

RECORD_ADDRESS
is the address of the record being rewritten.

RECORD_LENGTH
is the length of the record being rewritten.

SHUNTED
indicates whether or not the unit of work has ever been shunted (due to
some failure during syncpoint). It can have either of these values:
YES|NO

Output parameters

RESPONSE
is DFHFCCLJ’s response to the call. It can have any of these values:
OK|INVALID|PURGED|DISASTER
[REASON]

is returned when RESPONSE is DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LG_RETURNED_ERROR, RM_RETURNED_ERROR</td>
</tr>
</tbody>
</table>

**FCLJ WRITE_ADD Function**

This function causes a write_add log record to be written to the system log if the file is recoverable, and if the destination parameter specifies BOTH. It causes a write_add log record to be written to the autojournal if journaling of write adds was specified on the file definition. The log record is built using the input parameters.

**Input parameters**

- **BASE_ESDS_RBA**
  
is the RBA of the record being added, if the file is an ESDS.

- **FCTE_ADDRESS**
  
is the address of the file control table entry for the file being written to.

- **KEY_ADDRESS**
  
is the address of the key of the record being added.

- **KEY_LENGTH**
  
is the key length of the record being written to.

- **MASSINSERT**
  
indicates whether or not the record is being added as part of a mass insert.
  
It can have either of these values:
  
YES|NO

- **DESTINATION**
  
specifies whether the log record is to be written to the autojournal only, or to both the autojournal and the system log. It is used to suppress writing records that would otherwise be requested by the file definition. It can have either of these values:
  
JOURNAL|BOTH

- **RECORD_ADDRESS**
  
is the address of the record being added.

- **RECORD_LENGTH**
  
is the length of the record being added.

- **SHUNTED**
  
indicates whether or not the unit of work has ever been shunted (due to some failure during syncpoint). It can have either of these values:

YES|NO

**Output parameters**

- **RESPONSE**
  
is DFHFCIJ’s response to the call. It can have any of these values:

OK|INVALID|PURGED|DISASTER

- **[REASON]**
  
is returned when RESPONSE is DISASTER. Possible values are:
FCLJ WRITE_ADD_COMPLETE Function

This function causes a write_add_complete log record to be written to the forward recovery log if the file (or associated data set) is forward recoverable, and to the autojournal if write_add_complete journaling is specified on the file definition. It causes a truncated write_add_complete log record to be written to the system log if the file is a recoverable ESDS accessed in non-RLS mode. If MASSINSERT(YES) and MASSINSERT_STAGE(LAST) are specified, then only the system log record is written, and not the forward recovery log or autojournal record. The log record is built using the input parameters.

**Input parameters**

**BASE_ESDS_RBA**
- is the RBA of the record that has been added, if the file is an ESDS.

**FCTE_ADDRESS**
- is the address of the file control table entry for the file that has been written to.

**KEY_ADDRESS**
- is the address of the key of the record which has been added.

**KEY_LENGTH**
- is the key length for the file which has been written to.

**MASSINSERT**
- indicates whether or not the record was added as part of a mass insert. It can have either of these values:
  - YES
  - NO

**[MASSINSERT_STAGE]**
- is an optional parameter which indicates whether the record is either the first or last record added during a massinsert sequence. It can have either of these values:
  - FIRST
  - LAST

**RECORD_ADDRESS**
- is the address of the record which has been added.

**RECORD_LENGTH**
- is the length of the record which has been added.

**SHUNTED**
- indicates whether or not the unit of work has ever been shunted (due to some failure during syncpoint). It can have either of these values:
  - YES
  - NO

**Output parameters**

**RESPONSE**
- is DFHFLJ’s response to the call. It can have any of these values:
  - OK
  - INVALID
  - PURGED
  - DISASTER

**[REASON]**
- is returned when RESPONSE is DISASTER. Possible values are:

---

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LG_RETURNED_ERROR, RM_RETURNED_ERROR</td>
</tr>
</tbody>
</table>
### FCLJ WRITE_DELETE Function

This function causes a write_delete log record to be written to the forward recovery log if the file (or associated data set) is forward recoverable, and to the autojournal if journaling of write_deletes is specified on the file definition. The log record is built using the input parameters.

#### Input parameters

- **BASE_ESDS_RBA**
  - is the RBA of the record being deleted, if the file is an ESDS.

- **FCTE_ADDRESS**
  - is the address of the file control table entry for the file.

- **KEY_ADDRESS**
  - is the address of the key of the record being deleted.

- **KEY_LENGTH**
  - is the key length for the file.

- **BASE_KEY_ADDRESS**
  - is the address of the base key of the record being deleted, which is used if the data set is being accessed via a path.

- **SHUNTED**
  - indicates whether or not the unit of work has ever been shunted (due to some failure during syncpoint). It can have either of these values: YES|NO

#### Output parameters

- **RESPONSE**
  - is DFHFCLJ’s response to the call. It can have any of these values: OK|INVALID|PURGED|DISASTER

- **[REASON]**
  - is returned when RESPONSE is DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LG_RETURNED_ERROR, RM_RETURNED_ERROR</td>
</tr>
</tbody>
</table>

### FCLJ SYNCHRONIZE_READ_UPDATE Function

This function causes any log records previously written to the system log for this file to be synchronized (forced). The log token returned on a previous call to write a log record for this file is supplied as input.

#### Input parameters

- **FCTE_ADDRESS**
  - is the address of the file control table entry for the file being read for update.

- **LOG_TOKEN**
  - is the token returned on a previous call. The system log record written by the previous call, plus any log records written prior to that, are hardened.
Output parameters

RESPONSE is DFHFCLEJ’s response to the call. It can have any of these values:
- OK
- INVALID
- PURGED
- DISASTER

[REASON] is returned when RESPONSE is DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, RM_RETURNED_ERROR</td>
</tr>
</tbody>
</table>

FCLJ TAKE_KEYPOINT Function

Provided that BWO copy is supported by this CICS (indicated by a flag in file control static storage), then this function performs a scan of the file control table and, unless it has been called within the last half hour, writes a tie up record for each file open for update in non-RLS mode that is BWO-eligible and forward recoverable to the forward recovery log.

A tie up record specifies which CICS system within the sysplex opened the file, and the data set which the file was opened against. Tie up records are used by forward recovery utilities, for example CICSVR.

Input parameters
None

Output parameters

KEYPOINT_TAKEN indicates whether or not the set of tie up records was successfully written. It can have either of these values:
- YES
- NO

RESPONSE is DFHFCLEJ’s response to the call. It can have any of these values:
- OK
- INVALID
- PURGED
- DISASTER

[REASON] is returned when RESPONSE is DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LG_RETURNED_ERROR, TM_GETNEXT_FCTE_FAILED</td>
</tr>
</tbody>
</table>

FCLJ DATASET_COPY Function

This function is called when DFSMSdss initiates a copy of an RLS data set via the VSAM RLS quiesce mechanism. The function causes a ‘tie up record’ to be written to the log of logs if either the data set is forward recoverable, or some flavor of autojournaling has been specified in the file definition. In addition, if applicable, a record is written to the forward recovery log.

A tie up record specifies which CICS system within the sysplex opened the file, and the data set which the file was opened against. Tie up records are used by forward recovery utilities, for example CICSVR.
**Input parameters**

**FCTE_ADDRESS**

is the address of the file control table entry for the file associated with a data set being copied.

**Output parameters**

**RESPONSE**

is DFHFLJ’s response to the call. It can have any of these values:

OK|INVALID|PURGED|DISASTER

**[REASON]**

is returned when RESPONSE is DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LG_RETURNED_ERROR</td>
</tr>
</tbody>
</table>

**FCQI INITIATE_QUIESCE Function**

This function takes a quiesce request of type QUIESCE, IMMQUIESCE, UNQUIESCE, QUIESCE_CANCEL, NONBWO_CANCEL or BWO_CANCEL and creates a FC Quiesce Send Element (FCQSE) to describe the request. The FCQSE is added to a chain anchored in FC static, and an ECB associated with the chain (also in FC static) is posted. DFHFCQI then either suspends until the quiesce request completes or returns immediately to its caller, depending on whether busy WAIT or NOWAIT was specified on the call.

When DFHFCQI posts the ECB, the CFQS transaction (DFHFCQS) wakes up and processes the FCQSE on the chain, calling DFHFCCA QUIESCE to issue the appropriate flavor of IDAQUIES macro to SMSVSAM. When the IDAQUIES has completed, DFHFCQS will resume DFHFCQI if it was suspended, communicating the results of the IDAQUIES via the FCQSE. The FCQSE can then be unchained and freed.

**Input parameters**

**QUIESCE_TYPE**

indicates the type of quiesce being initiated and can have any of these values:

QUIESCE|IMMQUIESCE|UNQUIESCE|NONBWO_CANCEL|BWO_CANCEL|QUIESCE_CANCEL

**DSNAME**

is the 44-character name of the base data set to be quiesced.

**BUSY**

indicates whether DFHFCQI is to wait for the quiesce to complete, or is to return immediately to the caller, and can take either of these values:

WAIT|NOWAIT

**SOURCE**

indicates whether the source of the quiesce request was CICS or a user, and can take either of these values:

CICS|USER

**Output parameters**

**RESPONSE**

is DFHFCQI’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED
[REASON] is returned when RESPONSE is INVALID, EXCEPTION or DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>INVALID_QUIESCE_TYPE</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>NOT_SUPPORTED, UNKNOWN_VSAM_DATASET, QUIESCE_NOT_POSSIBLE, UNQUIESCE_NOT_POSSIBLE, CANCELLED, TIMED_OUT, IOERR, SERVER_FAILURE, DATASET_MIGRATED</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND, CATALOG_ERROR, DISASTER_PERCOLATION</td>
</tr>
</tbody>
</table>

**FCQI INQUIRE_QUIESCE Function**

This function returns the quiesce state of a data set as QUIESCED, UNQUIESCED, or QUIESCING. DFHFCAT is called to inquire on the state of the ‘quiesced’ bit in the VSAM ICF catalog, which will return QUIESCED or UNQUIESCED. If UNQUIESCED is returned, the FCQSE chain is then scanned to find an FCQSE specifying the data set in question. If such an FCQSE is found for a quiesce or immquiesce request then a state of QUIESCING is returned. There is no UNQUIESCING state as the unquiesce operation is far quicker than quiesce.

**Input parameters**

DSNAME  
is the 44-character name of the base data set for which quiesce state information is to be returned.

**Output parameters**

QUIESCESTATE  
indicates the quiesce state of the data set, and can have any of these values:

QUIESCED|UNQUIESCED|QUIESCING

RESPONSE  
is DFHFCQI’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NOT_SUPPORTED, UNKNOWN_VSAM_DATASET, IOERR</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND, CATALOG_ERROR, DISASTER_PERCOLATION</td>
</tr>
</tbody>
</table>

**FCQI COMPLETE_QUIESCE Function**

This function is invoked whenever CICS has finished the processing for those quiesce requests for which SMSVSAM must be notified with an IDAQUIES QUICMP. Such quiesce requests are VSAM QUICLOSE (quiesce), QIOCOPY (non-BWO backup) and QUIBWO (BWO backup). This is achieved by calling DFHFCCA QUIESCE_COMPLETE to issue the IDAQUIES QUICMP macro to SMSVSAM.
Input parameters

**DSNAME**

is the 44-character name of the base data set for which quiesce processing has been completed by this CICS.

**QUIESCE_TOKEN**

is the token which was supplied by SMSVSAM when it drove the quiesce exit for the original quiesce request, and which must be returned on the IDAQUIES QUICMP.

Output parameters

**RESPONSE**

is DFHFCQI’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]

is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>IOERR, SERVER_FAILURE</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND, DISASTER_PERCOLATION</td>
</tr>
</tbody>
</table>

FCQR RECEIVE_QUIESCES Function

This function consists of a forever loop around a dispatcher wait on an ECB. It receives work from the CICS RLS quiesce exit DFHFCQX whenever SMSVSAM requires CICS to perform processing for a quiesce request. DFHFCQX queues the request to DFHFCQR by adding an FC Quiesce Receive Element (FCQRE) to a chain anchored in file control static storage, and posting the ECB associated with the chain, also in FC static.

The posting of the ECB wakes the CFQR transaction, which executes the code in DFHFCQR. The FCQREs on the chain are processed, and DFHFCQU is called with function PROCESS_QUIESCE to perform the actual work. The ECB might also be posted to inform DFHFCQR that CICS is terminating. When DFHFCQU has finished processing, DFHFCQR unchains and frees the FCQRE.

Input parameters

None.

Output parameters

**RESPONSE**

is DFHFCQR’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]

is returned when RESPONSE is DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, PROCESS_QUIESCE_ERROR, DISASTER_PERCOLATION</td>
</tr>
</tbody>
</table>
FCQS SEND_QUIESCES Function

This function consists of a forever loop around a dispatcher wait on a list of ECBs. Work is received from tasks that wish to send a quiesce request to SMSVSAM. Such tasks call DFHFCQI with function INITIATE_QUIESCE, which queues the request to DFHFCQS by adding an FC Quiesce Send Element (FCQSE) to the chain anchored in file control static storage, and posting an ECB associated with the chain, also in FC static.

When the ECB is posted, it wakes the CFQS transaction, which executes the code in DFHFCQS. The FCQSEs on the chain are processed, and DFHFCCA is called with function QUIESCE_REQUEST to issue the appropriate flavor of IDAQUIES macro to SMSVSAM. This is an asynchronous operation, and SMSVSAM returns the address of an ECB that will be posted when the IDAQUIES completes. This is saved in the FCQSE.

DFHFCQS then goes back into its dispatcher wait. It is actually waiting on a list of ECBs, the ECB for the chain plus an ECB for each IDAQUIES request. It wakes and processes the chain whenever one of these ECBs is posted. The wait also specifies a timeout interval, so that IDAQUIES requests that hang can be detected. When DFHFCQS wakes up, this can mean that: there is new work on the chain, or a quiesce request has completed, or a quiesce request timed out, or CICS is terminating. When a quiesce request has completed or timed out, DFHFCQS will resume the initiating task if it is waiting, after issuing appropriate messages and invoking global user exit XFCQUIS if active.

Input parameters
None.

Output parameters
RESPONSE is DFHFCQS’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, TIMEOUT_CANCEL_ERROR, DISASTER_PERCOLATION</td>
</tr>
</tbody>
</table>

FCQU PROCESS_QUIESCE Function

DFHFCQU PROCESS_QUIESCE is called whenever a quiesce request is received from VSAM RLS. The quiesce exit DFHFCQX queues requests to the CFQR system transaction (DFHFCCR), which calls DFHFCQU to process each one in turn. The PROCESS_QUIESCE function is also called to implement a non-RLS variant of QUIESCE called NON_RLS_CLOSE. This is for non-RLS files, is only used internally by CICS, and does not run under the CFQR system transaction. Each quiesce request type is processed in a different way by DFHFCQU.

QUIESCE corresponds to an SMSVSAM QUICLOSE. All files open against the data set are closed, the file state of each file is set to unenabled but with a flag that says re-enable on QUIOPEN, and a QUICMP is issued for the QUICLOSE back to VSAM RLS to indicate our QUICLOSE processing is complete. The immediate option on the DFHFCQU call governs how file
closes are to be performed. If NO or omitted then closes will occur when all UOWs using the data set have completed normally. If YES then all such UOWs will be force purged to speed things up.

UNQUIESCE

 corresponds to an SMSVSAM QUIOPEN. All files associated with the data set are checked to see if the file state requires resetting back to enabled, because it had been set unenabled by a QUICLOSE.

NONBWO_START

 corresponds to an SMSVSAM QUICOPY. CICS prepares for a non-BWO backup of the data set by preventing new units of work from updating the data set, allowing existing UOWs to finish updating the data set, and then issuing a QUICMP for the QUICOPY back to SMSVSAM to indicate that QUICOPY processing is complete. The files involved are not closed.

NONBWO_END

 corresponds to an SMSVSAM QUICEND. All files associated with the data set are checked to see if the file state requires resetting back to enabled because it had been set unenabled by an OPEN failure, and a set of ‘tie up records’ are written for the data set.

BWO

 corresponds to an SMSVSAM QUIBWO. CICS prepares for a BWO backup of the data set by writing a set of ‘tie up records’ allowing existing units of work to finish updating the data set, and then issuing a QUICMP for the QUIBWO back to SMSVSAM to indicate that QUIBWO processing is complete. The files involved are not closed, nor are updates prevented.

BWO_END

 corresponds to an SMSVSAM QUIBEND. The only processing involved is to stop an existing BWO quiesce if one is in progress.

LOST_LOCKS_RECOVERED

 corresponds to an SMSVSAM QUILLRC. It notifies CICS that lost locks recovery has been completed for the data set throughout the sysplex. DFHFCRR is called with function LOST_LOCKS_RECOVERED to process the availability of the data set.

FORWARD_RECOVERY_COMPLETE

 corresponds to an SMSVSAM QUIFRC. It notifies CICS that forward recovery has been completed for the data set. DFHFCRR is called with function RESOURCE_AVAILABLE to process the availability of the data set.

CACHE_AVAILABLE

 corresponds to an SMSVSAM QUICA. It notifies CICS that a previously failed cache structure is now available. DFHFCRR is called with function RESOURCE_AVAILABLE to process the availability of the cache.

NON_RLS_CLOSE

 processes a non-RLS variant of type CLOSE called NON_RLS_CLOSE. All ACBs open against the specified non-RLS data set are closed.

Some of the requests cause global user exit XFCVSDS to be invoked if active and a DSNB exists for the data set, and XFCVSDS can suppress certain of the requests if desired. Suppression causes the quiesce request to be cancelled throughout the sysplex (by issuing the inverse quiesce request).

The types of quiesce that DFHFCQU can receive fall into two ‘completion’ categories.
1. Those for which VSAM does not require completion notification. For these no IDAQUIES QUICMP is issued. The successful return of the quiesce exit DFHFCQX to VSAM is sufficient. The requests in this category are:
   UNQUIESCE, NONBWO_END, BWO_END, CACHE_AVAILABLE, LOCKS_RECOVERY_COMPLETE, FORWARD_RECOVERY_COMPLETE.

2. Those for which VSAM requires completion notification because CICS must complete some critical processing. For these an IDAQUIES QUICMP must be issued when CICS processing is complete. The requests in this category are:
   QUIESCE, NONBWO_START, BWO_START.

**Input parameters**

**QUIESCE_TYPE**
indicates the type of quiesce being requested. It can have any of these values:
   QUIESCE|UNQUIESCE|NONBWO_START|NONBWO_END|BWO_START|BWO_END|LOCKS_RECOVERY_COMPLETE|FORWARD_RECOVERY_COMPLETE|CACHE_AVAILABLE|NON_RLS_CLOSE

**DSNAME|CACHE_NAME**
either specifies the 44-character name of the data set to which the quiesce request applies, or (when the quiesce_type is CACHE_AVAILABLE) the 16-character name of the cache structure which has become available.

**[IMMEDIATE]**
applies when the quiesce_type is QUIESCE or NON_RLS_CLOSE, and indicates whether units of work which have updated the data set will be forced to complete immediately, or whether the request will wait for such units of work to complete naturally. It can have either of these values:
   YES|NO

**[CONCURRENT]**
applies when the quiesce_type is NONBWO_START or BWO_START, and indicates whether the concurrent copy technique is being used. It is purely informational, and has no effect on the processing. It can have either of these values:
   YES|NO

**[QUIESCE_TOKEN]**
is a token which is supplied by SMSVSAM when certain quiesce requests are initiated, and must be passed back when the quiesce complete is issued.

**Output parameters**

**RESPONSE**
is DFHFCQU’s response to the call. It can have any of these values:
   OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**[REASON]**
is returned when RESPONSE is INVALID, EXCEPTION or DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>INVALID_QUIESCE_TYPE</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>DSNB_NOT_FOUND</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND, DISASTER_PERCOLATION, DFHFCRR_ERROR, DFHFCQI_ERROR, DFHFCFS_ERROR, DFHTM_FAILURE</td>
</tr>
</tbody>
</table>
**FCRR RESTART_RLS Function**

This function performs a restart of the RLS component of file control. The exact processing depends on the type of restart being performed.

**COLD and INITIAL**
The RLS control ACB is registered, and RLS is cold started, both via calls to DFHFCCA.

**WARM and EMERGENCY**
The RLS control ACB is registered, and recovery information is inquired upon from SMSVSAM, both via calls to DFHFCCA. If the recovery information indicates that there are data sets in lost locks status, then the corresponding DSNBs are marked as being lost locks, and preparation for lost locks recovery is carried out. Any orphan locks are eliminated.

**DYNAMIC**
This type of restart occurs when a new instance of the SMSVSAM server becomes available following a previous server failure.

Having waited for file control restart to complete if it was still in progress, and for any in-progress dynamic RLS restart to complete, RLS access is drained if this has not already been done, the control ACB is registered, and recovery information is inquired upon from SMSVSAM, all three via calls to DFHFCCA. If the recovery information indicates that there are data sets in lost locks status, then the corresponding DSNBs are marked as being lost locks, and preparation for lost locks recovery is carried out. Any orphan locks are eliminated. The CICS recovery manager is called to unshunt any units of work that are backout-failed due to the SMSVSAM server failure or a general file backout failure, and any units of work that are commit-failed due to the SMSVSAM server failure.

**Input parameters**

**TYPE_OF_RESTART**
indicates the type of RLS restart being performed, and can have any of these values:

- COLD
- WARM
- EMERGENCY
- DYNAMIC

**Output parameters**

**RESPONSE**
is DFHFCRR’s response to the call. It can have any of these values:

- OK
- EXCEPTION
- DISASTER
- INVALID
- KERNERROR
- PURGED

**[REASON]**
is returned when RESPONSE is INVALID, EXCEPTION or DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION, INVALID_RESTART_TYPE</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>REGISTER_CTL_ACB_FAILED, COLD_START_RLS_FAILED, DRAIN_RLS_FAILED, LOST_LOCKS_INFO_LOST,</td>
</tr>
<tr>
<td></td>
<td>INQUIRE_RECOVERY_FAILED, LOST_LOCKS_COMPLETE_FAILED, ORPHAN_RELEASE_FAILED</td>
</tr>
<tr>
<td>DISASTER</td>
<td>DSSR_FAILED, TM_LOCATE_FAILED, TM_UNLOCK_FAILED, ABEND, DISASTER_PERCOLATION</td>
</tr>
</tbody>
</table>
FCRR RESOURCE_AVAILABLE function

This function causes the CICS recovery manager to be notified of the availability of the specified resource. When the resource_type is DSET, an RMRE AVAIL call is issued for the specified data set. When the resource_type is CACHE, an RMRE avail call is issued for every data set that has outstanding work shunted due either to a cache failure or to a general file backout failure. When the resource_type is OTHER, an RMRE AVAIL call is issued for the specified resource.

Input parameters

**RESOURCE_TYPE**

is the type of resource which has become available, and can have any of these values:

DSET | CACHE | OTHER

**RESOURCE_NAME**

is the 44-character field containing the name of the resource which has become available.

**RESOURCE_NAME_LENGTH**

is a halfword containing the actual length of the resource name.

Output parameters

**RESPONSE**

is DFHFCRR’s response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON]

is returned when RESPONSE is INVALID or DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION, INVALIDRESOURCE_TYPE</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND, DISASTER_PERCOLATION</td>
</tr>
</tbody>
</table>

FCRR LOST_LOCKS_RECOVERED function

This function is called when lost locks recovery for a data set has been completed by all CICS regions that were sharing it, and causes the flag in the DSNB which indicates that the data set is in lost locks state to be cleared.

Input parameters

**RESOURCE_NAME**

is the 44-character field containing the name of the resource (data set) for which lost locks recovery has been completed.

Output parameters

**RESPONSE**

is DFHFCRR’s response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON]

is returned when RESPONSE is INVALID, EXCEPTION or DISASTER. Possible values are:
### File Control

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>SPHERE_UNKNOWN</td>
</tr>
<tr>
<td>DISASTER</td>
<td>TM_LOCATE_FAILED, TM_UNLOCK_FAILED, ABEND, DISASTER_PERCOLATION</td>
</tr>
</tbody>
</table>

### File Control’s call back gates

Table 56 summarizes file control’s call back gates. It shows the FC level-1 trace point IDs of the modules providing the functions for the gate, the functions provided by the gate, and the format for calls to the gate.

**Table 56. File control’s call back gates**

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>RMRO</td>
<td>FC 0BE0</td>
<td>PERFORM_PREPARE</td>
<td>RMRO</td>
</tr>
<tr>
<td></td>
<td>FC 0BE1</td>
<td>PERFORM_COMMIT</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>START_BACKOUT</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>DELIVER_BACKOUT_DATA</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>END_BACKOUT</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PERFORM_SHUNT</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PERFORM_UNSHUNT</td>
<td></td>
</tr>
<tr>
<td>RMKP</td>
<td>FC 0BE0</td>
<td>TAKE_KEYPOINT</td>
<td>RMKP</td>
</tr>
<tr>
<td></td>
<td>FC 0BE1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RMLK</td>
<td>FC 24A0</td>
<td>PREPARE</td>
<td>RMLK</td>
</tr>
<tr>
<td></td>
<td>FC 24A1</td>
<td>COMMIT</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>SEND_DO_COMMIT</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>SHUNT</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>UNSHUNT</td>
<td></td>
</tr>
<tr>
<td>RMDE</td>
<td>FC 0BE0</td>
<td>START_DELIVERY</td>
<td>RMDE</td>
</tr>
<tr>
<td></td>
<td>FC 0BE1</td>
<td>DELIVER_RECOVERY</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>DELIVER_FORGET</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>END_DELIVERY</td>
<td></td>
</tr>
<tr>
<td>LGGL</td>
<td>FC 2350</td>
<td>ERROR</td>
<td>LGGL</td>
</tr>
<tr>
<td></td>
<td>FC 2351</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DMEN</td>
<td>FC 0BD0</td>
<td>NOTIFY_SMSVSAM_AVAILABLE</td>
<td>DMEN</td>
</tr>
<tr>
<td></td>
<td>FC 0BD1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

You can find descriptions of these functions and their input and output parameters, in the chapters on the recovery manager, log manager, and domain manager.

The functions of the RMRO gate are processed by DFHFCRC. For PERFORM_PREPARE and PERFORM_COMMIT, DFHFCRC performs prepare and commit processing respectively for any file resources involved in the unit of work. For START_BACKOUT, DELIVER_BACKOUT_DATA and END_BACKOUT, DFHFCRC backs out changes made to file resources by the unit of work. For PERFORM_SHUNT and PERFORM_UNSHUNT, DFHFCRC respectively shunts and unshunts the file control structures representing recoverable parts of the unit of work.
The functions of the RMKP gate are processed by DFHFCRC. For TAKE_KEYPOINT, DFHFCRC performs processing required for forward recovery of BWO-eligible non-RLS files.

The functions of the RMLK gate are processed by DFHFCDW, which performs syncpoint and recovery functions for recoverable coupling facility data tables.

The functions of the RMDE gate are passed through by DFHFCRC to DFHFCIR. For START_DELIVERY, DFHFCIR takes no action. For DELIVER_RECOVERY and DELIVER_FORGET, DFHFCIR uses the log records that are delivered to it to rebuild file control structures representing the recoverable parts of each unit of work, and also rebuilds locks for non-RLS files. For END_DELIVERY, DFHFCIR notifies file control that the rebuilding of recovery information at CICS restart is now complete.

The functions of the LGGL gate are processed by DFHFCLF. For ERROR, DFHFCLF takes actions to handle a log stream failure for a general log used by file control.

The functions of the DMEN gate are processed by DFHFCES. For NOTIFY_SMSVSAM_AVAILABLE, DFHFCES calls DFHFCRR with a function of RESTART_RLS and TYPE_OF_RESTART as DYNAMIC.

### Exits

The following global user exit points are provided for file control:

- **In DFHEIFC**
  - XFCREQ and XFCREQC

- **In DFHFCFS**
  - XFCSREQ and XFCSREQC

- **In DFHFCN**
  - XFCNREC

- **In DFHFCRC**
  - XFCBFAIL, XFCBOUT, XFCBOVER and XFCLDEL

The following global user exit points are provided specifically for data table services: XDTAD, XDTLC, and XDTRD.

See the [CICS Customization Guide](#) for further information.

### Trace

The following point IDs are provided for file control:

- AP 04xx, for which the trace levels are FC 1, FC 2, and Exc
- AP 0Bxx, for which the trace levels are FC 1, FC 2, and Exc.
- AP 23xx, for which the trace levels are FC 1, FC 2, and Exc.
- AP 24xx, for which the trace levels are FC 1, FC 2, and Exc.

**Note:** Trace entries for shared data table services have point IDs at the lower end of the AP 0Bxx range, and a corresponding trace level of FC 2. Trace entries for coupling facility data tables are from AP 2440 upwards.

For more information about the trace points, see the [CICS Trace Entries](#). For more information about using traces in problem determination, see the [CICS Problem Determination Guide](#).
Chapter 41. Front end programming interface (FEPI)

The front end programming interface (FEPI) is an integral part of CICS Transaction Server for z/OS, Version 2 Release 1. The function is called a front end programming interface because it enables you to write CICS application programs that access other CICS or IMS programs. In other words, it provides a front end to those programs.

Design overview

This section describes how FEPI works at a high level. It discusses how the FEPI functions are provided within CICS.

FEPI as a CICS transaction

The main functions of FEPI are provided through the CSZI transaction, which is defined in group DFHFEPI. CSZI runs the FEPI Resource Manager, which is responsible for most of the functions of FEPI.

The FEPI Resource Manager transaction is attached during a late stage of CICS initialization. CSZI runs as a high-priority CICS system task, and cannot be canceled by an operator; it is terminated during CICS shutdown processing.

The FEPI commands communicate with the Resource Manager through the FEPI adapter program, which is loaded when CICS initializes, and is part of the CICS nucleus.

The FEPI adapter receives information from FEPI commands through two EXEC stubs, DFHESZ and DFHEIQSZ. DFHESZ handles the FEPI application programming commands, while DFHEIQSZ handles the system programming commands.

These two EXEC stubs call the adapter to do FEPI work. The adapter communicates with the Resource Manager through work queues. See "Application flows" for details of these flows.

Application flows

"FEPI as a CICS transaction" outlined the main components of FEPI. This section shows the pathways followed by a FEPI command.

Application programming command flows

The FEPI application programming commands flow through the normal EXEC CICS route into DFHEIP, from where they are routed to DFHESZ. DFHESZ passes the command parameter list to the FEPI adapter. After checking and other processing, the adapter generates another parameter list in internal format, and places it on a queue for the FEPI Resource Manager to process.

While the adapter is waiting for the Resource Manager to process the command, it issues a wait. The event control block (ECB) for this wait is contained in the parameter list queued to the Resource Manager. Consequently, the application that issued the FEPI command is in a wait state while the Resource Manager is processing the FEPI command. For information about wait processing, see the CICS Problem Determination Guide.
**Front end programming interface (FEPI)**

When the Resource Manager has retrieved the command from its queue, and processed it, the ECB is posted, thus ending the wait.

Control returns from the adapter to DFHEIP, and the application program in the normal fashion.

Figure 62 shows this processing. Note that the details are for illustration only.

---

**System programming command flows**

The FEPI system programming commands flow through DFHEIQSZ rather than DFHESZ, but the overall picture is the same as for FEPI application programming requests.

However, some system commands can flow directly to the FEPI Resource Manager, bypassing the EXEC stub. These commands are mainly concerned with FEPI processing to be done at “special” events, such as task termination and CICS shutdown.

Figure 63 on page 659 shows this processing. The details are for illustration only.
Logic flow within the FEPI adapter

Figure 64 on page 661 shows the logic flow within the FEPI adapter in more detail. In particular, it shows the points at which the FEPI global user exits, XSZBRQ and XSZARQ, and the FEPI journaling function, are invoked.

Journaling of data occurs after the Resource Manager has processed the request, but before XSZARQ is called (if active). Data is not journaled if your XSZBRQ exit program rejects the request.
The FEPI adapter and Resource Manager

The FEPI adapter runs as part of the invoking CICS task, and so runs under the QR task control block (TCB). The FEPI Resource Manager, running as CSZI, runs under the SZ TCB (reserved for use by the Resource Manager).

Consequently, the interface between the adapter and the Resource Manager uses waits and queues to synchronize access. The control block used to pass information between the adapter and the Resource Manager is called the DQE.

Figure 65 on page 661 shows this interaction. The details are for illustration only.
The FEPI Resource Manager work queues

When organizing its work, the FEPI Resource Manager uses a mechanism that is optimized for the FEPI environment. Each DQE is chained to a queue representing the work to be done next.

The most common mechanism used for this movement between queues is the connection on which the original FEPI command is operating.

**Summary of Resource Manager work queues**

In addition to the application queue, there are other queues used only by the Resource Manager. They are:

- **API/Norm**
  - Used for FEPI application requests

- **API/Expd**
  - Used for FEPI high-priority application requests

- **PRB**
  - Used for Resource Manager internal work

- **PRB/Time**
  - Used for Resource Manager internal time-dependent work

- **IRB**
  - Used to control work done in VTAM exits

- **IRB/Time**
  - Used to control time-dependent work done in VTAM exits

- **TPEND8**
  - Used to process VTAM TPEND8 conditions

- **Timer**
  - Used to control timer-related work

- **Free**
  - Used to hold VTAM RBs that have to be freed
Front end programming interface (FEPI)

**Discard**
Used to control requests initiated by FEPI DISCARD commands.

**CICS work**
Used to schedule work that has to run under the CICS QR TCB.

Control blocks

This section lists some of the FEPI control blocks and their resident storage subpools, where applicable. For details of the subpools, see Chapter 81, Storage manager domain (SM) on page 1097.

**DFHSZSDS (Static area)**
Used to anchor all FEPI storage

**DFHSZDCM (Common area)**
Used to anchor all FEPI Resource Manager storage (SZSPFCCM)

**DFHSZDND (Node)**
Represents a node (SZSPFCND)

**DFHSZDPD (Pool)**
Represents a pool (SZSPFCPD)

**DFHSZDTD (Target)**
Represents a target (SZSPFCTD)

**DFHSZDPS (Propertyset)**
Represents a property set (SZSPFCPS)

**DFHSZDCD (Connection)**
Represents a connection (a node-target pair) (SZSPFCCD)

**DFHSZDCV (Conversation)**
Represents a FEPI conversation (SZSPFCCV)

**DFHSZDSR (Surrogate)**
Used to associate nodes, pools, and targets with other control blocks—not to be confused with a CICS surrogate terminal (SZSPFCSR)

**DFHSZDQE (Queue element)**
Used to schedule Resource Manager work (SZSPFCWE).

Some of the relations between FEPI control blocks are shown in Figure 66 on page 663.
This section documents the areas that can be listed by the FEPI dump routines. For information about how to use these facilities for problem determination, see the CICS Problem Determination Guide.

**Note:** The length of areas described in this section may change in future versions or releases of CICS. Any status values interpreted may also be open to change. So you should use diagrams and descriptions in this section only as illustrations of how to interpret FEPI dumps.

Here is a list all the FEPI areas that can be interpreted. If an area does not exist in your system, it does not appear in the dump—no error message is produced.

- The static area
- The common area:

![FEPI control block relationships](image-url)
Front end programming interface (FEPI)

- The temporary ACB.
- Property sets
- Pools:
  - Connections within the pool
  - Node surrogates chained to the pool
  - Target surrogates chained to the pool
  - Queued allocate DQEs waiting within the pool
- Nodes:
  - Connections used by the node
  - Pool surrogates chained to the node
  - Node’s ACB
  - Node’s RPL
  - Unsolicited BINDs queued to the node
- Targets:
  - Connections used by the target
  - Connections queueing on the target
  - Pool surrogates chained to the target
- Connections:
  - Current API request
  - Connection’s RPL
  - Connection’s RESP data
  - Formatted data extension:
    - Graphics plane
    - Attributes
    - Highlights
    - Color
    - Selection
    - Validation
- Active conversations
- Browse conversations
- Inactive conversations
- CICS work queues
- PRB DQEs
- PRB time DQEs
- IRB DQEs
- IRB time DQEs
- T Pend8 DQEs
- Discard DQEs
- API normal DQEs
- API expd DQEs
- Timer DQEs
- Free RBs
- The stacks (level 2 only).

A DQE is interpreted further, as follows:
- The DRP representing the DQE
- The DQE associated storage
- Any horizontal DQE extension (chained) DQEs.

The following sections describe some of the areas interpreted.
Front end programming interface (FEPI)

The static area

This shows the status of the FEPI system; that is, whether or not it was running.

The common area

This shows whether the FEPI Resource Manager was active.

This shows details of the currently executing item.

This shows whether one of the VTAM exits was active (none in this case). If an exit was active, it is shown preceded by an equals (=) character.

Property sets

This shows details of the property set, which defines the characteristics of FEPI pools.

Pools

Chapter 41. Front end programming interface (FEPI)
Front end programming interface (FEPI)

1. Refers to the property set that defines the characteristics of the pool.
2. Shows the connections within the pool.
3. Shows the node surrogates chained to the pool.
4. Shows the pool surrogates chained to the pool.
5. There are no ACB, RPL, and unsolicited bind areas for this pool (no queued allocated DQEs within the pool). If there were, the areas would be shown here.

Nodes

**SZ.Node FEPI Nodes**

<table>
<thead>
<tr>
<th>Node</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SZDM.I6ADZM6 03B9C1C0 FEPI Node</td>
<td>Shows the connections used by the node.</td>
</tr>
<tr>
<td>SZDM.I6ADZM6 03B9C1C0 FEPI Node</td>
<td>Shows the pool surrogates chained to the node.</td>
</tr>
<tr>
<td>SZDM.I6ADZM6 03B9C1C0 FEPI Node</td>
<td>Shows the nodes' ACB.</td>
</tr>
<tr>
<td>SZDM.I6ADZM6 03B9C1C0 FEPI Node</td>
<td>Shows the node's RPL.</td>
</tr>
<tr>
<td>SZDM.I6ADZM6 03B9C1C0 FEPI Node</td>
<td>There are no unsolicited binds outstanding for this nodes. If there were, the areas would be shown here.</td>
</tr>
</tbody>
</table>

Targets

**SZ.Target FEPI Targets**

<table>
<thead>
<tr>
<th>Target</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SZDM.I6ADZM6 03B9C1C0 FEPI Target</td>
<td>Shows the applid (network id) of the target.</td>
</tr>
<tr>
<td>SZDM.I6ADZM6 03B9C1C0 FEPI Target</td>
<td>Shows the connections used by the target.</td>
</tr>
<tr>
<td>SZDM.I6ADZM6 03B9C1C0 FEPI Target</td>
<td>There are no connections queuing on the target. If there were, the areas would be shown here.</td>
</tr>
</tbody>
</table>
### Connections

<table>
<thead>
<tr>
<th>FEPI request (these can be nulls if there is not a current task).</th>
</tr>
</thead>
<tbody>
<tr>
<td>combination of the CICS transid, termid, and task number for the task running the conversation.</td>
</tr>
<tr>
<td>The internal id of the conversation is a combination of the CICS transid, termid, and task number for the task running the conversation.</td>
</tr>
<tr>
<td>Shows the pool surrogates chained to the target.</td>
</tr>
</tbody>
</table>

#### FEPI request (these can be nulls if there is not a current task).

| Shows the address (if any) of the connection that is running the conversation, and the internal id of the conversation. |
| Shows the conversation type (active, browse, or inactive). |
| Shows the address of the conversation (if any) currently running over the connection. |
| Shows the screen sizes Current/Default/Alt 024*080/024*080/024*080. |
| Shows the status is DTR, Reset, Free, OutB, ¬CD, Lose, RA Data abs, RA Resp abs. |
| Shows the status is DTR, Reset, Free, OutB, ¬CD, Lose, RA Data abs, RA Resp abs. |
| Shows the more significant VTAM statuses of the connection. |
| Shows the address of the conversation (if any) currently running over the connection. |
| Shows the more significant VTAM statuses of the connection. |

### Conversations

#### FEPI Active Conversations

| Shows the address (of the connection that is running the conversation, and the internal id of the conversation. The internal id of the conversation is a combination of the CICS transid, termid, and task number for the task running the FEPI request (these can be nulls if there is not a current task). |
| Shows the conversation type (active, browse, or inactive). |
| Shows the address (of the connection that is running the conversation, and the internal id of the conversation. The internal id of the conversation is a combination of the CICS transid, termid, and task number for the task running the FEPI request (these can be nulls if there is not a current task). |
| Shows the address (of the connection that is running the conversation, and the internal id of the conversation. The internal id of the conversation is a combination of the CICS transid, termid, and task number for the task running the FEPI request (these can be nulls if there is not a current task). |
| Shows the address (of the connection that is running the conversation, and the internal id of the conversation. The internal id of the conversation is a combination of the CICS transid, termid, and task number for the task running the FEPI request (these can be nulls if there is not a current task). |

---

Chapter 41. Front end programming interface (FEPI) 667
Front end programming interface (FEPI)

DQE
e

FEPI and VTAM

This section outlines how FEPI interacts with VTAM, and discusses VTAM control blocks and exits.

You should refer to the OS/390 eNetwork Communications Server: SNA Programming manual for all information relating to VTAM programming.

VTAM control blocks

FEPI uses standard VTAM programming facilities for its communication. The way in which VTAM control blocks interact with FEPI control blocks is as follows:

ACBs Each FEPI node represents a terminal connected to the partner system. Consequently, each node has an access control block (ACB). This ACB is opened when the node is acquired, and closed when the node is released.

NIBs Each FEPI target contains the applid of the back-end system. This is used to build a node initialization block (NIB), when a connection is acquired by issuing a VTAM REQSESS request. In common with CICS data communication, the “confidential” flag is set off.

RPLs There are two types of request parameter list (RPL) used by FEPI:

- Each FEPI outbound request causes the generation of an RPL. This RPL lasts only for the duration of the FEPI request.
- Each FEPI node has a “Receive-Any” RPL. When an inbound flow occurs, this RPL is attached to the FEPI connection, and turned into a “Receive-Specific” RPL. When the flow has been received, a new “Receive-Any” RPL is generated and attached to the node.
VTAM exits

FEPI communicates with VTAM as asynchronously as possible. Therefore, VTAM exits are extensively used for FEPI communication. The following VTAM exits receive control at specific stages of the communication process:

**DFASY**
Processes the receipt of expedited-data-flow control indicators.

**LOGON**
Processes the receipt of a CINIT in which FEPI is acting as the primary logical unit (PLU).

**LOSTERM**
Processes the loss of a session.

**NSEXIT**
Processes:
- The failure of a process that was responded to positively
- A session outage
- The receipt of network service RUs.

**SCIP**
Processes the receipt of session-control requests.

**TPEND**
Processes the termination of VTAM.

### Modules

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHSZATC</td>
<td>adaptor command tables</td>
</tr>
<tr>
<td>DFHSZATR</td>
<td>adaptor program</td>
</tr>
<tr>
<td>DFHSZBCL</td>
<td>cleanup API requests at error routine</td>
</tr>
<tr>
<td>DFHSZBCS</td>
<td>RM collect statistics</td>
</tr>
<tr>
<td>DFHSZBFT</td>
<td>FREE transaction requests scheduler</td>
</tr>
<tr>
<td>DFHSZBLO</td>
<td>lost session reporter</td>
</tr>
<tr>
<td>DFHSZBRS</td>
<td>RM collect resource ID statistics</td>
</tr>
<tr>
<td>DFHSZBSI</td>
<td>signon exit scheduler</td>
</tr>
<tr>
<td>DFHSZBST</td>
<td>STSN transaction scheduler</td>
</tr>
<tr>
<td>DFHSZBUN</td>
<td>unsolicited data transaction scheduler</td>
</tr>
<tr>
<td>DFHSZBUS</td>
<td>RM unsolicited statistics recording</td>
</tr>
<tr>
<td>DFHSZDUF</td>
<td>dump formatting routine</td>
</tr>
<tr>
<td>DFHSZFRD</td>
<td>formatted 3270 RECEIVE support</td>
</tr>
<tr>
<td>DFHSZFSI</td>
<td>formatted 3270 SEND support</td>
</tr>
<tr>
<td>DFHSZIDX</td>
<td>SLU P queue install/discard exit</td>
</tr>
<tr>
<td>DFHSZPCP</td>
<td>SLU P flow controller</td>
</tr>
<tr>
<td>DFHSZPDX</td>
<td>SLU P drain completion exit</td>
</tr>
<tr>
<td>DFHSZPID</td>
<td>SLU P send data processor</td>
</tr>
<tr>
<td>DFHSZPIX</td>
<td>SLU P send completion exit</td>
</tr>
<tr>
<td>DFHSZPOA</td>
<td>SLU P send response processor</td>
</tr>
<tr>
<td>DFHSZPOD</td>
<td>SLU P receive data processor</td>
</tr>
</tbody>
</table>
## Front end programming interface (FEPI)

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHSZPOR</td>
<td>SLU P response processor</td>
</tr>
<tr>
<td>DFHSZPOX</td>
<td>SLU P receive specific response exit</td>
</tr>
<tr>
<td>DFHSZPOY</td>
<td>SLU P receive specific response processor</td>
</tr>
<tr>
<td>DFHSZPQ5</td>
<td>SLU P REQSESS (request session) issuer</td>
</tr>
<tr>
<td>DFHSZPQX</td>
<td>SLU P REQSESS exit</td>
</tr>
<tr>
<td>DFHSZPSB</td>
<td>SLU P bind processor</td>
</tr>
<tr>
<td>DFHSZPSC</td>
<td>SLU P session controller</td>
</tr>
<tr>
<td>DFHSZPSD</td>
<td>SLU P SDT processor</td>
</tr>
<tr>
<td>DFHSZPSH</td>
<td>SLU P SHUTC processor</td>
</tr>
<tr>
<td>DFHSZPSQ</td>
<td>SLU P quiesce complete (QC) processor</td>
</tr>
<tr>
<td>DFHSZPSR</td>
<td>RESETSR processor CSECT</td>
</tr>
<tr>
<td>DFHSZPSS</td>
<td>SLU P STSN processor</td>
</tr>
<tr>
<td>DFHSZPSX</td>
<td>SLU P OPNSEC completion exit</td>
</tr>
<tr>
<td>DFHSZPTE</td>
<td>SLU P TERMSESS processor</td>
</tr>
<tr>
<td>DFHSZRCA</td>
<td>node control processor</td>
</tr>
<tr>
<td>DFHSZRCT</td>
<td>issue processor</td>
</tr>
<tr>
<td>DFHSZRDC</td>
<td>delete connection processor</td>
</tr>
<tr>
<td>DFHSZRDG</td>
<td>discard node processor</td>
</tr>
<tr>
<td>DFHSZRDN</td>
<td>delete node processor</td>
</tr>
<tr>
<td>DFHSZRDP</td>
<td>dispatcher</td>
</tr>
<tr>
<td>DFHSZRD5</td>
<td>discard property set processor</td>
</tr>
<tr>
<td>DFHSZRDT</td>
<td>discard target processor</td>
</tr>
<tr>
<td>DFHSZREQ</td>
<td>request passticket module</td>
</tr>
<tr>
<td>DFHSZRFC</td>
<td>FREE completion processor</td>
</tr>
<tr>
<td>DFHSZRGR</td>
<td>Dispatcher work queue processor</td>
</tr>
<tr>
<td>DFHSZRIA</td>
<td>allocate processor</td>
</tr>
<tr>
<td>DFHSZRIC</td>
<td>define connection processor</td>
</tr>
<tr>
<td>DFHSZRID</td>
<td>discard processor</td>
</tr>
<tr>
<td>DFHSZRIF</td>
<td>install free processor</td>
</tr>
<tr>
<td>DFHSZRII</td>
<td>install processor</td>
</tr>
<tr>
<td>DFHSZRN</td>
<td>install node processor</td>
</tr>
<tr>
<td>DFHSZRO</td>
<td>ACB open processor</td>
</tr>
<tr>
<td>DFHSZRI</td>
<td>install pool processor</td>
</tr>
<tr>
<td>DFHSZRIQ</td>
<td>inquire processor</td>
</tr>
<tr>
<td>DFHSZRS</td>
<td>install processor</td>
</tr>
<tr>
<td>DFHSZRT</td>
<td>install target processor</td>
</tr>
<tr>
<td>DFHSZRIW</td>
<td>SET processor</td>
</tr>
<tr>
<td>DFHSZRNC</td>
<td>NODE processor</td>
</tr>
<tr>
<td>DFHSZRNO</td>
<td>NOOP processor</td>
</tr>
<tr>
<td>DFHSZRP</td>
<td>timer services</td>
</tr>
<tr>
<td>DFHSZRPW</td>
<td>request preparation</td>
</tr>
<tr>
<td>Module</td>
<td>Function</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------------------------------</td>
</tr>
<tr>
<td>DFHSZRQR</td>
<td>queue for REQSESS processing</td>
</tr>
<tr>
<td>DFHSZRQW</td>
<td>request queue processor</td>
</tr>
<tr>
<td>DFHSZRRD</td>
<td>RECEIVE request processor</td>
</tr>
<tr>
<td>DFHSZRRT</td>
<td>request release processor</td>
</tr>
<tr>
<td>DFHSZRSC</td>
<td>connection processor</td>
</tr>
<tr>
<td>DFHSZRSE</td>
<td>SEND request processor</td>
</tr>
<tr>
<td>DFHSZRST</td>
<td>START request processor</td>
</tr>
<tr>
<td>DFHSZRTM</td>
<td>recovery services</td>
</tr>
<tr>
<td>DFHSZRXD</td>
<td>EXTRACT processor</td>
</tr>
<tr>
<td>DFHSZRZZ</td>
<td>TERMINATE processor</td>
</tr>
<tr>
<td>DFHSZSIP</td>
<td>initialization processor</td>
</tr>
<tr>
<td>DFHSZVBN</td>
<td>copy NIB mask to real NIB</td>
</tr>
<tr>
<td>DFHSZVGF</td>
<td>get queue element FIFO</td>
</tr>
<tr>
<td>DFHSZVQS</td>
<td>REQSESS dispatcher</td>
</tr>
<tr>
<td>DFHSZVRA</td>
<td>VTAM receive_any processor</td>
</tr>
<tr>
<td>DFHSZVRI</td>
<td>VTAM receive_any issuer</td>
</tr>
<tr>
<td>DFHSZVSC</td>
<td>delayed bind processor</td>
</tr>
<tr>
<td>DFHSZVSL</td>
<td>SETLOGON request issuer</td>
</tr>
<tr>
<td>DFHSZVSQ</td>
<td>VTAM feedback interpreter</td>
</tr>
<tr>
<td>DFHSZVSR</td>
<td>VTAM feedback interpreter</td>
</tr>
<tr>
<td>DFHSZVSY</td>
<td>VTAM feedback interpreter</td>
</tr>
<tr>
<td>DFHSZWSL</td>
<td>RPL exit after SETLOGON</td>
</tr>
<tr>
<td>DFHSZXDA</td>
<td>VTAM DFASY exit</td>
</tr>
<tr>
<td>DFHSZXFR</td>
<td>RPL exit to free request block</td>
</tr>
<tr>
<td>DFHSZXLG</td>
<td>VTAM logon exit</td>
</tr>
<tr>
<td>DFHSZXLT</td>
<td>VTAM LOSTERM (lost terminal) exit</td>
</tr>
<tr>
<td>DFHSZXNS</td>
<td>VTAM NSEXIT (network services) exit</td>
</tr>
<tr>
<td>DFHSZXPM</td>
<td>STIMER IRB exit routine</td>
</tr>
<tr>
<td>DFHSZXRA</td>
<td>VTAM RECEIVE_ANY exit</td>
</tr>
<tr>
<td>DFHSZXSC</td>
<td>VTAM SCIP (session control) exit</td>
</tr>
<tr>
<td>DFHSZXTP</td>
<td>VTAM TPEND exit</td>
</tr>
<tr>
<td>DFHSZYLG</td>
<td>RPL exit following logon reject</td>
</tr>
<tr>
<td>DFHSZYQR</td>
<td>post for REQSESS processing</td>
</tr>
<tr>
<td>DFHSZYRI</td>
<td>VTAM RECEIVE_ANY issuer</td>
</tr>
<tr>
<td>DFHSZYSC</td>
<td>VTAM SCIP exit extension</td>
</tr>
<tr>
<td>DFHSZYSR</td>
<td>VTAM feedback interpreter</td>
</tr>
<tr>
<td>DFHSZSY</td>
<td>VTAM feedback interpreter</td>
</tr>
<tr>
<td>DFHSZZAG</td>
<td>get RECEIVE_ANY request block</td>
</tr>
<tr>
<td>DFHSZZFR</td>
<td>free RECEIVE_ANY request block</td>
</tr>
<tr>
<td>DFHSZZNG</td>
<td>get session control request block</td>
</tr>
<tr>
<td>DFHSZZRG</td>
<td>get RPL request block</td>
</tr>
</tbody>
</table>
## Front end programming interface (FEPI)

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHSZ2CP</td>
<td>SLU2 flow controller</td>
</tr>
<tr>
<td>DFHSZ2DX</td>
<td>SLU2 drain completion exit</td>
</tr>
<tr>
<td>DFHSZ2ID</td>
<td>SLU2 send data processor</td>
</tr>
<tr>
<td>DFHSZ2IX</td>
<td>SLU2 send completion exit</td>
</tr>
<tr>
<td>DFHSZ2OA</td>
<td>SLU2 send response processor</td>
</tr>
<tr>
<td>DFHSZ2OD</td>
<td>SLU2 receive data processor</td>
</tr>
<tr>
<td>DFHSZ2OR</td>
<td>SLU2 response processor</td>
</tr>
<tr>
<td>DFHSZ2OX</td>
<td>SLU2 receive specific completion exit</td>
</tr>
<tr>
<td>DFHSZ2OY</td>
<td>SLU2 receive specific action module</td>
</tr>
<tr>
<td>DFHSZ2QS</td>
<td>SLU2 REQSESS issuer</td>
</tr>
<tr>
<td>DFHSZ2QX</td>
<td>SLU2 REQSESS exit</td>
</tr>
<tr>
<td>DFHSZ2SB</td>
<td>SLU2 bind processor</td>
</tr>
<tr>
<td>DFHSZ2SC</td>
<td>SLU2 session controller</td>
</tr>
<tr>
<td>DFHSZ2SD</td>
<td>SLU2 SDT processor</td>
</tr>
<tr>
<td>DFHSZ2SH</td>
<td>SLU2 SHUTC processor</td>
</tr>
<tr>
<td>DFHSZ2SQ</td>
<td>SLU2 QC processor</td>
</tr>
<tr>
<td>DFHSZ2SR</td>
<td>SLU2 RESETSR processor</td>
</tr>
<tr>
<td>DFHSZ2SX</td>
<td>SLU2 OPNSEC processor</td>
</tr>
<tr>
<td>DFHSZ2TE</td>
<td>SLU2 TERMSESS processor</td>
</tr>
</tbody>
</table>
Chapter 42. Function shipping

Function shipping allows a transaction from one CICS system to access a resource owned by another CICS system.

The CICS function shipping facility enables separate CICS systems to be connected so that a transaction in one system is able to retrieve data from, send data to, or initiate a transaction in, another CICS system. The facility is available to application programs that use the command-level interface of CICS.

Design overview

Figure 67 gives an overview of the function shipping component of CICS.

This section provides an overview of the operation of CICS when it is being used to communicate with other connected CICS systems for CICS function shipping.
Application programming functions with CICS function shipping

The functions provided by CICS are extended for CICS function shipping so that an application program can issue the following types of command and have them executed on another system:

- Temporary-storage commands
- Transient data commands
- Interval control commands
- File control commands
- DL/I calls
- Program link commands (DPL).

Application programs can use these extended functions without having to know where the resources are actually located; information about where resources are located is contained in the appropriate tables prepared by the system programmer. Alternatively, provision is made for an application program to name a remote system explicitly for a particular request.

Support for syncpoints, whether explicit (through EXEC CICS SYNCPOINT commands) or implicit (through DL/I TERM calls), allows updates to be made in several systems as part of a single logical unit of work.

Error handling routines may need to be extended to handle additional error codes that may be returned from a remote system. See the CICS Intercommunication Guide for the relevant conditions.

Local and remote names

For a transaction to access a resource (such as a file or transient data destination) in a remote system, it is usually necessary for the local resource table to contain an entry for the remote resource. The name of this entry (that is, the name by which the resource is known in the local system) must be unique within the local system. The entry also contains the identity (SYSIDNT) of the remote system and, optionally, a name by which the resource is known in the remote system. (If this latter value is omitted, it is assumed that the name of the resource in the remote system is the same as the name by which it is known in the local system.)

Mirror transactions

When a transaction issues a command for a function to be executed on a remote system, the local CICS system encodes the request and sends it to the system identified in the appropriate CICS table, or on the command itself. The receipt of this request at the remote system results in the attachment of one of the CICS-supplied mirror transactions, namely, CSMI, CSM1, CSM2, CSM3, and CSM5, or transactions CVMI and CPMI. All these transactions use the mirror program, DFHMIRS. (CVMI services LU6.2 sync level 1 requests, including those from CICS/VM, and CPMI services function shipping from CICS OS/2.)

For distributed program link (DPL) requests shipped from a CICS application region to a CICS resource region, the name of the mirror transaction to be attached may be specified by the user. If you specify your own mirror transaction, you must define the transaction in the resource region and associate it with the CICS-supplied mirror program, DFHMIRS.
The CVMI and CPMI transactions service requests sent as part of an LU6.2 synlevel 1 conversation, unlike the other transactions that service requests sent as part of an LU6.2 synlevel 2 conversation or an MRO or LU6.1 conversation.

A mirror transaction executes the initiating transaction’s request and reflects back to the local system the response code and any control fields and data that are associated with the request. If the execution of the request causes the mirror transaction to abend, this information is also reflected back to the initiating transaction.

If a resource has browse place holders or is recoverable, or the lock has been acquired, the mirror transaction becomes a **long-running mirror** and does not end until the issuing transaction ends the logical unit of work (that is, a SYCNPOINT or RETURN). Any resources the mirror has acquired are freed when the initiating transaction issues the appropriate command to free those resources.

**Initialization of CICS for CICS function shipping**

If CICS has been generated with the appropriate options for intercommunication, the initialization of CICS with the ISC=YES system initialization parameter specified causes the following modules to be loaded:

- DFHISP (intersystem communication program)
- DFHXFP (data transformation program)
- DFHXFX (optimized data transformation program).

The entry point addresses of these modules are contained in the optional features list, which is addressed by CSAOPFILA in the CSA.

The mirror program, DFHMIRS, is not loaded until a request is received from a remote system. (This program can only be loaded if there is an associated PPT entry and PCT entries for mirror transactions CSM1, CSM2, CSM3, and CSM5 or for transactions CVMI and CPMI; sample entries are created by the CSD group DFHISC.)

**Note:** The ISC=YES system initialization parameter causes other modules besides those specified earlier to be loaded; the ones mentioned here are those specifically required for CICS function shipping.

**Communication with a remote system**

For multiregion operation, communication between CICS systems can be implemented:

- Through support in CICS terminal control management modules and by use of a CICS-supplied interregion program (DFHIRP) loaded in the link pack area (LPA) of MVS. DFHIRP is invoked by a type 3 supervisory call (SVC). The SVC moves the data to an intermediate area in key 0 MVS CSA storage, and schedules an SRB to move the data from the intermediate area to the target.
- By MVS cross-memory services (DFHXMP), which you can select as an alternative to the CICS type 3 SVC mechanism. Here, DFHIRP is used only to open and close the interregion links. Cross-memory services do not require intermediate MVS CSA storage areas.
- By the cross-system coupling facility (XCF) of MVS. XCF is required for MRO links between CICS regions in different MVS images of an MVS sysplex. It is selected dynamically by CICS for such links, if available.
Function shipping

For ISC, communication between CICS systems takes place via ACF/VTAM links. CICS and the CICS application programmer are independent of, and unaware of, the type of physical connection used by ACF/VTAM to connect the two systems.

Protocols

Requests and replies exchanged between systems for CICS interval control, CICS transient data, CICS temporary storage, and DL/I functions are shipped using the standard protocol as defined for SNA logical unit type 6.1.

Requests and replies for CICS file control functions are shipped using a private protocol (with function management headers of type 43).

Symmetrical bracket protocol

Logical unit type 6.1 (LU6.1) sessions between two CICS systems require most protocols to be symmetrical; therefore, CICS receives (as well as sends) end bracket.

Shutdown protocol

The LU6.1 shutdown protocol does not use the SHUTDOWN command; it uses the data flow control commands SBI (stop bracket initiation) and BIS (bracket initial stopped). Shutdown is executed as part of session termination (by DFHZCCLS) and ensures that, when a session is terminated normally (as a result of a master terminal release command or a normal CICS shutdown), there are no unfinished syncpoint requests on the session. This means that when the session is initiated, no resynchronization sequence is required.

Sender error recovery protocol (ERP)

CICS support for LU6.1 uses a symmetrical SNA protocol called Sender ERP. In addition, when CICS wishes to send a negative response to a remote system, it sends a special negative response (0846), which indicates that an ERP message is to follow. This ERP message contains the real system and user sense values, together with a text message. The negative response and ERP message are built by DFHZEMW, and are received and processed by DFHZRAC, DFHZRVX, and DFHZNAC.

Resynchronization protocol

CICS support for LU6.1 sessions that use the syncpoint protocol has associated resynchronization logic, which is used during the initiation of a session after a previous session has terminated abnormally. This logic is used to generate messages concerning the outcome of any logical units of work that were in doubt when the previous session failed. The modules involved are DFHZRSY, DFHZSCX, and DFHZNAC.

CICS function shipping environment

This section describes the system entries for function shipping in the terminal control table, and how function shipping requests or replies are transformed between the format suitable for transmission and the internal parameter list format.

System entries in the terminal control table

All remote systems with which a given system is able to communicate are identified and described in terminal control table system entries (TCTSEs). The name of the system entry is the name specified in the SYSIDNT field of the CICS table entry describing a remote resource.
CICS uses the TCTSE as an anchor point to queue requests made by CICS transactions for connection to the remote system.

Figure 68 shows three TCTTEs. If a transaction fails and you get a transaction dump, this figure shows you how to find the relevant TCTTEs from the TCA.

Figure 68. Task’s view of CICS function shipping TCTTEs
Transformation of requests and replies for transmission between systems

Before a request or reply can be transmitted, it must be transformed from its internal, parameter list (EXEC interface) format to a format suitable for transmission; when received after transmission, the request must be transformed back into a parameter list format.

There are four such transformations (numbered 1 through 4), which are performed by DFHXFP, or by DFHXFX if optimized data transformations are possible. The latter only applies to data transformations for function shipping in an MRO environment, excluding those relating to DL/I requests.

Transformation 1
For a request to be sent by the originating system; transforms from parameter list format to transmission format.

Transformation 2
For a request received by the mirror transaction; transforms from transmission format to parameter list format.

Transformation 3
For a reply to be sent by the mirror transaction; transforms from parameter list format to transmission format.

Transformation 4
For a reply received by the originating system; transforms from transmission format to parameter list format.

The parameter list format above refers to the parameter list that is normally passed to DFHEIP (for CICS requests) or to DFHDLI (for DL/I requests).

The transmission formats of these requests and replies (excluding those for syncpoint protocol) are described in the DFHFMHDS DSECT.

Information that DFHXFP and DFHXFX need to retain between transformations 1 and 4 (in the originating system) or between transformations 2 and 3 (in the mirror system) is stored in a transformer storage area called XFRDS; Figure 69 on page 679 shows some of the more important fields in XFRDS.
This section describes the sending and receiving of requests and replies (other than DL/I or syncpoint requests) between two connected systems at the application-layer level; see Figure 70 on page 680. (The function management and data flow control layers, implemented by CICS terminal control, work in the same way, regardless of the type of request being transmitted.)

### CICS function shipping—handling of EXEC CICS commands

This section describes the sending and receiving of requests and replies (other than DL/I or syncpoint requests) between two connected systems at the application-layer level; see Figure 70 on page 680. (The function management and data flow control layers, implemented by CICS terminal control, work in the same way, regardless of the type of request being transmitted.)

---

**Figure 69. Transformer storage area (XFRDS) fields**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XFRSYSNM</td>
<td>Name of remote system</td>
</tr>
<tr>
<td>XFRATCSE</td>
<td>Address of TCTSE for the named remote system</td>
</tr>
<tr>
<td>XFRATCTE</td>
<td>Address of session TCTTE</td>
</tr>
<tr>
<td>XFRATIOA</td>
<td>Address of current TIOA</td>
</tr>
<tr>
<td>XFRAUIB</td>
<td>Address of UIB created as part of DL/I schedule issued in mirror task</td>
</tr>
<tr>
<td>XFRPLIST</td>
<td>Address of EXEC parameter list</td>
</tr>
<tr>
<td>XFRATABN</td>
<td>Address of entry in resource table (for example, FCT for file control</td>
</tr>
<tr>
<td></td>
<td>resources, DCT for transient data)</td>
</tr>
<tr>
<td>XFRFORMN</td>
<td>Data transformation index (1, 2, 3, or 4)</td>
</tr>
<tr>
<td>DRXPCBAL</td>
<td>Address of local PCB address list. This field is set by XFR4 during schedule</td>
</tr>
<tr>
<td></td>
<td>call, and is used during DB calls</td>
</tr>
<tr>
<td>DRXETAD</td>
<td>Address of point in transformer to which retry routine should return</td>
</tr>
</tbody>
</table>
Sending a request to a remote system

A CICS command is handled for an application program by the EXEC interface program, DFHEIP. DFHEIP analyzes the arguments of each statement to determine the requested function and to assign values into the appropriate CICS control blocks; DFHEIP also performs storage control and error checking on behalf of the application programmer.

Figure 70. Overview of CICS function shipping
If the system has been initialized with the ISC=YES system initialization parameter, and if the request is for one of the functions that could be executed on a remote system (see \textit{"Application programming functions with CICS function shipping" on page 674}), DFHEIP invokes a local/remote decision routine, which inspects the appropriate CICS table to determine whether the request is for a local or a remote resource (unless a remote system has specifically been requested). For all requests except file control, this local/remote decision is taken in DFHEIP. For file control requests, the decision is taken in the file control function shipping interface module, DFHFCRF (see \textit{"Chapter 40. File control" on page 533}).

If the resource is local:

- DFHEIP invokes the appropriate EXEC interface processor module to process the request locally.
- DFHEIFC calls the file control file request handler, DFHFCFR, to process the request locally, and finally returns control to DFHEIP.

\textbf{Note:} A SYSID value that names the local system also causes the request to be processed locally.

If the resource is remote, DFHEIP or DFHFCRF:

1. Allocates a transformer storage area (XFRDS) chained off the EXEC interface storage EIS. XFRDS (see \textit{Figure 69 on page 673}) provides a central area in which all information about processing of the request can be accessed.

2. Places the following data in XFRDS:
   - Name of remote system, for subsequent use by DFHISP (in XFRDS field XFRSYSNM)
   - Address of the application’s list of parameters (EXEC parameter list) associated with the command being executed (in XFRDS field XFRPLIST)
   - Address of the table (FCT, if DFHFCRF; DCT, and so on, otherwise) for the requested resource (in XFRDS field XFRATABN).

3. Issues a DFHIS \texttt{TYPE=CONVERSE} macro, which passes control to the CICS function shipping program DFHISP.

DFHISP obtains the address of the TCTSE for the remote system and places it in XFRDS field XFRATCSE. DFHISP obtains the address of the TCTTE that controls the session with the remote system and places it in XFRDS field XFRATCTE. (DFHISP obtains the address by issuing a DFHTC \texttt{TYPE=POINT} macro. If no session is established, there is no TCTTE; in this case DFHISP issues a DFHTC \texttt{TYPE=ALLOCATE} macro to establish the session TCTTE.)

If no session can be allocated because, for example, all sessions are out of service, DFHISP determines whether or not the function request can be queued for shipping at a later time. If it the request can be queued, then XFRATCTE is set to zero.

 Optionally (if a TIOA already exists from an earlier CICS function shipping request from the same application), DFHISP also places the address of the TIOA in XFRDS field XFRATIOA.

DFHISP then invokes DFHXFP, or DFHXFX for optimized transformations, to transform the requested command and parameter list into a form suitable for transmission. This is known as \textbf{transformation 1}, which:

1. Transforms the original \textbf{command} into an appropriate type of request for transmission.
2. Converts the EXEC parameter list into a **data unit** having a standardized character-string format (together with a function control header) suitable for transmission. The data unit is built in the TIOA and contains a copy of each of the parameters that are addressed by the EXEC parameter list. (For economy of transmission, certain types of data are compressed before being placed in the TIOA.)

3. Returns control to DFHISP.

**Note:** If local queuing is in effect, the data unit is built in user storage.

DFHISP then invokes terminal control to transmit the contents of the TIOA to the remote system and waits for the reply from the remote system, if necessary.

If local queuing is in effect, DFHISP issues a DFHIC TYPE=PUT macro specifying transaction CMPX, which sends the data unit at a later time.

**Receiving a request at a remote system**

Terminal control receives the request transmission and attaches one of the mirror transactions.

The mirror program allocates space for XFRDS in its LIFO storage area. As in the requesting system, XFRDS is a central area in which all information about the processing of the received request can be accessed. The mirror program places the following data in XFRDS:

- Address of the session TCTTE (in XFRDS field XFRATCTE)
- Address of the TIOA (in XFRDS field XFRATIOA).

The mirror program also allocates scratch pad storage in the LIFO storage area for use by DFHXFP (or DFHXFX) in building argument lists. The address of this storage is placed in XFRPLIST.

The mirror program then invokes DFHXFP, or DFHXFX for optimized transformations, to transform the received request into a form suitable for execution by DFHEIP. This is known as **transformation 2**, which:

1. Transforms the received request (as coded in the function management header of the data unit) into an appropriate CICS command.

2. Decodes the TIOA and builds (in the first part of the STORAGE area) an EXEC parameter list that basically consists of addresses that point to fields in the TIOA. (Those fields that were compressed for transmission are expanded and placed in the second part of the STORAGE area; for these fields, the EXEC parameter list points to the expanded versions, not the compressed versions in the TIOA.)

**Note:** The NOHANDLE option is specified on each EXEC CICS command that is created; this has the effect of suppressing DFHEIP’s branching to an error routine.

3. Returns control to the mirror program.

The mirror program then invokes DFHEIP (in the same way as for an application program), passing to it (in register 1) the address of the EXEC parameter list just built.

DFHEIP or DFHFCRF determines whether the request is for a remote resource on yet another system or for a local resource. If the resource is remote, DFHEIP or
DFHFCRF allocates a new and separate transfer storage area XFRDS and invokes DFHISP (as described under "Sending a request to a remote system" on page 680).

If the resource is local, the reply is processed for the mirror program in the usual way.

**Sending a reply at a remote system**
The process of sending a reply in response to a request from another system is similar to that for sending a request; see "Sending a request to a remote system" on page 680.

When DFHEIP has successfully completed execution of the command, control is returned to the mirror program with the results of the execution in the EXEC interface block (EIB). The mirror program then invokes DFHXFP, or DFHXFX for optimized transformations, to transform the command response into a suitable form for the transmission of the reply. This is known as transformation 3, which:

1. Checks whether the existing TIOA is long enough to take the reply; if not, DFHXFP (or DFHXFX) frees the existing TIOA and creates a new one.
2. Converts the EXEC parameter list (kept in the scratch pad area STORAGE) into a data unit having the standardized character-string format suitable for transmission. The data unit is built in the TIOA. If the request is received by the mirror program without CD (that is, the requesting system did not expect a reply), the mirror program issues a DFHTC TYPE=READ or TYPE=FREE macro. If an error is detected, the mirror program is forced to abend, so that at least a record of the request failure is written.
3. Returns control to the mirror program.

The mirror program then invokes terminal control to transmit the TIOA. (The mirror program does this by issuing a DFHTC TYPE=(WRITE,WAIT,READ) macro if the mirror program holds any state information that must be held for a further request or until a syncpoint. Otherwise, a DFHTC TYPE=(WRITE,LAST) macro is issued.

**Receiving a reply from a remote system**
Terminal control receives the reply and returns control to the initiating task; in particular, control is passed to DFHISP, which has been waiting for the reply.

DFHISP invokes DFHXFP, or DFHXFX for optimized transformations, (passing to it the address of the XFRDS area) in order to transform the reply into the form expected by the application program. This is known as transformation 4, which:

1. Obtains the addresses of the TIOA and of the original EXEC parameter list from XFRATIOA and XFRPLIST in the XFRDS area.
2. Uses data in the reply to complete the execution of the original command. For example:
   - Sets return codes in the EIB from status bits in the reply
   - Stores other received data (if any) in locations specified in the original EXEC parameter list.
3. Frees the TIOA.
4. Returns control to DFHISP.

DFHISP returns control to DFHEIP (if appropriate through DFHEIFC), which raises any error conditions associated with return codes set in the EIB. DFHEIP then returns control to the application program.
CICS function shipping—handling of DL/I requests

DL/I requests are handled in a similar manner to that for CICS commands; see Figure 71 on page 683.
Sending a DL/I request to a remote system
All DL/I requests are handled by DFHDLI.

DFHDLI determines whether the request is for a remote, or DBCTL database, and routes the request to the appropriate DL/I call processor. If the request is for a remote database, DFHDLI invokes DFHDLIRP, which passes control to DFHISP by issuing a DFHIS TYPE=CONVERSE macro.

DFHISP then:
1. Invokes DFHXFP to transform the request into a form suitable for transmission
2. Invokes terminal control to transmit the request.

Receiving a DL/I request at a remote system
As for a CICS request, the appropriate mirror transaction (in this case, CSM5) is attached.

The mirror program invokes DFHXFP to transform the received request into a form suitable for execution by DFHDLI.

The mirror program then passes the request to DFHDLI in the same way as any other application program would. DFHDLI determines what type of DL/I request is being made and then routes the request to the appropriate DL/I call processor: DFHDLIRP (remote, that is, daisy-chained to yet another system), or DFHDLIDP (DBCTL).

Sending a DL/I reply at a remote system
When DFHDLI has successfully completed the request, control is returned to the mirror program with the results in the user interface block (UIB). The mirror program then:
1. Invokes DFHXFP to transform the results into a form suitable for transmission
2. Invokes terminal control to transmit the reply.

Receiving a DL/I reply from a remote system
On receipt of the reply, terminal control returns control to DFHISP, which has been waiting for the reply; DFHISP then invokes DFHXFP to transform the reply into a form that can be used by DFHDLI. DFHXFP sets the return codes in an intermediate control block, DFHDRX, so that they can ultimately be copied to the UIB or the TCA for the application program. Control is then returned from DFHISP through DFHDLIRP to DFHDLI, and finally back to the application program.

Terminal control support for CICS function shipping
Terminal control support for CICS function shipping falls into the following three main areas:
1. TCTTE allocation functions, ALLOCATE, POINT, and FREE. These functions are used mainly by DFHISP to allow a CICS transaction to own additional TCTTEs. These are session TCTTEs to remote systems; these functions are supported by DFHZISP.
2. Syncpoint functions, SPR, COMMIT, ABORT, and PREPARE. These functions are used by the recovery manager connectors to implement the syncpoint protocol; these functions are supported by DFHZIS1.
3. LU6.1 functions. These functions are used by users of terminal control to support the data flow control protocols used in a LU6.1 session.
The functions described in areas 1 and 2 above are extensions to the DFHTC macro that are intended for internal use by CICS control programs only; they are not documented in the user manuals.

**TCTTE allocation functions**

Terminal control provides the following TCTTE-related functions:

**ALLOCATE function**

This allocates to the requesting transaction a session TCTTE for communication with a remote system. The name of the remote system is passed as a parameter. The address of the allocated TCTTE or a return code is returned to the requester. DFHZISP uses the DFHZCP automatic transaction initiation (ATI) mechanism to allocate the session.

If the allocation request cannot be satisfied immediately, an automatic initiate descriptor (AID) is created, and is chained off the system entry; the AID is used to remember, and subsequently to process, the outstanding allocation request.

Parallel sessions can be allocated explicitly, or implicitly by reference to a remote resource; sessions are automatically initiated at allocation time, if necessary. They can also be initiated by a master terminal ACQUIRE command, or automatically during CICS initialization if CONNECT=AUTO is specified in the TCTTE.

**POINT function**

This causes terminal control to supply the requesting task with the address of a session TCTTE for a named remote system. The TCTTE must have been previously allocated to the requesting task.

**FREE function**

This detaches a TCTTE from the owning task and makes it available for allocation to another transaction. (The FREE function is the opposite of the ALLOCATE function.)

**TERM=YES operand**

This operand enables the issuer of a terminal control macro to select explicitly the TCTTE to which the requested function is to be applied. The address of the TCTTE to be processed is passed as a parameter of the request; the TCTTE must have been previously allocated to the requesting task.

**FREE TCTTE indicator**

This indicator is set as a result of the remote system issuing a (WRITE,LAST) or FREE request to show that the current conversation has finished and that the session should be freed by the current owner of the TCTTE. The receiver of the FREE indicator (usually DFHISP) must issue a FREE request.

**Syncpoint functions**

For ISC, terminal control provides the following syncpoint functions (the equivalent functions for IRC are provided by DFHZIS1):

**SPR (syncpoint request) function**

This request is issued by the recovery manager connector during syncpoint processing, and causes terminal control (DFHZSDR) to send a request that has a definite DR2 response requested. This tells the other side of the session that a syncpoint is required.

**COMMIT function**

This request is issued by the recovery manager connector when syncpoint has been completed. It causes a positive DR2 response to be sent, signaling the successful completion of syncpoint protocol.
Function shipping

**ABORT function**
This request causes either a negative DR2 response or an LUSTATUS command to be sent, indicating that a requested syncpoint operation could not be completed successfully, or that there has been an abnormal end of the current logical unit of work.

**PREPARE function**
This request causes an LUSTATUS command to be sent to the mirror in the remote system and indicates that a syncpoint should be taken.

**VTAM secondary half-session support**
CICS acts as both the primary and the secondary halves of an LUTYPE6 session. To implement secondary half-session support, CICS VTAM terminal control has to do two things:
1. Implement the secondary half of the data flow control and session control protocols that CICS already uses as a primary.
2. Use the secondary API provided by VTAM.

The terminal control functions provided by CICS are independent of primary/secondary considerations. Differences between the primary and secondary VTAM interfaces are contained within the CICS modules that issue the appropriate VTAM request. The secondary support functions appear principally in the DFHZCP modules shown in Table 57.

<table>
<thead>
<tr>
<th>Modules</th>
<th>Function</th>
<th>Secondary function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHZSIM</td>
<td>Request LOGON</td>
<td>Use REQSESS macro</td>
</tr>
<tr>
<td>DFHZOPN</td>
<td>OPNDST</td>
<td>Use OPNSEC macro</td>
</tr>
<tr>
<td>DFHZSCX</td>
<td>SCIP exit</td>
<td>Receive and process BIND, STSN, SDT, CLEAR, and UNBIND commands</td>
</tr>
<tr>
<td>DFHZCLS</td>
<td>CLSDST</td>
<td>Use TERMSESS macro</td>
</tr>
<tr>
<td>DFHZRSY</td>
<td>Resynchronization</td>
<td>Build STSN responses</td>
</tr>
<tr>
<td>DFHZSKR</td>
<td>Respond to</td>
<td>Send responses to BIND, SDT, and STSN commands</td>
</tr>
<tr>
<td>DFHZRAC,</td>
<td>Receive</td>
<td>Receive and process BID commands</td>
</tr>
<tr>
<td>DFHZRVX</td>
<td>Bracket protocol</td>
<td>Implement secondary contention resolution using bracket protocol</td>
</tr>
<tr>
<td>DFHZATI,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DFHZRVC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DFHZNSP</td>
<td>Network services error exit</td>
<td>Handle secondary LOSTERM type of errors</td>
</tr>
</tbody>
</table>

**NOCHECK option function handling**
The transmission of a START NOCHECK command and associated data is handled in a slightly different manner from that for other CICS function shipping commands. Compared with the process described earlier in \[Security manager domain’s generic gates\] on page 1044, the major differences are:
- After DFHISP has allocated the session TCTTE to the requesting task, the transformation program DFHXFP (or DFHFXF) performs transformation 1. In addition, the transformation program detects that a START NOCHECK command is being processed and passes this fact to DFHISP in its return code. Accordingly, DFHISP issues a DFHTC TYPE=WRITE macro, which is deferred until syncpoint, return, or another function-shipped request on that ISC session.
- DFHISP returns to its caller.
Function shipping

- On the receiving system, DFHEIP handles the START NOCHECK command in the usual way and then terminates when the command has been executed; no response is sent back to the first system.

Exits

There are two global user exit points in DFHISP: XISCONA and XISLCLQ. For further information about using these exit points, see the CICS Customization Guide.

Trace

The following point ID is provided for the intersystem program:
- AP 00DF, for which the trace level is IS 1.

The following point IDs are provided for function shipping data transformation:
- AP D9xx, for which the trace level is IS 1.

For more information about the trace points, see the CICS Trace Entries. For more information about using traces in problem determination, see the CICS Problem Determination Guide.
Chapter 43. “Good morning” message program

The CICS good morning program issues a “good morning” message for VTAM logical units.

Design overview

This module is invoked by running the CSGM system transaction.

If a satisfactory OPNDST has occurred (detected in the OPNDST exit, DFHZOPX) and if a “good morning” message has been requested on the TCT TYPE=TERMINAL entry, an NACP request is queued. NACP issues a DFHIC TYPE=INITIATE for this transaction.

This module determines the terminal type, sets up the appropriate control characters, gets a TIOA, and writes the message.

For a 3270 terminal, if the operator has entered data before the message has been received, NACP may be invoked to handle intervention required. In this case the transaction is abended and the write operation terminated.

A default message text is generated by DFHTCXPX and can be overridden by an option on the TCT TYPE=INITIAL statement. The text is stored in the TCT prefix.

Modules

DFHGMM

Exits

The XGMTEXT global user exit point is provided in DFHGMM. For further information about this, see the CICS Customization Guide.

Trace

No trace points are provided for this function.
Chapter 44. IIOP domain (II)

The IIOP domain represents the non-Java portion of the IIOP EJB support encompassing the following:
- Request Receiver
- Request Handler
- Request Processor
- Request Models
- Command Processor

IIOP domain’s specific gates

Table 58 summarizes the IIOP domain’s specific gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and whether or not the functions are available through the exit programming interface (XPI).

Table 58. IIOP domain’s specific gates

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>XPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>IIRR</td>
<td>II 0100</td>
<td>PROCESS_REQUESTS</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>II 0101</td>
<td>FIND_REQUEST_STREAM</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>II 0200</td>
<td>FIND_REQUEST_STREAM</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>II 0201</td>
<td>FIND_REQUEST_STREAM</td>
<td>NO</td>
</tr>
<tr>
<td>IIHR</td>
<td>II 0300</td>
<td>ADD_REPLACE_RQMODEL</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>II 0301</td>
<td>DELETE_RQMODEL</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>II 0400</td>
<td>INQUIRE_RQMODEL</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>II 0401</td>
<td>START_BROWSE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>II 0500</td>
<td>GET_NEXT</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>II 0501</td>
<td>END_BROWSE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>II 0600</td>
<td>MATCH_RQMODEL</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>II 0601</td>
<td>MATCH_RQMODEL</td>
<td>NO</td>
</tr>
<tr>
<td>IIRQ</td>
<td>II 0700</td>
<td>GET_INITIAL_DATA</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>II 0701</td>
<td>RECEIVE_REQUEST</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>II 0800</td>
<td>INVOKE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>II 0801</td>
<td>SEND_REPLY</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>II 0900</td>
<td>RECEIVE_REPLY</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>II 0901</td>
<td>INITIALISE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>II 1000</td>
<td>TERMINATE</td>
<td>NO</td>
</tr>
<tr>
<td>IICP</td>
<td>II 0000</td>
<td>ABSTRACT</td>
<td>NO</td>
</tr>
</tbody>
</table>

IILS gate, ADD_LOGICAL_SERVER function

The ADD_LOGICAL_SERVER function of the IILS gate is used to invoke JAVA code to add a logical server.

Input parameters

- **logical_server**
  - Name of the logical server to be added.
- **shelf**
  - Name of the shelf to which the logical server is to be added.

Output parameters

- **RESPONSE**
  - is the domain’s response to the call. It can have any of these values:
IIO domain (II)

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>SHELF_ACCESS_ERROR</td>
</tr>
</tbody>
</table>

IILS gate, DELETE_LOGICAL_SERVER function

The DELETE_LOGICAL_SERVER function of the IILS gate is used to invoke JAVA code to delete a logical server.

Input parameters

- **logical_server**: Name of the logical server to be deleted.
- **shelf**: Name of the shelf to which the logical server is to be deleted.

Output parameters

- **RESPONSE**: is the domain’s response to the call. It can have any of these values:
  OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

  [REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NOT_FOUND SHELF_ACCESS_ERROR</td>
</tr>
</tbody>
</table>

IILS gate, PUBLISH_LOGICAL_SERVER function

The PUBLISH_LOGICAL_SERVER function of the IILS gate is used to invoke JAVA code to publish a logical server to JNDI.

Input parameters

- **logical_server**: Name of the logical server to be published.

Output parameters

- **RESPONSE**: is the domain’s response to the call. It can have any of these values:
  OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

  [REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NOT_FOUND JNDI_ERROR</td>
</tr>
</tbody>
</table>
IILS gate, RETRACT_LOGICAL_SERVER function

The RETRACT_LOGICAL_SERVER function of the IILS gate is used to invoke JAVA code to retract a logical server from the JNDI.

**Input parameters**

- **logical_server**
  - Name of the logical server to be retracted.

**Output parameters**

- **RESPONSE**
  - is the domain’s response to the call. It can have any of these values: OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

- **[REASON]**
  - is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NOT_FOUND, JNDI_ERROR</td>
</tr>
</tbody>
</table>

IIMM gate, ADD_REPLACE_RQMODEL function

The ADD_REPLACE_RQMODEL function of the IIMM gate is used to install or delete and install a request model.

**Input parameters**

- **rqmodel_name**
  - Name of the request model to be installed.

- **corbaserver**
  - Name of the corbaserver for this request model.

- **model_type**
  - can be any of the following values: EJB | CORBA | GENERIC

- **[module_pattern]**
  - If CORBA - name of the module.

- **[interface_pattern]**
  - If CORBA - name of the interface.

- **[bean_pattern]**
  - If EJB - name of the bean.

- **[interface_type]**
  - If EJB - HOME | REMOTE | BOTH

- **operation_pattern**
  - name of the operation.

- **tranid**
  - Transaction name.

- **catalog**
  - YES | NO. If YES the request model is added to the catalogue.

**Output parameters**

- **duplicate_model_name**
  - Name of a model with the same parameters if DUPLICATE_PATTERN set.
RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>INVALID_NAME DUPLICATE_PATTERN</td>
</tr>
</tbody>
</table>

**IIIM gate, DELETE_RQMODEL function**
The DELETE_RQMODEL function of the IIIM gate is used to delete an installed request model.

**Input parameters**

rqmodel_name
Name of the request model to be deleted.

**Output parameters**

RESPONSE
is the domain’s response to the call it can have any of these values
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NOT_FOUND</td>
</tr>
</tbody>
</table>

**IIIM gate, COMMIT_RQMODELS function**
The COMMIT_RQMODELS function of the IIIM gate is used to commit the request model to the catalogue.

**input parameters**

commit_token
Token for catalogue writes.

**Output parameters**

RESPONSE
is the domain’s response to the call it can have any of these values
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NOT_FOUND</td>
</tr>
</tbody>
</table>
IIRH gate, FIND_REQUEST_STREAM function

The FIND_REQUEST_STREAM function of the IIRH gate is used to examine the incoming GIOP request and to find a new or existing request stream using request models and the directory.

Input parameters

request_block
Address and length of the GIOP Request - the block must contain the whole of the request header. It need not contain the body.

[urm_commarea_block]
Storage used as input to the security User Replaceable Module (URM).

[urmname]
Name of the security URM.

[userid]
userid to be used by the ORB.

[force_create]
YES | NO - YES indicates that IIRH must CREATE a new request stream. NO indicates that normal logic is used to see if a request stream exists and to JOIN it if it does or CREATE a new one if it does not.

Output parameters

request_stream_token
The token, representing the request stream, to be used as input for the SEND_REQUEST.

result
JOINED | CREATED - whether the request stream was joined or created.

RESPONSE
is the domain’s response to the call it can have any of these values OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>URM_DENIED_PERMISSION URM_USERID_NOTAUTH INVALID_OBJECT_KEY REQUEST_ERROR</td>
</tr>
</tbody>
</table>

IIRH gate, PARSE function

The PARSE function of the IIRH gate is used to examine the incoming GIOP request or reply and to return selected information in the output parameters.

Input parameters

request_block
Address and length of the GIOP Request/reply - the block must contain the whole of the request/reply header. It need not contain the body.

Output parameters

[response_expected]
YES | NO - YES is set if the response_expected bit is on in the request header.
IIOP domain (II)

[reply_status]
NO_EXCEPTION | USER_EXCEPTION | SYSTEM_EXCEPTION |
LOCATION_FORWARD This extracts the reply status from a reply header.

codeset_context]
This is a block containing a pointer to and the length of the named context
if it exists within the request or reply. The pointer and length are set to 0 if
the context does not exist.

[sending_context]
This is a block containing a pointer to and the length of the named context
if it exists within the request or reply. The pointer and length are set to 0 if
the context does not exist.

[tracking_context]
This is a block containing a pointer to and the length of the named context
if it exists within the request or reply. The pointer and length are set to 0 if
the context does not exist.

[connection_context]
This is a block containing a pointer to and the length of the named context
if it exists within the request or reply. The pointer and length are set to 0 if
the context does not exist.

[redirection_context]
This is a block containing a pointer to and the length of the named context
if it exists within the request or reply. The pointer and length are set to 0 if
the context does not exist.

[requestId]
is the requestId extracted from the request or reply header.

RESPONSE
is the domain’s response to the call it can have any of these values
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values
are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>REQUEST_ERROR</td>
</tr>
</tbody>
</table>

IIRP gate, GET_INITIAL_DATA function

The GET_INITIAL_DATA function of the IIRP gate is used by the ORB program
DFIIRP to set up an environment to allow it to issue further IIRP requests and to
return the output parameters below.

Input parameters
none

Output parameters
rp_token
token to allow further calls for the same Request Processor
server_name
name of the corba server held by the request stream for the incoming request.

public_id
The public_id that identifies the request stream for the incoming request.

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ERROR_REENTERED ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>REQUEST_STREAM_NOT_CURRENT NO_SERVER_DATA NO_PUBLIC_ID</td>
</tr>
</tbody>
</table>

### IIOP domain (II)

#### IIRP gate, RECEIVE_REQUEST function

The RECEIVED REQUEST function of the IIRP gate is used by the ORB program DFJIIRP to receive a request via a request stream from a Request Receiver. This is for INBOUND requests.

**Input parameters**

- RP_TOKEN
  Token supplied by GET_INITIAL_DATA or INITIALISE representing state storage.

- CONTINUE
  YES | NO - YES is set if RECEIVE_REQUEST is to listen for a further request.

- REQUEST_BUF
  A buffer, into which the received request is to be placed.

- RECEIVE_TYPE
  FULL | OVERFLOW - FULL is set for the first receive_request.

  OVERFLOW is set if the buffer supplied to the first receive_request was too small.

**Output parameters**

- correlation_id
  The correlation id returned by the request stream receive_request.

- bytes_available
  Set if BUFFER_TOO_SMALL is set. It contains the actual size of of the buffer needed which is obtained from the GIOP header received by receive_request.

- RESPONSE
  is the domain’s response to the call. It can have any of these values:
  OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

  [REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:
The INVOKE function of the IIRP gate is used by the ORB program DFJIIRP to send an outbound request and to receive its reply.

**Input parameters**
- **RP_TOKEN**
  Token supplied by GET_INITIAL_DATA or INITIALISE representing state storage.
- **RS_TOKEN**
  Token representing the outbound request stream.
- **CONTINUE**
  YES | NO - YES is set if RECEIVE_REQUEST is to listen for a further request.
- **REQUEST_BUF**
  A block holding the request to be sent.
- **REPLY_BUF**
  A buffer, into which the reply is to be placed.

**Output parameters**
- **bytes_available**
  Set if BUFFER_TOO_SMALL is set. It contains the actual size of the buffer needed for the reply which is obtained from the GIOP reply header received by INVOKE
- **RESPONSE**
  is the domain’s response to the call. It can have any of these values: OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED
  [REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ERROR_REENTERED ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>REQUEST RECEIVED LISTEN FAILED MESSAGE NOT RECEIVABLE RECEIVE_REPLY FAILED SEND_REQUEST FAILED BUFFER TOO SMALL TIMEOUT NOTIFIED GIOP_REQ_HEADER INVALID GIOP_FRAGMENT NOT EXPECTED GIOP_CLOSE_CONN RECEIVED GIOP_FRAGMENT_EXPECTED GIOP_INVALID_VERSION GIOP_INVALID_MESSAGE_TYPE GIOP_MESSAGE_ERROR RCVD REQUEST INVALID REDIRECTION RECEIVED</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID RP_TOKEN</td>
</tr>
</tbody>
</table>

The RECEIVE_REPLY function of the IIRP gate is used by the ORB program DFJIIRP to receive an outbound reply to an outbound request. It is used, following INVOKE, if INVOKE indicated that a further request was ready before the reply was available (loopback) or if the reply buffer supplied by INVOKE was too small.
Input parameters

RP_TOKEN
Token supplied by GET_INITIAL_DATA or INITIALISE representing state storage.

RS_TOKEN
Token representing the outbound request stream.

RECEIVE_TYPE
FULL | OVERFLOW. FULL is set if all of the reply is to be received. OVERFLOW is set if the previous INVOKE or RECEIVE_REPLY ended in BUFFER_TOO_SMALL.

REPLY_BUF
A buffer, into which the reply is to be placed.

Output parameters

bytes_available
Set if BUFFER_TOO_SMALL is set. It contains the actual size of the buffer needed for the reply which is obtained from the GIOP reply header received by INVOKE.

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ERROR_REENTERED ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>REQUEST_RECEIVED LISTEN_FAILED MESSAGE_NOT_RECEIVABLE RECEIVE_REPLY_FAILED SEND_REQUEST_FAILED BUFFER_TOO_SMALL TIMEOUT_NOTIFIED GIOP_REP_HEADER_INVALID GIOP_FRAGMENT_NOT_EXPECTED GIOP_CLOSE_CONN_RECEIVED GIOP_FRAGMENT_EXPECTED GIOP_INVALID_VERSION GIOP_INVALID_MESSAGE_TYPE GIOP_MESSAGE_ERROR_RCVD REQUEST_INVALID_REDIREDIRECTION_RECEIVED</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_RP_TOKEN</td>
</tr>
</tbody>
</table>

IIRP gate, SEND_REPLY function

The SEND_REPLY function of the IIRP gate is used by the ORB program DFJIIRP to send a reply via a request stream to an inbound request.

Input parameters

RP_TOKEN
Token supplied by GET_INITIAL_DATA or INITIALISE representing state storage.

REPLY_BUF
A buffer containing the reply to be sent.

CORRELATION_ID
of the request returned by IIRP RECEIVE_REQUEST.

Output parameters

RESPONSE
is the domain’s response to the call. It can have any of these values:
IIOP domain (II)

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[RREASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>SEND_REPLY_FAILED</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_RP_TOKEN</td>
</tr>
</tbody>
</table>

IIRP gate, INITIALISE function

The INITIALISE function of the IIIP gate is used by the ORB program DFJIIP to set up an environment to allow it to issue further IIIP requests. This is used during COMMAND PROCESSING. For example when DFJIIP is processing an ADD_CORBASERVER command.

Input parameters

rp_token
token to allow further calls for the same Request Processor.

Output parameters

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[RREASON]
is returned when RESPONSE is DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ERROR_REENTERED ABEND</td>
</tr>
</tbody>
</table>

IIRP gate, TERMINATE function

The TERMINATE function of the IIIP gate is used by the ORB program DFJIIP in normal and command processing mode to free any storage obtained by GET_INITIAL_DATA or INITIALISE. If necessary, it will also leave the request stream.

Input parameters

rp_token
token to allow further calls for the same Request Processor. If rp_token is 0 then TERMINATE checks to see if this task has any state storage to be freed.

Output parameters

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[RREASON]
is returned when RESPONSE is DISASTER. Possible values are:
IIRQ gate, INQUIRE_RQMODEL function

The INQUIRE_RQMODEL function of the IIRQ gate is used to inquire on a particular model, returning the output parameters below.

**Input parameters**

rqmodel_name
Name of the request model for which information is needed.

**Output parameters**

[corbaserver]
Name of the corbaserver for this request model.

[model_type]
EJB | CORBA | GENERIC

[module_pattern]
If CORBA - name of the module.

[interface_pattern]
If CORBA - name of the interface.

[bean_pattern]
If EJB - name of the bean.

[interface_type]
If EJB - HOME | REMOTE | BOTH

[operation_pattern]
name of the operation.

[tranid]
Transaction name.

**RESPONSE**

is the domain’s response to the call it can have any of these values

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

**[REASON]**

is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>NOT_FOUND</td>
</tr>
</tbody>
</table>

IIRQ gate, START_BROWSE function

The START_BROWSE function of the IIMM gate is used to return a token to allow all the request models to be browsed.

**Input parameters**

none
IIOPI domain (II)

**Output parameters**

**browse_token**

- token to be used by get_next and end_browse.

**RESPONSE**

- is the domain’s response to the call. It can have any of these values:
  - OK
  - EXCEPTION
  - DISASTER
  - INVALID
  - KERNERROR
  - PURGED

**[REASON]**

- is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NONE</td>
</tr>
</tbody>
</table>

**IIOP gate, GET_NEXT function**

The GET_NEXT function of the IIMM gate is used to pass back the output parameters for the next request model.

**Input parameters**

**browse_token**

- token created by start_browse representing the next request model

**Output parameters**

**rqmodel_name**

- Name of the request model to be installed.

**[corbaserver]**

- Name of the corbaserver for this request model.

**[model_type]**

- EJB | CORBA | GENERIC

**[module_pattern]**

- If CORBA - name of the module.

**[interface_pattern]**

- If CORBA - name of the interface.

**[bean_pattern]**

- If EJB - name of the bean.

**[interface_type]**

- If EJB - HOME | REMOTE | BOTH

**[operation_pattern]**

- name of the operation.

**[tranid]**

- Transaction name.

**RESPONSE**

- is the domain’s response to the call. It can have any of these values:
  - OK
  - EXCEPTION
  - DISASTER
  - INVALID
  - KERNERROR
  - PURGED

**[REASON]**

- is returned when RESPONSE is EXCEPTION. Possible values are:
IIRQ gate, **END_BROWSE function**

The END_BROWSE function of the IIMM gate is used to end the browse session.

**Input parameters**

- **browse_token**
  
  Token created by start_browse.

**Output parameters**

- **RESPONSE**
  
  is the domain’s response to the call. It can have any of these values:
  
  OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

- **[REASON]**
  
  is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>none</td>
</tr>
</tbody>
</table>

IIRQ gate, **MATCH_RQMODEL function**

The MATCH_RQMODEL function of the IIRQ gate is used to find the most specific request model that matches the input parameters.

**Input parameters**

- **corbaserver**
  
  Name of the corbaserver for this request model.

date

- **[module_name_block]**
  
  If CORBA - name of the module.

- **[interface_name_block]**
  
  If CORBA - name of the interface.

- **[bean_name_block]**
  
  If EJB - name of the bean.

- **[interface_type]**
  
  If EJB - HOME | REMOTE | BOTH

- **operation_pattern**
  
  name of the operation.

**Output parameters**

- **[tranid]**
  
  Transaction name from the matching model. If no match is found the default of CIRP is returned.

**RESPONSE**

is the domain’s response to the call it can have any of these values

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED
IIOp domain (II)

[REASON]

is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>none</td>
</tr>
</tbody>
</table>

IIRR gate, PROCESS_REQUESTS function

The PROCESS_REQUESTS function of the IIRR gate is used to receive a GIOP request from a socket, find a request stream, send the request over the request stream, optionally receive a reply and send the reply to the socket. This process continues until the socket is closed or no further data is available.

Input parameters

none

Output parameters

RESPONSE

is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]

is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ERROR_REENTERED ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>RESCHEDULE NO_PERMISSION GIOP_FRAGMENT_NOT_EXPECTED GIOP_CLOSE_CONN_RECEIVED GIOP_FRAGMENTEXPECTED GIOP_INVALID_HEADER GIOP_INVALID_VERSION GIOP_INVALID_MESSAGETYPE GIOP_MESSAGE_ERROR_RCVD GIOP_REPLY_RECEIVED IIRH_FIND_EXCEPTION SOCK_RECEIVE_TIMEOUT</td>
</tr>
</tbody>
</table>

IIOP domain’s generic gates

Table 49 on page 442 summarizes the II domain’s generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 59. II domain’s generic gates

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>IIDM</td>
<td>II 0000</td>
<td>INITIALISE_DOMAIN</td>
<td>DMDM</td>
</tr>
<tr>
<td></td>
<td>II 0001</td>
<td>QUIESCE_DOMAIN</td>
<td></td>
</tr>
<tr>
<td></td>
<td>II 0001</td>
<td>TERMINATE_DOMAIN</td>
<td></td>
</tr>
<tr>
<td>IIST</td>
<td>II 0600</td>
<td>COLLECT_STATISTICS</td>
<td>STST</td>
</tr>
<tr>
<td></td>
<td>II 0601</td>
<td>COLLECT_RESOURCE_STATS</td>
<td></td>
</tr>
<tr>
<td>IIXM</td>
<td>AP 09E0</td>
<td>INIT_XM_CLIENT</td>
<td>XMAC</td>
</tr>
<tr>
<td></td>
<td>AO 09E1</td>
<td>BIND_XM_CLIENT</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TRANSACTION_HANG ABEND_TERMINATE</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>RELEASE_XM_CLIENT</td>
<td></td>
</tr>
</tbody>
</table>

For descriptions of these functions and their input and output parameters, refer to the §§ dealing with the corresponding generic formats.
Functions and parameters

INIT_XM_CLIENT ‘saves’ a pointer obtained by the IIRR socket notify gate for connection data and makes it available to the task.

The Domain Manager gates perform normal internal state initialisation and termination functions.

Modules

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHIICP</td>
<td>II domain command processor DFHIICP provides a common mechanism for the following OT and EJ requests to call JAVA ORB code.</td>
</tr>
<tr>
<td></td>
<td>RESYNC_COORDINATOR</td>
</tr>
<tr>
<td></td>
<td>RESYNC_SUBORDINATE</td>
</tr>
<tr>
<td></td>
<td>PUBLISH_CORBASERVER</td>
</tr>
<tr>
<td></td>
<td>RETRACT_CORBASERVER</td>
</tr>
<tr>
<td></td>
<td>PRE_INSTALL_DJAR</td>
</tr>
<tr>
<td></td>
<td>INSTALL_DJAR</td>
</tr>
<tr>
<td></td>
<td>DISCARD_DJAR</td>
</tr>
<tr>
<td></td>
<td>PUBLISH_DJAR</td>
</tr>
<tr>
<td></td>
<td>RETRACT_DJAR</td>
</tr>
<tr>
<td>DFHIIDM</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>PRE_INITIALIZE</td>
</tr>
<tr>
<td></td>
<td>INITIALIZE_DOMAIN</td>
</tr>
<tr>
<td></td>
<td>QUIESC_DOMAIN</td>
</tr>
<tr>
<td></td>
<td>TERMINATE_DOMAIN</td>
</tr>
<tr>
<td>DFHIIDUF</td>
<td>II domain offline dump formatting routine</td>
</tr>
<tr>
<td>DFHIIMM</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>ADD_REPLACE_RQMODEL</td>
</tr>
<tr>
<td></td>
<td>DELETE_RQMODEL</td>
</tr>
<tr>
<td></td>
<td>COMMIT_RQMODELS</td>
</tr>
<tr>
<td>DFHIILS</td>
<td>Handles the following requests via DFHIICP:</td>
</tr>
<tr>
<td></td>
<td>ADD_LOGICAL_SERVER</td>
</tr>
<tr>
<td></td>
<td>DELETE_LOGICAL_SERVER</td>
</tr>
<tr>
<td></td>
<td>PUBLISH_LOGICAL_SERVER</td>
</tr>
<tr>
<td></td>
<td>RETRACT_LOGICAL_SERVER</td>
</tr>
<tr>
<td>DFHIIRH</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>FIND_REQUEST_STREAM</td>
</tr>
<tr>
<td></td>
<td>PARSE</td>
</tr>
</tbody>
</table>
### IIOP domain (II)

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHIIRP</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>GET_INITIAL_DATA</td>
</tr>
<tr>
<td></td>
<td>RECEIVE_REQUEST</td>
</tr>
<tr>
<td></td>
<td>INVOKE</td>
</tr>
<tr>
<td></td>
<td>SEND_REPLY</td>
</tr>
<tr>
<td></td>
<td>RECEIVE_REPLY</td>
</tr>
<tr>
<td></td>
<td>INITIALISE</td>
</tr>
<tr>
<td></td>
<td>TERMINATE</td>
</tr>
<tr>
<td>DFHIIRQ</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>INQUIRE_RQMODEL</td>
</tr>
<tr>
<td></td>
<td>START_BROWSE</td>
</tr>
<tr>
<td></td>
<td>GET_NEXT</td>
</tr>
<tr>
<td></td>
<td>END_BROWSE</td>
</tr>
<tr>
<td></td>
<td>MATCH_RQMODEL</td>
</tr>
<tr>
<td>DFHIIRR</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>PROCESS_REQUESTS</td>
</tr>
<tr>
<td>DFHIIST</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>COLLECT_STATISTICS</td>
</tr>
<tr>
<td></td>
<td>COLLECT_RESOURCE_STATS</td>
</tr>
<tr>
<td>DFHIITRI</td>
<td>Interprets II domain trace entries</td>
</tr>
<tr>
<td>DFHIIXM</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>INIT_XM_CLIENT</td>
</tr>
<tr>
<td></td>
<td>BIND_XM_CLIENT</td>
</tr>
<tr>
<td></td>
<td>TRANSACTION_HANG</td>
</tr>
<tr>
<td></td>
<td>ABEND_TERMINATE</td>
</tr>
<tr>
<td></td>
<td>RELEASE_XM_CLIENT</td>
</tr>
</tbody>
</table>

### Exits

There is one user replaceable module DFHXOPUS, which is called by DFHIIRR during Request Receiver processing.

### Trace

The point IDs for the storage manager domain are of the form IIxxxx; the corresponding trace levels are II 1, II 2 and Exc. Trace points III000-1FFF are II JRAS trace points.

For more information about the trace points, see the CICS User’s Handbook. For more information about using traces in problem determination, see the CICS Problem Determination Guide.
Chapter 45. Interregion communication (IRC)

CICS multiregion operation (MRO) enables CICS regions that are running in the same MVS image, or in the same MVS sysplex, to communicate with each other. MRO does not support communication between a CICS system and a non-CICS system such as IMS.2

ACF/VTAM and SNA networking facilities are not required for MRO. The support within CICS that enables region-to-region communication is called **interregion communication** (IRC). IRC can be implemented in three ways:

- Through support in CICS terminal control management modules and by use of a CICS-supplied interregion program, DFHIRP, loaded in the MVS link pack area. DFHIRP is invoked by a type 3 supervisory call (SVC).
- By MVS cross-memory services, which you can select as an alternative to the CICS type 3 SVC mechanism. Here, DFHIRP is used only to open and close the interregion links.
- By the cross-system coupling facility (XCF) of MVS. XCF is required for MRO links between CICS regions in different MVS images of an MVS sysplex. It is selected dynamically by CICS for such links, if available.

This section describes the communication part of MRO. “Chapter 58. Multiregion operation (MRO)” on page 847 gives a brief description of multiregion operation.

**Design overview**

For information about the design and implementation of interregion communication facilities, and about the benefits of cross-system MRO, see the CICS Intercommunication Guide.

**Control blocks**

IRC uses two levels of control blocks:

1. A CICS/MRO terminal control layer
2. An interregion SVC layer interfaced by the DFHIR macro.

**Terminal control layer**

The CICS/MRO terminal control layer is shown in Figure 72 on page 710.

This layer uses the cross-region block (CRB). This is a global (that is, one per CICS system) block that is created in the CICS dynamic storage area above the 16MB line (the ECDSA) when IRC is initialized, and provides information to communicate with the IRC SVC. See Figure 73 on page 712.

---

2. The external CICS interface (EXCI) uses a specialized form of MRO link to support: communication between MVS batch programs and CICS; DCE remote procedure calls to CICS programs.
Interregion communication (IRC)

CSA

\[ X'128' \]
- CSATCTBA
  Address of TCT prefix

TCTFX

\[ X'3C' \]
- TCTVSEBA
  Address of local system entry

TCTSE (local)

\[ X'90' \]
- TCSENEXT
  Address of first remote system entry

TCTSE (remote)

\[ X'00' \]
- TCTETI
  Connection name of remote system B

\[ X'08' \]
- TCSEDAID

\[ X'0C' \]
- TCSESUSF
  Address of head of AID chain

\[ X'28' \]
- TCSEVC1
  Address of first primary session TCTTE

\[ X'2C' \]
- TCSEVC2
  Address of first secondary session TCTTE

\[ X'50' \]
- TCSESTAS
  Statistics area

Secondary TCTTE system B

Primary TCTTE for system B:
- a VTAM logical unit type 6
- or IRC terminal entry for session with remote system

\[ X'EC' \]
- TCTESLNK
  ISC system ownership chain

AID

\[ X'10' \]
- AIDCHNF
  Address of next AID

\[ X'10' \]
- AIDCHNF
  Address of dummy AID

\[ X'24' \]
- AIDTCAA
  TCA

Secondary TCTTE system B

Primary TCTTE system C
Notes:

1. The first TCTTE on the chain is not necessarily the TCTTE for the task’s primary terminal.
2. A task has allocated MRO sessions to other systems.
3. TCTTEs are described more fully in "Chapter 93, Terminal control" on page 1195.

Figure 72. CICS/MRO terminal control layer of control blocks (Part 2 of 2)
Interregion communication (IRC)

4. Primary TCTTEs relate to Receive sessions, and secondary TCTTEs relate to Send sessions.

5. TCSEVC1 is the label on the address of the TCTTE of the first primary session. TCSEVC2 is that of the first secondary session.

6. The primary and secondary sessions each have sets of TCTTEs. These are found by using the DFHTC CTYPE=LOCATE macro.

7. A TCTTE is allocated for a surrogate session in transaction routing.

DFHIR layer

The interregion SVC layer interfaced by the DFHIR macro is shown in Figure 74 on page 713.
This layer uses the following control blocks, which, unless otherwise stated, reside in subpool 241 in MVS storage:

- Global (that is, one per MVS system) housekeeping (used by DFHIRP)

  **Subsystem control table extension (SCTE)**
  The SCTE is dynamically created, and contains information about the number of regions logged on to DFHIRP. It is used to locate the LACB. See also [Figure 87 on page 1122](#), which shows the subsystem interface control blocks, including a pointer to the SCTE in the CICS subsystem anchor block (SAB).

- Logon address control block (LACB)
  The LACB contains entries to identify the regions that have logged on, and contains the address of the region’s logon control block (LCB).

- Local housekeeping (used by DFHIRP)

  **Logon control block (LCB)**
  The LCB is created for each successful log on.

  **Logon control block entry (LCBE)**
  The LCBE contains the basic control information for each IRC system with which this system communicates. It addresses the connection control blocks (CCBs).

  **Subsystem user definition block (SUDB)**
  A SUDB provides access to IRC control blocks. There is one SUDB for each TCB that is currently logged on (so each SUDB may have multiple LCBs associated with it). The SUDB contains TCB-related data and working storage.

---

*Figure 74. Interregion SVC layer of control blocks interfaced by the DFHIR macro*
Connection control block (CCB)
A CCB is created for each IRC send-receive session, and contains information controlling the connection to the other region. When the connection is in use, it addresses the CSB.

Connection status block (CSB)
The CSB provides status information about the connection between two regions.

MVS transfer buffers (MVS SRB mode)
The MVS transfer buffers are used to transfer IRC data between regions, and reside in subpool 231 in MVS storage.

Terminal control layer and DFHIR layer
Figure 75 shows the control blocks that are accessed by both the terminal control layer and the DFHIR layer. Figure 76 on page 715 shows the location of these control blocks in MVS virtual storage.

The following blocks are used by both the terminal control layer and the DFHIR layer. These blocks are allocated at logon time within a single MVS GETMAIN, and, unless otherwise stated, reside in subpool 251 of MVS storage.

Subsystem logon control block (SLCB)
The SLCB is used by the IRC SVC and region and contains the master ECB, posted when the region has IRC activity. It is pointed to by the CRB and LCB.

Subsystem connection address control block (SCACB)
The SCACB contains entries allowing the addressing of SCCBs from the SLCB.

Subsystem connection control block (SCCB)
The SCCB is created for each IRC send-receive session, and is allocated at logon. It contains the ECB, posted when input for the session is available.
Note: There is a one-to-one relationship between TCTTEs and SCCBs when they are in use.

MRO ECB summary

The following is a summary of the MRO event control blocks (ECBs):

<table>
<thead>
<tr>
<th>Name</th>
<th>Location</th>
<th>Who waits</th>
<th>Who posts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dependent ECB</td>
<td>SCCB</td>
<td>Application (TC WAIT)</td>
<td>DFHIRP</td>
</tr>
<tr>
<td>LOGON ECB</td>
<td>SLCB</td>
<td>CICS (KCP, Op sys WAIT list)</td>
<td>DFHIRP</td>
</tr>
<tr>
<td>Link ECB</td>
<td>LCB</td>
<td>DFHIRP (Op sys WAIT)</td>
<td>DFHIRP</td>
</tr>
<tr>
<td>Work queue ECB</td>
<td>QUEUE</td>
<td>CSNC transaction</td>
<td>DFHIRP, DFHZIS2, DFHZLOC</td>
</tr>
</tbody>
</table>

See the [CICS Data Areas](CICS Data Areas) manual for a detailed description of the CICS control blocks.

Modules

Figure 77 on page 716 gives an overview of the modules involved with interregion communication.
Interregion communication (IRC)

The modules for IRC are of two types:
1. The interregion programs: DFHIRP and DFHXMP
2. CICS address space modules: DFHCRC, DFHCRNP, DFHCRR, DFHCRSP, DFHZCP, and DFHZCX.

Interregion programs

DFHIRP (interregion communication (SVC) program)
The interregion communication program (DFHIRP) is used to pass data from one region to another in the same processing unit. The programs running in the regions usually are CICS programs, but DFHIRP does not assume that to be the case.

Each user of this program must first issue a LOGON request specifying an 8-character name. This user identifier is added to a table maintained in key 0 storage.

After the user has logged on, CONNECT requests may be issued to establish data paths to other users who have also logged on. The users must cooperate in this process by specifying, when they log on, to whom and from whom they are to be connected and by how many data paths.

After a connection has been established, either end of the connection may issue a SWITCH request to send data to the other end of the connection. The receiver of the data must provide a buffer into which the data is to be written. If the buffer is too small, the receiver is notified of the actual data length and, possibly having obtained a larger buffer, may issue a PULL request to retrieve as much data as is required. After the first data has been sent, the link must be used by each end alternately.

A connection may be broken by either end by issuing a DISCONNECT request. When all links have been disconnected, a user may log off.
When MVS cross-memory services are requested (ACCESSMETHOD(XM) in the RDO CONNECTION definition), DFHXMP is used (DFHIRP performs initialization and termination for DFHXMP); otherwise, communication is performed by DFHIRP running as an SVC. In this case, it is invoked by an SVC call to a startup program (DFHCSVC), which calls the required DFHIRP routine.

**DFHXMP (MVS cross-memory program)**

When the MVS cross-memory services are used for interregion communication, the SWITCH and PULL functions are performed by DFHXMP, which is entered by issuing a *program call* (PC) instruction instead of an SVC. DFHXMP does not need a commonly addressable buffer or service request blocks (SRBs) to effect data transfer between address spaces.

Code in DFHIRP performs the cross-memory initialization and termination functions for DFHXMP as follows:

**LOGON:** Acquire and initialize the cross-memory resources (authorization index (AX), linkage index (LX), and entry table (ET)), unless this has already been done by a previous logon in this address space.

**CONNECT:** Update the authority tables (ATs) of both address spaces to allow each one to establish addressability to the other, unless this was done when a previous connection was established between them.

**DISCONNECT:** If the last cross-memory connection between a pair of address spaces is being removed, update the caller’s AT so that the other system is no longer permitted to access the caller’s address space.

**LOGOFF:** Free the cross-memory resources acquired by logon if they are no longer required by the caller’s address space.

**CICS address space modules**

The CICS address space modules control the handling of requests between this address space and other address spaces. They include several MRO management modules such as DFHCRSP (see [DFHCRSP (CICS IRC startup module)](page) and DFHCRNP (see [DFHCRNP (connection manager—CSNC transaction)](page)), and several terminal-control modules (see “[DFHZCX (CICS terminal control routines)](page)”).

These modules provide the CICS address space with a DFHTC-level interface to interregion communication (in the same way as DFHZCP provides a DFHTC-level interface to VTAM). This enables other CICS modules (such as DFHISP) to allocate and execute input/output operations on IRC sessions. The IRC sessions are used for all forms of IRC communication, and the macro-level services available for IRC are broadly the same. Thus DFHISP works for both IRC and intersystem communication (ISC) function shipping.

The functions of each module are as follows:

**DFHCRSP (CICS IRC startup module)**

Execution of this module makes interregion communication possible between this address space and other address spaces. DFHCRSP, which can be invoked either at system initialization or by the master terminal, allocates the cross-region block (CRB), issues a LOGON request to the SVC routine, and attaches the CSNC transaction (connection manager program, DFHCRNP).
Interregion communication (IRC)

**DFHCRNP (connection manager—CSNC transaction)**
Interregion communication is controlled by the interregion control program, DFHCRNP, which runs as transaction CSNC. This is attached when CICS first logs on to the interregion program, and it remains attached until interregion communication is closed.

The main purpose of CSNC is to perform housekeeping and control on IRC sessions, and to simulate the access method. Its functions include the following:

1. Establish connections to other address spaces (by issuing CONNECT requests)
2. Detect unsolicited input data on connections and attach requested tasks to process such data
3. Disconnect unallocated (between-bracket) sessions during QUIESCE
4. Issue DFHKC AVAIL for any secondary sessions which have become available for reallocation, and are in demand
5. Issue PC RETURN when QUIESCE is complete.

CSNC is attached by DFHCRSP (IRC startup), and waits when it is not processing work. It is resumed by the dispatcher when the MRO work queue ECB has been posted, or the delay interval (if set) has expired and there is delayed work to be retried.

Whenever CSNC is posted, it checks first whether it has been invoked because quiescing of the interregion facility is complete.

- **If CSNC has no been resumed to complete interregion quiesce processing, it checks each of the following:**
  1. If the “delay-queue” is not empty, CSNC attempts to process any work it finds there. (An element is added to the queue whenever a transaction cannot be attached by CSNC. The system could, for example, have been at maximum tasks or short on storage when the previous attempt was made. It is also possible that a remote system tried to start a new conversation before the local system had freed the required session from an earlier conversation.)
  2. If a new conversation has been received:
     - If this is the first conversation on a new connection, and the connecting region is not a batch region, session recovery is performed. This means that if the name of the secondary connecting matches the name of the secondary connected in the previous session, the old session is bound once again.
     - If there is no match, or if a batch region is connecting, the first available session is allocated.
     - CSNC attempts to attach the required transaction, identified in the attach header included in the data stream. It is possible for a request to arrive for this session before the session has been freed from the transaction that last used it. In such a case, the transaction to be attached is added to the delay-queue.
     - The input data stream is built into a TIOA for the session.
  3. If this region is a secondary, and there is no task associated with the connection, and the connection is in quiesce, CSNC disconnects the session.
  4. If this region is a primary, and it has received a “disconnect” request from the connected secondary, CSNC disconnects the session if:
     - There is no associated TCTTE
     - There is no task associated with the link.
- **If CSNC has been resumed to complete interregion quiesce processing, it:**
1. Sends message DFHIR3762 to the CSMT log.
2. Resumes any suspended mirror tasks with a facility address of zero, so they can detach themselves.
3. Disable immediate and delay queues. Any remaining work on those queues (for example, old retry work which has not been serviced yet) is automatically discarded.
4. Logs off from the interregion SVC.
5. Detaches, using a DFHLFM TYPE=RETURN request.

**DFHCRR (CICS session recovery module)**

Whenever a new connection is established (via a successful CONNECT request), DFHCRNP links to DFHCRR at the secondary end of the connection (that is, at the source of the connection). DFHCRNP sends a data stream down to the other end of the connection (the primary end) which causes DFHCRNP to link to DFHCRR at the primary end. The two DFHCRRs exchange information in order to determine whether either end of the connection was in doubt when the previous use of the connection was terminated, and, if so, whether the two ends were in sync or out of sync. In the case of an in-doubt connection, the sequence numbers are compared, diagnostics are issued, and the session is freed.

**DFHCRC (interregion abnormal exit module)**

This module contains the ESTAE exit routine corresponding to the ESTAE macro issued by DFHKESIP. It is invoked if the ESTAE exit, DFHKESTX, decides to continue the abend, or if an X22 abend (which can’t be handled by DFHKESTX) occurs.

The purpose of the exit is to free links with other subsystems to which connection has been made by the interregion SVC, and to free links with the SVC itself. This is done by issuing to the SVC a CLEAR request (to break links with other subsystems).

**DFHZCX (CICS terminal control routines)**

DFHZCX is a load module consisting of a set of object modules, including DFHZIS1 (ISC or IRC syncpoint) and DFHZIS2 (IRC internal functions).

DFHZIS2 provides the following routines:

- **I/O request routine (IORENT)**
  Provides a WRITE/WAIT/READ interface to interregion connections.

- **GETDATA routine (GDAENT)**
  Retrieves input data from an IRC connection and puts it into a TIOA.

- **RECEIVE routine (RECENT)**
  Receives unsolicited data (begin-bracket in SNA terms) and checks validity.

- **DISCONNECT routine (DSCENT)**
  Cleans up this end of a connection, and issues DISCONNECT request to DFHIRP.

- **OPRENT routine (OPRENT)**
  Issues an INSRV request to DFHIRP, in order to allow future connections between this subsystem and a specified subsystem.

- **RECABRT routine (RCAENT)**
  Is invoked when an ABORT FMH (FMH07) is received (indicating that the connected transaction has abended). The routine issues a message describing the failure.
**Interregion communication (IRC)**

**STOP routine (STPENT)**
Is invoked when communication with other address spaces is to be terminated. The routine issues a QUIESCE request to DFHIRP.

**LOGOFF routine (LGFENT)**
Is invoked when quiesce is complete (and during system termination and abend processing). The routine issues a LOGOFF request to the SVC routine.

DFHZIS1 also contains routines representing terminal control services which are supported by IRC (in common with VTAM). These routines include PREPARE, SPR, COMMIT, and ABORT.

**DFHZCP (CICS terminal management program)**
DFHZCP is a load module consisting of a set of object modules, including DFHZARQ (application request handler), DFHZISP (intersystem program allocation routines), and DFHZSUP (startup task).

DFHZARQ is used (in common with all other telecommunication access methods) to handle WRITE/WAIT/READ-level requests against IRC connections (sessions). Routine ZARQIRC in DFHZARQ specifically handles IRC requests by performing SNA request header processing and invoking IORENT (see DFHZCX) in order to perform the I/O on the session.

DFHZISP includes routines such as ALLOCATE and FREE.

---

**Exits**

No global user exit points are provided for this function.

---

**Trace**

The following point IDs are provided for this function:
- AP DDxx, for which the trace levels are IS 1 and IS 2.

For more information about the trace points, see the [CICS Trace Entries](#). For more information about using traces in problem determination, see the [CICS Problem Determination Guide](#).
Chapter 46. Intersystem communication (ISC)

CICS intersystem communication (ISC) allows the following:
- CICS-to-CICS communication
- CICS-to-IMS communication
- CICS-to-LUTYPE6.2 terminal or application communication.

These can be execute simultaneously within the same or a different CEC. ISC can use VTAM LU6.1 or LU6.2 (LU6.2 is preferred for CICS operation). For information about these methods of communication, see the CICS Intercommunication Guide.

The facilities provided by ISC include:
- Transaction routing
- Distributed transaction processing
- Function shipping
- Asynchronous processing
- Distributed program link
- SAA Communications interface.

For information about the design and operation of intersystem communication, see "Chapter 107. VTAM LU6.2" on page 1373. For descriptions of the facilities provided by ISC, see "Chapter 102. Transaction routing" on page 1311, "Chapter 25. Distributed transaction processing" on page 331, "Chapter 42. Function shipping" on page 673, and "Chapter 73. SAA Communications and Resource Recovery Interfaces" on page 1001.
Chapter 47. Interval control

Interval control provides various optional task-related functions based on specified intervals of time, or specified time of day.

Design overview

The following services are performed by interval control in response to a specific request from either an application program or another CICS function:

Time of day

The EXEC CICS ASKTIME command retrieves the current time-of-day in either binary or packed decimal format.

Time-dependent task synchronization

Time-dependent task synchronization provides the user with three optional services:

1. The EXEC CICS DELAY command allows a task to temporarily suspend itself for a specified period of time. When the time has elapsed, the task resumes execution.

2. The EXEC CICS POST command allows a task to be notified when the specified interval of time has elapsed or the specified time of day occurs. The task proceeds to execute while the time interval is elapsing.

3. The EXEC CICS CANCEL command allows a task to terminate its own or another task’s request for a DELAY, POST or START service.

Automatic time-ordered transaction initiation

Automatic time-ordered transaction initiation provides for the automatic initiation of a transaction at a specified time of day (or after a specified interval of time has elapsed) and for the sending of data that is to be accessed by the transaction. The user can also cancel a pending request for automatic time-ordered transaction initiation.

Optional user exits are provided as follows:

- Before determining what type of request for time services was issued
- Upon expiration of a previously requested time-dependent event
- If a START request names an unknown terminal.

Time-of-day control

The EXEC CICS RESETTIME command causes CICS to reset its internal date and time of day information to accord with that of the operating system. This is done by calling DFHICP with a DFHIC TYPE=RESET macro. This macro is also issued by DFHAPTIM - the program run by the “midnight task” attached by interval control initialization - whenever it is resumed by the TI domain, i.e. at midnight.

DFHICP issues a KETI RESET_LOCAL_TIME call to the TI domain if the reason for the reset was a time of day change. This allows the TI domain to readjust its clocks to the operating system time. DFHICP then calls DFHTAJP to readjust other CICS clocks to match the operating system time and to make any necessary
Interval control

changes to the ICE chain resulting from possible changes in the time-to-expiry of time controlled ICEs. Finally DFHICP scans the ICE chain in order to process any that may have become expired as a result of the time change, and to reset the time interval for which the expiry task, DFHAPTIX, will wait, until the next ICE expires.

Control blocks

An interval control element (ICE—see Figure 78) is created for each time-dependent request received by interval control. These ICEs are chained from the CSA in expiration time-of-day sequence.

![Diagram of Interval Control Element (ICE)]

Note: An ECA (event control area) exists only after an EXEC CICS POST command.

Figure 78. Interval control element (ICE)

Expired time-ordered requests are processed by Interval Control when called from the DFHAPTIX module, which runs under a system task that has been resumed by the timer domain. The type of service represented by the expired ICE is initiated, if all resources required for the service are available, and the ICE is removed from the chain. If the resources are not available, the ICE remains on the chain and another attempt to initiate the requested service is made later.

See the CICS Data Areas manual for a detailed description of this control block.

Modules

DFHAPTIM, DFHAPTIX, DFHICP, DFHICRC, and DFHTAJP
Exits

There are three global user exit points in DFHICP: XICEXP, XICREQ, and XICTENF. See the CICS Customization Guide for further information.

Trace

The following point ID is provided for DFHICP:
• AP 00F3, for which the trace level is IC 1.

For more information about the trace points, see the CICS Trace Entries. For more information about using traces in problem determination, see the CICS Problem Determination Guide.
Chapter 48. JVM domain (SJ)

The JVM domain provides services to:
- Invoke a Java program
- Inquire/Set the JVM pool

JVM domain’s specific gates

Table 60 summarizes the JVM domain’s specific gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and whether or not the functions are available through the exit programming interface (XPI).

Table 60. JVM domain’s specific gates

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>XPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>SJIN</td>
<td>SJ 0201</td>
<td>INVOKE_JAVA_PROGRAM</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>SJ 0202</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SJIS</td>
<td>SJ 0301</td>
<td>INQUIRE_JVMPOOL</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>SJ 0302</td>
<td>SET_JVMPOOL</td>
<td>NO</td>
</tr>
</tbody>
</table>

SJIN gate, INVOKE_JAVA_PROGRAM function

The INVOKE_JAVA_PROGRAM function of the SJIN gate is used to invoke a user Java program.

Input parameters

<table>
<thead>
<tr>
<th>PROGRAM</th>
<th>Program name of the program to be invoked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRANSACTION</td>
<td>Transaction id of current transaction</td>
</tr>
<tr>
<td>JVM_PROFILE_NAME</td>
<td>The name of the profile to be used when setting up the JVM for this program to run in.</td>
</tr>
<tr>
<td>USER_CLASS</td>
<td>The name of the user’s class to be run.</td>
</tr>
</tbody>
</table>

Output parameters

<table>
<thead>
<tr>
<th>ABEND_CODE</th>
<th>The CICS abend code returned if an abend occurs.</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESPONSE</td>
<td>is the domain’s response to the call. It can have any of these values: OK</td>
</tr>
<tr>
<td>[REASON]</td>
<td>is returned when RESPONSE is EXCEPTION. Possible values are:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>JVM_START_FAILURE JVM_POOL_DISABLED JVM_PROFILE_MISSING JVM_PROFILE_INVALID SYSTEM_PROPERTIES_MISSING SYSTEM_PROPERTIES_INVALID TRANSACTION_ABENDED USER_CLASS_NOT_FOUND</td>
</tr>
</tbody>
</table>
JVM domain (SJ)

SJIS gate, INQUIRE_JVM_POOL function

The INQUIRE_JVM_POOL function of the SJIS gate is used to retrieve information about the JVM pool.

Input parameters
None.

Output parameters

- [TOTAL]
  The total number of JVMs in the pool

- [STATUS]
  The status of the JVM pool it can have one of these values:
  ENABLED|DISABLED

- [PHASINGOUT]
  The number of JVMs which are currently being phased-out.

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
None.

SJIS gate, SET_JVM_POOL function

The SET_JVM_POOL function of the SJIS gate is used to set the status of the JVM pool.

Input parameters

- [STATUS]
  It can have either of these values:
  ENABLED|DISABLED

- [TERMINATE]
  A termination to perform on the JVM pool, it can take one of these values:
  PHASEOUT | PURGE | FORCEPURGE

Output parameters

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
None.

Modules

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHSJIN</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>INVOKE_JAVA_PROGRAM</td>
</tr>
<tr>
<td>DFHSJIS</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>INQUIRE_JVMPOOL</td>
</tr>
<tr>
<td></td>
<td>SET_JVMPOOL,</td>
</tr>
<tr>
<td>Module</td>
<td>Function</td>
</tr>
<tr>
<td>--------</td>
<td>----------</td>
</tr>
</tbody>
</table>
| DFHSJCS | Is an internal module which handles the following requests:  
|        | sjcsbld (sjcs_build_jvm)  
|        | sjcsdes (sjcs_destroy_jvm)  
|        | sjcs call (sjcs_call_java_method)  
|        | sjcsreset (sjcs_reset_jvm_output_streams) |
| DFHSJPJP | Is an internal module which handles the following requests:  
|         | sjjp_process_jvm_profile |

## Exits

There are two user-replaceable modules which are used by the JVM domain.  
DFHSJJ80, which is loaded by SJ domain and used to set user-specified options for an LE enclave in which a JVM is to be started.  
DFHJVMAT, which is called during the startup of a non-resettable JVM and allows users to interogate and possibly alter environment variables which are used to moduify the starting JVM’s properties.

For further information about both these exit points see the CICS Customization Guide.

## Trace

The point IDs for the storage manager domain are of the form XMxxxx; the corresponding trace levels are XM 1, XM 2 and Exc.

For more information about the trace points, see the CICS User’s Handbook. For more information about using traces in problem determination, see the CICS Problem Determination Guide.
Chapter 49. Kernel domain (KE)

The kernel domain provides a consistent linkage and recovery environment for CICS.

The application programmer has no external interface to kernel linkage. However, services invoked by the application program result in execution of kernel linkage requests.

The CICS customization interface uses kernel linkage; this interface is described in the [CICS Customization Guide](#).

The kernel domain, with its associated trace entries and dumped storage, becomes the first point of reference for problems that cause system recovery to be invoked. The kernel domain returns errors to the caller as response codes, if they seem to be of a form such that the caller can be expected to take alternative action.

For serious system-wide errors, the kernel domain terminates CICS with a system dump.

When the kernel domain terminates CICS following a program check or abend, messages and abend codes are produced to indicate the event that caused the kernel domain recovery routines to consider that the error was not recoverable.

Kernel domain’s specific gates

Table 61 summarizes the kernel domain’s specific gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and whether or not the functions are available through the exit programming interface (XPI).

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>XPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>KEAR</td>
<td>KE 0701</td>
<td>DEREGISTER</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>KE 0702</td>
<td>READY</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>REGISTER</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>WAITPRED</td>
<td></td>
</tr>
</tbody>
</table>

3. Only the following KEDS functions are traced:
   SEND_DEFERRED_ABEND, START_PURGE_PROTECTION, STOP_PURGE_PROTECTION, and PROCESS_KETA_ERROR.

4. The CREATE_TASK function is processed by the DFHKETA module; all other KEDS functions are processed by the DFHKEDS module.
## Kernel domain (KE)

### Table 61. Kernel domain’s specific gates (continued)

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>XPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>KEDD KE 0201</td>
<td>ADD_DOMAIN</td>
<td></td>
<td>NO</td>
</tr>
<tr>
<td>KE 0202</td>
<td>INQUIRE_DOMAIN_BY_TOKEN</td>
<td></td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>INQUIRE_DOMAIN_BY_NAME</td>
<td></td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>SET_ANCHOR</td>
<td></td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>INQUIRE_ANCHOR</td>
<td></td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>ADD_GATE</td>
<td></td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>DELETE_GATE</td>
<td></td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>INQUIRE_GLOBAL_TRACE</td>
<td></td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>SET_GLOBAL_TRACE</td>
<td></td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>INQUIRE_DOMAIN_TRACE</td>
<td></td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>SET_DOMAIN_TRACE</td>
<td></td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>INQUIRE_TASK_TRACE</td>
<td></td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>SET_TASK_TRACE</td>
<td></td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>PERFORM_SYSTEM_ACTION</td>
<td></td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>SET_TRAP_OFF</td>
<td></td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>SET_TRAP_ON</td>
<td></td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>SET_DEFAULT_RECOVERY</td>
<td></td>
<td>NO</td>
</tr>
<tr>
<td>KEDS KE 0502</td>
<td>ABNORMALLY_TERMINATE_TASK</td>
<td></td>
<td>NO</td>
</tr>
<tr>
<td>KE 0503</td>
<td>CREATE_TASK$^4$</td>
<td></td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>CREATE_TCB</td>
<td></td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>DETACH_TERMINATED_OWN_TCBS</td>
<td></td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>END_TASK</td>
<td></td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>FREE_TCBS</td>
<td></td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>PUSH_TASK</td>
<td></td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>POP_TASK</td>
<td></td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>READ_TIME</td>
<td></td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>RESET_TIME</td>
<td></td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>START_RUNAWAY_TIMER</td>
<td></td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>STOP_RUNAWAY_TIMER</td>
<td></td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>RESTORE_STIMER</td>
<td></td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>SEND_DEFERRED_ABEND</td>
<td></td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>START_PURGE_PROTECTION</td>
<td></td>
<td>YES</td>
</tr>
<tr>
<td></td>
<td>STOP_PURGE_PROTECTION</td>
<td></td>
<td>YES</td>
</tr>
<tr>
<td></td>
<td>PROCESS_KETA_ERROR</td>
<td></td>
<td>NO</td>
</tr>
<tr>
<td>KEGD KE 0401</td>
<td>INQUIRE_KERNEL</td>
<td></td>
<td>NO</td>
</tr>
<tr>
<td>KE 0402</td>
<td>SET_KERNEL</td>
<td></td>
<td>NO</td>
</tr>
<tr>
<td>KEIN KE 0301</td>
<td>INITIALISE_KERNEL</td>
<td></td>
<td>NO</td>
</tr>
<tr>
<td>KE 0302</td>
<td>SET_STATIC_TASKS</td>
<td></td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>ADD_DYNAMIC_TASK</td>
<td></td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>ADD_TEMPORARY_STATIC_TASK</td>
<td></td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>DELETE_TASKS</td>
<td></td>
<td>NO</td>
</tr>
<tr>
<td>KETI KE 0101</td>
<td>ADJUST_STCK_TO_LOCAL</td>
<td></td>
<td>NO</td>
</tr>
<tr>
<td>KE 0102</td>
<td>CONVERT_TO_DECIMAL_TIME</td>
<td></td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>CONVERT_TO_STCK_FORMAT</td>
<td></td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>INQUIRE_DATE_FORMAT</td>
<td></td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>INQ_LOCAL_DATETIME_DECIMAL</td>
<td></td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>NOTIFY_RESET</td>
<td></td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>REQUEST_NOTIFY_OF_A_RESET</td>
<td></td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>RESET_LOCAL_TIME</td>
<td></td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>SET_DATE_FORMAT</td>
<td></td>
<td>NO</td>
</tr>
<tr>
<td>KEXM KE 0601</td>
<td>TRANSACTION_INITIALISATION</td>
<td></td>
<td>NO</td>
</tr>
<tr>
<td>KE 0602</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
KEAR gate, DEREGISTER function

The DEREGISTER function of the KEAR gate is used when performing a normal shutdown (and optionally at an immediate shutdown) to deregister CICS from the MVS automatic restart manager.

Input parameters
None.

Output parameters
RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|DISASTER|KERNERROR|PURGED

KEAR gate, READY function

The READY function of the KEAR gate is used at the end of CICS initialization to indicate to the MVS automatic restart manager that this CICS region is ready for work.

Input parameters
None.

Output parameters
RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|DISASTER|KERNERROR|PURGED

KEAR gate, REGISTER function

The REGISTER function of the KEAR gate is used very early in CICS initialization to register CICS with the MVS automatic restart manager.

Input parameters
None.

Output parameters
RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|INVALID|DISASTER

KEAR gate, WAITPRED function

The WAITPRED function of the KEAR gate is used to wait on predecessors in the restart policy for this CICS region, to ensure that prerequisite subsystems are available to CICS.

Input parameters
None.

Output parameters
RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|DISASTER|KERNERROR|PURGED
**KEDD gate, ADD_DOMAIN function**

The ADD_DOMAIN function of the KEDD gate is used to add a new domain to the domain table.

**Input parameters**

- **DOMAIN_NAME**
  - is the 8-character domain name for the new domain to be added.

- **DOMAIN_TOKEN**
  - is the 31-bit constant that uniquely identifies the domain, for example, DFHSM_DOMAIN for storage manager domain.

- **ENTRY_POINT**
  - is the 31-bit address of the entry point for that domain, for example, A(X'80000000' + DFHSMDM) for storage manager domain.

- **[DOMAIN_AFFINITY]**
  - is the TCB that the domain has affinity with for TERMINATE_DOMAIN. It can have any one of these values:
    - STEP|R0|QR|C0|SZ

**Output parameters**

- **RESPONSE**
  - is the domain’s response to the call. It can have any of these values:
    - OK|EXCEPTION|INVALID|DISASTER

  - **[REASON]**
    - is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>DUPLICATE_DOMAIN_TOKEN, DUPLICATE_DOMAIN_NAME</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_DOMAIN_TOKEN, INVALID_ENTRY_POINT</td>
</tr>
</tbody>
</table>

**KEDD gate, INQUIRE_DOMAIN_BY_TOKEN function**

The INQUIRE_DOMAIN_BY_TOKEN function of the KEDD gate is used to return the domain name for a specified domain token.

**Input parameters**

- **DOMAIN_TOKEN**
  - is the 31-bit constant that uniquely identifies the domain.

**Output parameters**

- **DOMAIN_NAME**
  - is the 8-character domain name for the new domain to be added.

- **RESPONSE**
  - is the domain’s response to the call. It can have any of these values:
    - OK|EXCEPTION|INVALID|DISASTER

  - **[REASON]**
    - is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:
### KEDD gate, INQUIRE_DOMAIN_BY_NAME function

The INQUIRE_DOMAIN_BY_NAME function of the KEDD gate is used to return the domain token for a given domain name.

**Input parameters**

**DOMAIN_NAME**
is the 8-character domain name for the new domain to be added.

**Output parameters**

**DOMAIN_TOKEN**
is the 31-bit constant that uniquely identifies the domain.

**RESPONSE**
is the domain’s response to the call. It can have any of these values: \( \text{OK} | \text{EXCEPTION} | \text{INVALID} | \text{DISASTER} \)

**[REASON]**
is returned when RESPONSE is EXCEPTION. It has this value:

\( \text{DOMAIN_NAME_NOT_FOUND} \)

### KEDD gate, SET_ANCHOR function

The SET_ANCHOR function of the KEDD gate is used to establish the calling domain’s global storage pointer.

**Input parameters**

**ANCHOR**
is the 31-bit address of the domain’s global storage.

**Output parameters**

**RESPONSE**
is the domain’s response to the call. It can have any of these values: \( \text{OK} | \text{EXCEPTION} | \text{INVALID} | \text{DISASTER} \)

**[REASON]**
is returned when RESPONSE is INVALID. It has this value:

\( \text{INVALID_DOMAIN_TOKEN} \)

### KEDD gate, INQUIRE_ANCHOR function

The INQUIRE_ANCHOR function of the KEDD gate is used to return the specified domain’s global storage pointer to the caller. If the domain token is omitted, the calling domain is assumed.

**Input parameters**

**[DOMAIN_TOKEN]**
is the 31-bit constant that uniquely identifies the domain.
Kernel domain (KE)

Output parameters

ANCHOR
is the 31-bit address of the domain’s global storage.

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|INVALID|DISASTER

[REASON]
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>DOMAIN_TOKEN_NOT_FOUND</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_DOMAIN_TOKEN</td>
</tr>
</tbody>
</table>

KEDD gate, ADD_GATE function

The ADD_GATE function of the KEDD gate is used to update the domain table to add a new gate to the calling domain’s gate table.

Input parameters

GATE_INDEX
is the 31-bit constant that uniquely identifies the gate in the domain’s gate table.

ENTRY_POINT
is the 31-bit address of the entry point for the gate.

Output parameters

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|INVALID|DISASTER

[REASON]
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>DUPLICATE_GATE_INDEX</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_ENTRY_POINT, INVALID_GATE_INDEX, INVALID_DOMAIN_TOKEN</td>
</tr>
</tbody>
</table>

KEDD gate, DELETE_GATE function

The DELETE_GATE function of the KEDD gate is used to delete an existing gate from the calling domain’s gate table.

Input parameters

GATE_INDEX
is the 31-bit constant that uniquely identifies the gate in the domain’s gate table.
Output parameters

RESPONSE is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | INVALID | DISASTER

[REASON] When RESPONSE is INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>INVALID_GATE_INDEX, INVALID_DOMAIN_TOKEN</td>
</tr>
</tbody>
</table>

KEDD gate, INQUIRE_GLOBAL_TRACE function

The INQUIRE_GLOBAL_TRACE function of the KEDD gate is used to return the value of the global trace flags to the caller.

Input parameters

None.

Output parameters

[MASTER_TRACE_FLAG] determines whether tracing, for any of the trace destinations, is active. It can have either of these values:

ON | OFF

[System_TRACE_FLAG] determines whether tracing is allowed for tasks for which standard tracing is in effect. It can have either of these values:

ON | OFF

RESPONSE is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | INVALID | DISASTER

KEDD gate, SET_GLOBAL_TRACE function

The SET_GLOBAL_TRACE function of the KEDD gate is used to store the value of the global trace flags within the kernel.

Input parameters

[MASTER_TRACE_FLAG] determines whether tracing, for any of the trace destinations, is active. It can have either of these values:

ON | OFF

[System_TRACE_FLAG] determines whether tracing is allowed for tasks for which standard tracing is in effect. It can have either of these values:

ON | OFF

Output parameters

RESPONSE is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | INVALID | DISASTER
Kernel domain (KE)

**KEDD gate, INQUIRE_DOMAIN_TRACE function**

The INQUIRE_DOMAIN_TRACE function of the KEDD gate is used to return the value of the specified domain’s trace flags to the caller. If the domain token is omitted, the calling domain is assumed.

**Input parameters**

[DOMAIN_TOKEN] is the 31-bit constant that uniquely identifies the domain.

**Output parameters**

[STANDARD_TRACE_FLAGS] is the set of 32 bits which determines selectivity of tracing within the domain for standard tasks.

[SPECIAL_TRACE_FLAGS] is the set of 32 bits which determines selectivity of tracing within the domain for special tasks.

**RESPONSE** is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER

[REASON] is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>DOMAIN_TOKEN_NOT_FOUND</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_DOMAIN_TOKEN</td>
</tr>
</tbody>
</table>

**KEDD gate, SET_DOMAIN_TRACE function**

The SET_DOMAIN_TRACE function of the KEDD gate is used to store the value of the specified domain’s trace flags in the kernel. If the domain token is omitted, the calling domain is assumed.

The current task’s stack entries are updated to reflect the change. The trace count is incremented so that all other tasks have their stack entries refreshed when they are next dispatched.

**Input parameters**

[DOMAIN_TOKEN] is the 31-bit constant that uniquely identifies the domain.

[STANDARD_TRACE_FLAGS] is the set of 32 bits which determines selectivity of tracing within the domain for standard tasks.

[SPECIAL_TRACE_FLAGS] is the set of 32 bits which determines selectivity of tracing within the domain for special tasks.

**Output parameters**

**RESPONSE** is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER
KEDD gate, INQUIRE_TASK_TRACE function

The INQUIRE_TASK_TRACE function of the KEDD gate is used to return the value of the calling task’s trace flag to the caller.

Input parameters
None.

Output parameters

[TRACE_TYPE]

determines whether standard, special, or no tracing is required for this task. It can have any one of these values:
STANDARD|SPECIAL|SUPPRESSED

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|INVALID|DISASTER

KEDD gate, SET_TASK_TRACE function

The SET_TASK_TRACE function of the KEDD gate is used to store the value of the task trace flag in the current task’s task table entry.

The current task’s stack entries are updated to reflect the change.

Input parameters

TRACE_TYPE
determines whether standard, special, or no tracing is required for this task. It can have any one of these values:
STANDARD|SPECIAL|SUPPRESSED

Output parameters

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|INVALID|DISASTER

KEDD gate, PERFORM_SYSTEM_ACTION function

The PERFORM_SYSTEM_ACTION function of the KEDD gate is used in exceptional circumstances either to terminate CICS (with or without a dump) or to take an MVS SDUMP.

Normally, these services are invoked from domains during preinitialization before the dump domain is available.

5. Task table: A task table is a logical block of tasks, allocated together by the Kernel domain, and used to simplify the process of dynamically adding new tasks. Task tables are chained together, and vary in number.
Input parameters

[TERMINATE_SYSTEM (YES, NO)]
  specifies whether CICS is to be terminated or not. It can have either of these values:
  YES|NO

[DUMP_SYSTEM (YES, NO)]
  specifies whether an MVS SDUMP is to be taken or not. It can have either of these values:
  YES|NO

[NORMAL_TERMINATION(YES, NO)]
  specifies whether CICS is being terminated normally. Normal termination includes controlled and immediate shutdowns. It can have either of these values:
  YES|NO

  The default value is NO.

Output parameters

RESPONSE
  is the domain’s response to the call. It can have any of these values:
  OK|EXCEPTION|INVALID|DISASTER

KEDD gate, SET_TRAP_OFF function

The SET_TRAP_OFF function of the KEDD gate is used to reset the kernel global trap point.

Input parameters

None.

Output parameters

RESPONSE
  is the domain’s response to the call. It can have any of these values:
  OK|EXCEPTION|INVALID|DISASTER

KEDD gate, SET_TRAP_ON function

The SET_TRAP_ON function of the KEDD gate is used to set a kernel global trap point.

Input parameters

ENTRY_POINT
  is the 31-bit address of the kernel global trap.

Output parameters

RESPONSE
  is the domain’s response to the call. It can have any of these values:
  OK|EXCEPTION|INVALID|DISASTER

[REASON]
  is returned when RESPONSE is INVALID. It has this value:
  INVALID_ENTRY_POINT
**KEDD gate, SET_DEFAULT_RECOVERY function**

The SET_DEFAULT_RECOVERY function of the KEDD gate is used to establish the calling domain’s default recovery routine. Used by the Application domain to identify DFHSRP as its default recovery routine.

**Input parameters**

ENTRY_POINT

is the 31-bit address of the entry point for the recovery routine.

**Output parameters**

RESPONSE

is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | INVALID | DISASTER

[REASON]

is returned when RESPONSE is INVALID. It has this value:

INVALID_DOMAIN_TOKEN

---

**KEDS gate, ABNORMALLY_TERMINATE_TASK function**

The ABNORMALLY_TERMINATE_TASK function of the KEDS gate identifies the task which is to be abnormally terminated.

**Input parameters**

TASK_TOKEN

identifies the task which is to be abnormally terminated.

**Output parameters**

RESPONSE

is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION

[REASON]

is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>TERMINATE_FAILED</td>
</tr>
</tbody>
</table>

---

**KEDS gate, CREATE_TASK function**

The CREATE_TASK function of the KEDS gate is used to allocate a new executable task from the task table\(^6\).

When the task is first dispatched, the Kernel domain issues a KEDS_TASK_REPLY request, which passes control to the Dispatcher domain’s task reply gate. (See “KEDS format, TASK_REPLY function” on page 753)

The attach token input on the CREATE_TASK request is passed back to the dispatcher domain on the TASK_REPLY, to identify the CREATE_TASK and TASK_REPLY pair.

**Note:** The CREATE_TASK function is processed by the DFHKETA module.

---

\(^6\) **Task table**: A task table is a logical block of tasks, allocated together by the Kernel domain, and used to simplify the process of dynamically adding new tasks. Task tables are chained together, and vary in number.
Kernel domain (KE)

**Input parameters**

**ALLOCATE**
indicates whether or not the returned task should be allocated from those tasks pre-allocated for MXT. It can either of these values:
- **STATIC**
- **DYNAMIC**

**ATTACH_TOKEN**
is the 31-bit token that uniquely identifies the request. This token is returned on the corresponding TASK_REPLY to identify the request.

**Output parameters**

**TASK_TOKEN**
is the 31-bit token that uniquely identifies the newly created task.

**RESPONSE**
is the domain’s response to the call. It can have any of these values:
- **OK**
- **EXCEPTION**
- **INVALID**
- **DISASTER**

**[REASON]**
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>INQUIRE_ERROR</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>ADD_TASK_ERROR</td>
</tr>
</tbody>
</table>

**KEDS gate, CREATE_TCB function**

The CREATE_TCB function of the KEDS gate creates the default task 7 for a new MVS TCB, and MVS posts the TCB to start execution.

The kernel invokes the dispatcher domain at its KEDS gate with a TCB_REPLY request, under the new TCB’s default task.

The attach token is used to identify the CREATE_TCB and TCB_REPLY pair.

**Input parameters**

**ATTACH_TOKEN**
is the 31-bit token that uniquely identifies the request. This token is returned on the corresponding TCB_REPLY to identify the request.

**ESSENTIAL_TCB**
indicates whether CICS is to be terminated if a TCB in this mode has its ESTAE exit driven for a non recoverable error.

**EXEC_CAPABLE**
indicates whether support should be provided under the new TCB for CICS API commands.

**INHERIT_SUBSPACE**
indicates whether TCBs in this mode are to inherit the subspace of the attaching TCB.

---

7. **Default task**: The task, associated with the TCB, that executes the dispatcher loop which chooses the next CICS task (system or non-system) to be dispatched, or if no CICS task is to be dispatched, issues an MVS WAIT.
LE_ENVIRONMENT
indicates whether CICS should tell LE that it is running in a CICS environment under this TCB. If LE_CICS is specified, LE will issue CICS API commands.

[MODE]
specifies the mode of the new TCB. It can have any of these values:
RO|QR|CO|SZ|RP|FO

PARENT_MODENAME
identifies the mode of the TCB that is to ATTACH the new TCB.

PRTY_RELATIVE_TO_QR
gives the priority of this TCB relative to QR.

TCB_KEY
specifies the key to be specified on the ATTACH of TCBs in this mode. The value ends up in TCBPKF.

Output parameters

TASK_TOKEN
is the 31-bit token that uniquely identifies the new TCB’s task.

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|INVALID|DISASTER

[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>INQUIRE_ERROR</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>ADD_TASK_ERROR, ADD_TCB_ERROR, ATTACH_KTCB_ERROR</td>
</tr>
</tbody>
</table>

KEDS gate, DETACH_TERMINATEDOWN_TCB function
The DETACH_TERMINATEDOWN_TCB function of the KEDS gate detaches any terminated TCBs which were attached by the TCB on which this function is invoked.

Input parameters
None.

Output parameters

RESPONSE
is the domain’s response to the call. It can have this value:
OK

KEDS gate, END_TASK function
The END_TASK function of the KEDS gate is used to free any resources that have been acquired by the kernel domain during the lifetime of the current task and need freeing before the end of the task.

Input parameters
None.
**Kernel domain (KE)**

### Output parameters

**RESPONSE**

is the domain’s response to the call. It can have any one of these values:

OK|EXCEPTION|INVALID|DISASTER

---

### KEDS gate, FREE_TCBS function

The FREE_TCBS function of the KEDS gate conditionally frees control blocks, in collaboration with the Dispatcher for re-use, associated with any detached TCBs.

#### Input parameters

None.

#### Output parameters

**RESPONSE**

is the domain’s response to the call. It can have this value:

OK

---

### KEDS gate, PUSH_TASK function

Given a TCB executing its default task, the PUSH_TASK function of the KEDS gate is used to make it execute a CICS task instead.

#### Input parameters

**TASK_TOKEN**

is the 31-bit token that identifies the CICS task to be executed.

#### Output parameters

**[INTERVAL]**

is a doubleword containing the CPU time used by the task while it was pushed.

**RESPONSE**

is the domain’s response to the call. It can have any one of these values:

OK|EXCEPTION|INVALID|DISASTER

---

### KEDS gate, POP_TASK function

Given a TCB executing the current CICS task, the POP_TASK function of the KEDS gate is used to make it execute its default task instead.

#### Input parameters

None.

#### Output parameters

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER

---

### KEDS gate, READ_TIME function

The READ_TIME function of the KEDS gate is used to obtain the total CPU time that the current task has taken so far and the accumulated CPU time for the current TCB.

#### Input parameters

None.
Output parameters

[INTERVAL]
A doubleword containing the total CPU time used so far.

[ACCUM_TIME]
A doubleword containing the accumulated CPU time used so far by the current TCB.

RESPONSE
is the domain’s response to the call. It can have any of these values: OK|EXCEPTION|INVALID|DISASTER

KEDS gate, RESET_TIME function
The RESET_TIME function of the KEDS gate is used to reset the total CPU time that the current task has taken so far.

Input parameters
None.

Output parameters

[INTERVAL]
A doubleword containing the total CPU time used so far.

RESPONSE
is the domain’s response to the call. It can have any of these values: OK|EXCEPTION|INVALID|DISASTER

KEDS gate, STOP_RUNAWAY_TIMER function
The STOP_RUNAWAY_TIMER function of the KEDS gate is used to inhibit runaway detection for the current task. The remaining runaway interval is preserved until a START_RUNAWAY_TIMER request is issued. The stop runaway count is incremented by one; this allows STOP_RUNAWAY_TIMER requests to be nested.

Input parameters
None.

Output parameters

RESPONSE
is the domain’s response to the call. It can have any of these values: OK|EXCEPTION|INVALID|DISASTER

KEDS gate, START_RUNAWAY_TIMER function
The START_RUNAWAY_TIMER function of the KEDS gate is used to resume runaway timing for the current task. This reduces the stop runaway count by one. The timer is resumed only when all outstanding STOP_RUNAWAY_TIMER requests have been canceled.

Input parameters
None.

Output parameters

RESPONSE
is the domain’s response to the call. It can have any of these values: OK|EXCEPTION|INVALID|DISASTER
KEDS gate, RESTORE_STIMER function

The RESTORE_STIMER function of the KEDS gate is used to restore the kernel’s STIMER exit after MVS requests that use the MVS STIMER macro internally.

Input parameters
None.

Output parameters
RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|INVALID|DISASTER

KEDS gate, SEND_DEFERRED_ABEND function

The SEND_DEFERRED_ABEND function of the KEDS gate is used by the transaction manager to implement the deferred purge function. If a purge request is made against a task that is not in a suitable state to be purged, this function defers the abend of that task until the task is no longer protected against purge.

This function is used by the transaction manager to implement the deferred purge function.

Input parameters
[DS_TASK_TOKEN]
is the 31-bit dispatcher token that identifies the CICS task to be abended. If not supplied, DS_TASK_TOKEN defaults to the current task.

ABEND_CODE
is the four-character abend code for the abend.

[FORCE]
indicates whether or not the deferred abend is to be forced. It can have either of these values:
YES|NO

The default is NO.

Output parameters
RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|INVALID|DISASTER

KEDS gate, START_PURGE_PROTECTION function

The START_PURGE_PROTECTION function of the KEDS gate is used to inhibit purge, but not force-purge, for the current task.

In general, each START_PURGE_PROTECTION call should have a corresponding STOP_PURGE_PROTECTION function call to end the purge protection period on completion of any program logic that needs such protection.

This function increments by one the purge protection count for the task. You can issue several START_PURGE_PROTECTION commands for the same task, to increase the count for the task. (To enable the task to purged, the count must be decremented to zero by issuing STOP_PURGE_PROTECTION commands.)
**Input parameters**
None.

**Output parameters**

**RESPONSE**
is the domain’s response to the call. It can have any of these values:

- OK
- EXCEPTION
- INVALID
- DISASTER

---

**KEDS gate, STOP_PURGE_PROTECTION function**

The STOP_PURGE_PROTECTION function of the KEDS gate is used to enable again purge for the current task after purge has been suspended by a previous START_PURGE_PROTECTION function call.

This function decrements by one the purge protection count for the task. To enable the task to purged, the count must be decremented to zero by issuing the appropriate number of STOP_PURGE_PROTECTION commands.

You must design your exit programs to ensure that purge protection is correctly cancelled. For more information about using these functions to stop and start purge protection, see the [CICS Customization Guide](#).

**Input parameters**
None.

**Output parameters**

**RESPONSE**
is the domain’s response to the call. It can have any of these values:

- OK
- EXCEPTION
- INVALID
- DISASTER

---

**KEDS gate, PROCESS_KETA_ERROR function**

The PROCESS_KETA_ERROR function of the KEDS gate is used to handle any errors for the DFHKETA module. (The DFHKETA module handles the performance sensitive KEDS functions, and calls the DFHKEDS module when its recovery routine is invoked.)

**Input parameters**

**ERROR_DATA**
address of the error data that describes the error that has occurred in the DFHKETA module.

**Output parameters**

**RESPONSE**
is the domain’s response to the call. It can have any of these values:

- OK
- EXCEPTION
- INVALID
- DISASTER

---

**KEGD gate, INQUIRE_KERNEL function**

The INQUIRE_KERNEL function of the KEGD gate is used to obtain the global data maintained by the kernel.

**Input parameters**
None.
Kernel domain (KE)

Output parameters

[CICS_SVC_NUMBER]
  is the 8-bit CICS service SVC number.

[SPECIFIC_APPLID]
  is the 8-character specific applid that identifies the CICS system in the
  VTAM network.

[ GENERIC_APPLID]
  is the 8-character generic applid that identifies the active and alternate
  CICS systems to VTAM in an XRF environment.

[XRF_COMMAND_LIST]
  is the 8-character name of the command list table used by the XRF
  alternate CICS region.

[ALTERNATE_XRF_IDS]
  is the 8-character name of the recoverable service table used if the CICS
  region is running with XRF and DBCTL.

[SYSID]
  is the 4-character ZCP system entry name.

[SIT_NAME]
  is the 8-character SIT name.

[OS_PARMS]
  is the 8-byte block containing the 31-bit address and 31-bit length of the
  MVS parameters.

[OP_SYS]
  is the 1-character operating system identifier, for example, ‘B’ = MVS.

[OP_REL]
  is the 2-byte operating system release and modification level.

[HPO]
  specifies whether CICS is to use the VTAM high performance option. It can
  have either of these values:
  YES|NO

[SYSTEM_RUNAWAY_LIMIT]
  the ICVR time to be used by all tasks that have been defined to have the
  default runaway limit in the system.

[CPU_MONITORING]
  specifies whether the kernel is to perform CPU monitoring. It can have
  either of these values:
  YES|NO

RESPONSE
  is the domain’s response to the call. It can have any of these values:
  OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

KEGD gate, SET_KERNEL function

The SET_KERNEL function of the KEGD gate is used to change the global data
maintained by the kernel.

Input parameters

[CICS_SVC_NUMBER]
  is the 8-bit CICS service SVC number.
[SPECIFIC_APPLID]
is the 8-character specific applid that identifies the CICS system in the
VTAM network.

[ GENERIC_APPLID]
is the 8-character generic applid that identifies the active and alternate
CICS systems to VTAM in an XRF environment.

[XRF_COMMAND_LIST]
is the 8-character name of the command list table used by the XRF
alternate CICS region.

[ALTERNATE_XRF_IDS]
is the 8-character name of the recoverable service table used if the CICS
region is running with XRF and DBCTL.

[SYSID]
is the 4-character ZCP system entry name.

[SIT_NAME]
is the 8-character name of the system initialization table.

[HPO] specifies whether CICS is to use the VTAM high performance option. It can
have either of these values:
YES|NO

[System_RUNAWAY_LIMIT]
the ICVR time to be used by all tasks that have been defined to have the
default runaway limit in the system.

[CPU_MONITORING]
specifies whether the kernel is to perform CPU monitoring. It can have
either of these values:
YES|NO

Output parameters

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

KETI gate, RESET_LOCAL_TIME function
The RESET_LOCAL_TIME function of the KETI gate is used by the AP domain to
inform KETI that a local time reset has occurred.

Input parameters
None.

Output parameters

RESPONSE
is the domain’s response to the call. It can have any one of these values:
OK|INVALID|KERNERROR|PURGED|DISASTER

KETI gate, REQUEST_NOTIFY_OF_A_RESET function
The REQUEST_NOTIFY_OF_A_RESET function of the KETI gate requests a
shoulder tap from KETI whenever the local time is reset.

Input parameters
None.
Kernel domain (KE)

**Output parameters**

**RESPONSE**

is the domain’s response to the call. It can have any one of these values:

OK|INVALID|KERNERROR|PURGED|DISASTER

**KETI gate, SET_DATE_FORMAT function**

The SET_DATE_FORMAT function of the KETI gate is used to set the date format for the timer domain.

**Input parameters**

**DATE_FORMAT**

is the format to be set as the default for the timer domain. It can have any one of these values:

YYMMDD|DDMMYY|MMDDYY

**Output parameters**

**RESPONSE**

is the domain’s response to the call. It can have any one of these values:

OK|INVALID|KERNERROR|PURGED|DISASTER

**KETI gate, INQUIRE_DATE_FORMAT function**

The INQUIRE_DATE_FORMAT function of the KETI gate is used to return the current date format.

**Input parameters**

None.

**Output parameters**

**DATE_FORMAT**

is the current default date format for the timer domain. It can have any one of these values:

YYMMDD|DDMMYY|MMDDYY

**RESPONSE**

is the domain’s response to the call. It can have any one of these values:

OK|INVALID|KERNERROR|PURGED|DISASTER

**KETI gate, INQ_LOCAL_DATETIME_DECIMAL function**

The INQ_LOCAL_DATETIME_DECIMAL function of the KETI gate is used to return the local date, and the local time in decimal format.

**Input parameters**

None.

**Output parameters**

**DECIMAL_DATE**

is an 8-character date in the format determined by FULL_DATE_FORMAT.

**DECIMAL_TIME**

is the current local decimal time in the format HHMMSS.

**DECIMAL_MICROSECONDS**

is the 6-character microseconds portion of DECIMAL_TIME.
FULL_DATE_FORMAT
is the current full date format determined by the default date format of the timer domain. It can have any one of these values:
YYYYMMDD|DDMMYYYY|MMDDYYYY

RESPONSE
is the domain's response to the call. It can have any one of these values:
OK|INVALID|KERNERROR|PURGED|DISASTER

KETI gate, CONVERT_TO_DECIMAL_TIME function
The CONVERT_TO_DECIMAL_TIME function of the KETI gate is used to convert dates and times in the internal store clock (STCK) format to decimal format.

Input parameters
STCK_TIME
is a doubleword containing a date and time in STCK format.

Output parameters
DECIMAL_DATE
is an 8-character date in the format determined by FULL_DATE_FORMAT

DECIMAL_TIME
is the current local decimal time in the format HHMMSS

DECIMAL_MICROSECONDS
is the 6-character microseconds portion of DECIMAL_TIME

FULL_DATE_FORMAT
is the current full date format determined by the default date format of the timer domain. It can have any one of these values:
YYYYMMDD|DDMMYYYY|MMDDYYYY

RESPONSE
is the domain's response to the call. It can have any one of these values:
OK|INVALID|KERNERROR|PURGED|DISASTER

KETI gate, CONVERT_TO_STCK_FORMAT function
The CONVERT_TO_STCK_FORMAT function of the KETI gate is used to convert times and dates to STCK format.

Input parameters
DECIMAL_TIME
is the current local decimal time in the format HHMMSS.

[DECIMAL_DATE]
is an optional 8-character date in the format determined either by FULL_DATE_FORMAT or by the default for the timer domain if FULL_DATE_FORMAT is omitted.

[INSTANCE]
is required only if DECIMAL_DATE is omitted. It can have either of these values:
LAST|NEXT

[FULL_DATE_FORMAT]
is the current full date format. It can have any one of these values:
YYYYMMDD|DDMMYYYY|MMDDYYYY
Kernel domain (KE)

Output parameters

STCK_TIME
is a doubleword containing the GMT STCK value corresponding to the input local time.

RESPONSE
is the domain’s response to the call. It can have any one of these values:
OK| INVALID| KERNERROR| PURGED| DISASTER

KEXM gate, TRANSACTION_INITIALISATION function

The TRANSACTION_INITIALISATION function of the KEXM gate is used to perform kernel initialisation during XM task-reply.

Input parameters

TRANSACTION_TOKEN
is a token identifying the transaction for which kernel initialization is to be performed.

Output parameters

RESPONSE
is the domain’s response to the call. It can have any one of these values:
OK| INVALID| KERNERROR| PURGED| DISASTER

Kernel domain’s generic formats

Table 62 describes the generic formats owned by the kernel domain, and shows the functions performed on the calls.

<table>
<thead>
<tr>
<th>Format</th>
<th>Calling module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>KEDS</td>
<td>DFHKETA</td>
<td>TASK_REPLY</td>
</tr>
<tr>
<td></td>
<td>DFHKETCB</td>
<td>TCB_REPLY</td>
</tr>
<tr>
<td>KETI</td>
<td>DFHKETI</td>
<td>NOTIFY_RESET</td>
</tr>
</tbody>
</table>

In the descriptions of the formats that follow, the “input” parameters are input not to the kernel domain, but to the domain being called by the kernel domain. Similarly, the “output” parameters are output by the domain that was called by the kernel domain, in response to the call.

KEDS format, TASK_REPLY function

The TASK_REPLY function of the KEDS format is issued by the kernel to the issuer of CREATE_TASK, under the new task.

Input parameters

ATTACH_TOKEN
is the 31-bit token that uniquely identifies the corresponding CREATE_TASK request.

TASK_TOKEN
is the 31-bit token that uniquely identifies the new task.
Output parameters
RESPONSE is the domain’s response to the call. It can have any of these values:
OK|INVALID|KERNERROR|PURGED|DISASTER

KEDS format, TCB_REPLY function
The TCB_REPLY function of the KEDS format is issued by the kernel to the issuer of CREATE_TCB, under the new TCB’s default task.

Input parameters
ATTACH_TOKEN is the 31-bit token that uniquely identifies the corresponding CREATE_TCB request.

TASK_TOKEN is the 31-bit token that uniquely identifies the new TCB’s task.

Output parameters
RESPONSE is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|INVALID|DISASTER

KETI format, NOTIFY_RESET function
The NOTIFY_RESET function of the KETI format is used by KETI itself to inform domains that a RESET has occurred.

Input parameters
None.

Output parameters
RESPONSE is the domain’s response to the call. It can have any one of these values:
OK|KERNERROR|PURGED|DISASTER

Control blocks
Figure 79 on page 754 shows the MVS TCB structure used by CICS. Other TCBs are attached under the quasi-reentrant TCB by IBM DATABASE 2 (DB2) or IMS/ESA code, if those products are being used.
Kernel domain (KE)

**JOB-STEP TCB**
- Runs early initialization and late termination

**FILE OWNING TCB**
- Used to OPEN and CLOSE user files managed by File Control

**RESOURCE-OWNING TCB**
- Runs CICS code containing imbedded MVS WAIT, e.g., program load

**QUASI-REENTRANT TCB**
- Runs CICS code that is non-reentrant, and all application code

**Concurrent TCB**
- Runs some CICS code only if SUBTASKS\(\leq 0\)

**Secondary LU TCB**
- Used for FEPI work only if FEPI is being used

**ONC/RPC OWNING TCB**
- Used to make calls to the TCP/IP address space only if the ONC/RPC Feature is being used

**SL TCB**
- Required to run a JVM to interpret programs defined to CICS with JAVA language

**JB TCBs**
- The sockets listener TCB

**SO TCB**
- Sockets I/O TCB

**SH TCBs**
- Required for hot-pooling

**CAVM TCB**
- Surveillance and message transmission

**Transient TCBs**
- Communicate with JES and with MVS

**Attached only if XRF=YES**

**CAVM TCB**
- Attached only if XRF=YES

**GPS #1**
- General use

**GPS #2**
- General use

**GPS #3**
- Spooler input operations

**GPS #4**
- Spooler output operations

**H8 TCBs**
- Required for hot-pooling

**S8 TCBs**
- Required for secure sockets layer

**JO TCBs**
- Spooler input operations

**SL TCB**
- Spooler output operations

**Figure 79. MVS TCB structure used by CICS**
<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHKEAR</td>
<td>Implements KEAR service requests.</td>
</tr>
<tr>
<td>DFHKEDCL</td>
<td>Implements domain call requests.</td>
</tr>
<tr>
<td>DFHKEDD</td>
<td>Services KEDD-format requests.</td>
</tr>
<tr>
<td>DFHKEDRT</td>
<td>Implements domain return requests.</td>
</tr>
<tr>
<td>DFHKEDS</td>
<td>Services KEDS-format requests.</td>
</tr>
<tr>
<td>DFHKEDUF</td>
<td>Offline dump formatting routine to format the kernel domain control blocks.</td>
</tr>
<tr>
<td>DFHKEEDA</td>
<td>Handles deferred abends</td>
</tr>
<tr>
<td>DFHKEGD</td>
<td>Services KEGD-format requests.</td>
</tr>
<tr>
<td>DFHKEIN</td>
<td>Implements kernel domain initialization.</td>
</tr>
<tr>
<td>DFHKELCL</td>
<td>Implements LIFO Push.</td>
</tr>
<tr>
<td>DFHKELOC</td>
<td>Offline dump formatting routine to locate the kernel domain anchor blocks.</td>
</tr>
<tr>
<td>DFHKELRT</td>
<td>Implements LIFO Pop.</td>
</tr>
<tr>
<td>DFHKERCD</td>
<td>Constructs the kernel domain error data for error handling routines.</td>
</tr>
<tr>
<td>DFHKERER</td>
<td>Updates the kernel domain error table for error handling routines.</td>
</tr>
<tr>
<td>DFHKERET</td>
<td>Implements RESET_ADDRESS requests.</td>
</tr>
<tr>
<td>DFHKERKE</td>
<td>Handles KERNERROR responses for domain call requests which cannot handle them.</td>
</tr>
<tr>
<td>DFHKERPC</td>
<td>Implements recovery percolation both from RECOVERY_PERCOLATE requests and also other recovery events that, because of the existing environment, must be percolated.</td>
</tr>
<tr>
<td>DFHKERRI</td>
<td>Responsible for actually passing control to a recovery routine.</td>
</tr>
<tr>
<td>DFHKERRQ</td>
<td>Implements RECOVERY_REQUEST requests.</td>
</tr>
<tr>
<td>DFHKERRU</td>
<td>Implements runaway task error handling.</td>
</tr>
<tr>
<td>DFHKERRX</td>
<td>Implements RECOVERY_EXIT requests.</td>
</tr>
<tr>
<td>DFHKESCL</td>
<td>Implements subroutine call requests.</td>
</tr>
<tr>
<td>DFHKESFM</td>
<td>Handles freeing of stack segments.</td>
</tr>
<tr>
<td>DFHKESGM</td>
<td>Handles allocation of new stack segments.</td>
</tr>
<tr>
<td>DFHKESIP</td>
<td>Receives control from and returns control to MVS.</td>
</tr>
<tr>
<td>DFHKESRT</td>
<td>Implements subroutine return requests.</td>
</tr>
<tr>
<td>DFHKESTX</td>
<td>Is the CICS ESTAE exit and passes control to the appropriate level of recovery routine.</td>
</tr>
<tr>
<td>DFHKESVC</td>
<td>Provides authorised services for kernel domain functions.</td>
</tr>
<tr>
<td>DFHKETA</td>
<td>Implements KEDS CREATE_TASK requests.</td>
</tr>
<tr>
<td>DFHKETCB</td>
<td>Receives control from MVS for a kernel domain TCB.</td>
</tr>
<tr>
<td>DFHKETI</td>
<td>Provides service time functions at the KETI gate.</td>
</tr>
<tr>
<td>DFHKETIX</td>
<td>Performs task CPU monitoring functions and task runaway detection.</td>
</tr>
<tr>
<td>DFHKETRI</td>
<td>Offline trace formatting routine for kernel domain trace entries.</td>
</tr>
</tbody>
</table>
**Kernel domain (KE)**

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHKETXR</td>
<td>Allows an attaching TCB to determine that a TCB (but not a specific TCB) which it attached, has terminated. This allows for the possibility of initiating a more timely detach of TCBs which have terminated normally, and to detect TCBs which have prematurely terminated.</td>
</tr>
<tr>
<td>DFHKEXM</td>
<td>Implements KEXM_FORMAT requests.</td>
</tr>
</tbody>
</table>

**Exits**

No global user exit points are provided in this domain.

**Trace**

The point IDs for the kernel domain are of the form KE xxxx; the corresponding trace levels are KE 1 and Exc.

For more information about the trace points, see the [CICS Trace Entries](#). For more information about using traces in problem determination, see the [CICS Problem Determination Guide](#).
Chapter 50. Loader domain (LD)

The loader domain is used by the domains of the CICS system to obtain access to storage-resident copies of nucleus and application programs, maps, and tables. In order to provide this, the loader domain interfaces with MVS to perform the loading of programs into the CICS dynamic storage areas (DSAs), and the scanning of the MVS link pack area (LPA).

The most common functions provided by the loader domain are:

**ACQUIRE_PROGRAM**
- used to obtain the load point and entry point addresses and length of a usable program copy, and to reserve the copy for use by the caller.

**RELEASE_PROGRAM**
- used to inform the loader domain that a specific program copy is no longer required.

**DEFINE_PROGRAM**
- used to inform the loader domain of the CICS attributes of a program.

**REFRESH_PROGRAM**
- used to request the loader domain to rescan the LPA or DFHRPL library for a new copy of a program.

The loader domain is utilized by many domains in the system, but its most common user is the program manager domain, for access to application programs. The program manager domain issues the following requests:

**ACQUIRE_PROGRAM**
- whenever a program issues a LINK, XCTL, or LOAD command to link to, transfer control to, or load another program.

**DEFINE_PROGRAM**
- as part of a request to define or autoinstall a program, mapset, or partitionset.

**RELEASE_PROGRAM**
- whenever a called program issues a RETURN command to return control to the calling program, or a program issues a RELEASE command to release a loaded program.

**REFRESH_PROGRAM**
- as part of an EXEC CICS SET PROGRAM NEWCOPY or PHASEIN request.

### Loader domain’s specific gate

Table 63 on page 758 summarizes the loader domain’s specific gate. It shows the level-1 trace point IDs of the modules providing the functions for the gate, the functions provided by the gate, and whether or not the functions are available through the exit programming interface (XPI).
Table 63. Loader domain’s specific gate

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>XPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDLD</td>
<td>LD 0001</td>
<td>ACQUIRE_PROGRAM</td>
<td>YES</td>
</tr>
<tr>
<td></td>
<td>LD 0002</td>
<td>RELEASE_PROGRAM</td>
<td>YES</td>
</tr>
<tr>
<td></td>
<td></td>
<td>REFRESH_PROGRAM</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DEFINE_PROGRAM</td>
<td>YES</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INQUIRE_PROGRAM</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DELETE_PROGRAM</td>
<td>YES</td>
</tr>
<tr>
<td></td>
<td></td>
<td>START_BROWSE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GET_NEXT_PROGRAM</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GET_NEXT_INSTANCE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>END_BROWSE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IDENTIFY_PROGRAM</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SET_OPTIONS</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INQUIRE_OPTIONS</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CATALOG_PROGRAMS</td>
<td>NO</td>
</tr>
</tbody>
</table>

**LDLD gate, ACQUIRE_PROGRAM function**

The ACQUIRE_PROGRAM function of the LDLD gate is used to obtain the entry point and load point addresses and the length of a usable copy of the named program. The program must previously have been identified to the system in a DEFINE request, either during this session or in a previous session, if the catalog is in use.

**Input parameters**

**PROGRAM_NAME**

specifies the name of the required program.

**PROGRAM_TOKEN**

is a valid program-identifying token as returned by a previous DEFINE or ACQUIRE request for the same program name.

**[SUSPEND]**

indicates whether the caller expects to receive control with an exception response if the loader encounters a shortage of virtual storage, or other transient error conditions. It can have either of these values:

YES | NO

If there is insufficient storage to satisfy the request, SUSPEND(YES) causes the caller to be suspended until the request can be satisfied, and SUSPEND(NO) causes an exception response (reason NO_STORAGE) to be returned to the caller.

**Output parameters**

**ENTRY_POINT**

is the address of the entry point of the program instance.

**[LOAD_POINT]**

is the address of the load point of the program instance.

**[PROGRAM_LENGTH]**

is the length of the program instance in bytes.

**[NEW_PROGRAM_TOKEN]**

is the identifying token that may be used on subsequent ACQUIRE or RELEASE calls for this program name.
[PROGRAM_ATTRIBUTE] reflects the program attribute from the program definition, and is used by the program manager domain to recognize RELOAD programs.

[LOCATION] determines where the program instance for which the LOAD_POINT and ENTRY_POINT have been returned resides.

[COPY_STATUS] indicates whether this request resulted in a physical load of the program into storage, and is used by the program manager domain to recognize that a COBOL program requires initialization.

[FETCH_TIME] is the time taken to load the program from the DFHRPL library. This is represented as the middle 4 bytes of a doubleword stored clock (STCK) value. If the acquired program resides in the MVS link pack area (LPA) or has already been loaded into one of the CICS dynamic storage areas (DSAs), the returned value is zero.

RESPONSE is the domain’s response to the call. It can have any of these values:
OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>LIBRARY_IO_ERROR, OS_STORAGE_SHORTAGE, ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>PROGRAM_NOT_DEFINED, PROGRAM_NOT_FOUND, NO_STORAGE</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_PROGRAM_TOKEN</td>
</tr>
</tbody>
</table>

LDLD gate, RELEASE_PROGRAM function

The RELEASE_PROGRAM function of the LDLD gate is used to inform the loader domain that use of a copy of the named program is no longer required. The use count of the specified program instance is decremented; if the use count reaches zero, and the program is eligible to be removed from memory, it is removed from memory.

Input parameters

PROGRAM_NAME specifies the name of the program to be released.

PROGRAM_TOKEN is the identifying token returned by the ACQUIRE request for this program.

ENTRY_POINT specifies the address of the entry point of the module.

Output parameters

[LOAD_POINT] is the address of the load point of the program instance.

[PROGRAM_LENGTH] is the length of the program instance in bytes.
Loader domain (LD)

[LOCATION]
determines where the program instance for which the LOAD_POINT and ENTRY_POINT have been returned resides.

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>PROGRAM_NOT_DEFINED, PROGRAM_NOT_IN_USE</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_PROGRAM_TOKEN, INVALID_ENTRY_POINT</td>
</tr>
</tbody>
</table>

LDLD gate, REFRESH_PROGRAM function

The REFRESH_PROGRAM function of the LDLD gate is used to inform the loader domain that a new version of the program has been cataloged, and that this version of the named program should be used for all future ACQUIRE requests.

Input parameters

PROGRAM_NAME
specifies the name of the program that is to have a new version used.

Output parameters

[NEW_VERSION_FOUND]
indicates whether a new version of the program has been found.

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>LIBRARY_IO_ERROR, OS_STORAGE_SHORTAGE, ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>PROGRAM_NOT_DEFINED, PROGRAM_NOT_FOUND</td>
</tr>
</tbody>
</table>

LDLD gate, DEFINE_PROGRAM function

The DEFINE_PROGRAM function of the LDLD gate is used to introduce a new program to the CICS system or to update the details of an existing program.

Input parameters

PROGRAM_NAME
specifies the name of the program whose attributes are to be set.

CATALOG_MODULE
indicates whether the program definition should be written to one of the catalogs. It can have either of these values:
YES|NO

**UPDATE**
indicates whether the loader domain should update the program definition if the loader domain already has a program definition for the program. If UPDATE(NO) is specified, and the loader domain already has a program definition for the specified program, PROGRAM_ALREADY_DEFINED is returned. It can have either of these values:
YES|NO

**[EXECUTION_KEY]**
is the execution key for the program. This is used to determine which DSA the program instance resides in. It can have either of these values:
USER|CICS

**[PROGRAM_TYPE]**
is the type of program copy to be used. It can have any of these values:
PRIVATE|SHARED|TYPE_ANY

**[PROGRAM_USAGE]**
defines whether the program is part of the CICS nucleus, or is an application program defined by the user. This determines whether the program definition is written to the local catalog or to the global catalog. It can have either of these values:
NUCLEUS|APPLICATION

**[PROGRAM_ATTRIBUTE]**
is a residency attribute to be associated with the program. It can have any of these values:
RESIDENT|REUSABLE|TRANSIENT|RELOAD

**[REQUIRED_AMODE]**
is the addressing mode required by CICS for the program. A program that does not have the required residency mode is not loaded. It can have any of these values:
24|31|AMODE_ANY

**[REQUIRED_RMODE]**
is the residency mode required by CICS for the program. A program that does not have the required mode requirements is not loaded. It can have any of these values:
24|RMODE_ANY

### Output parameters

**[NEW_PROGRAM_TOKEN]**
is an identifying token that can be used on subsequent ACQUIRE or RELEASE calls for this program name.

**RESPONSE**
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**[REASON]**
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
</tbody>
</table>
Loader domain (LD)

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>CATALOG_NOT_OPERATIONAL, CATALOG_ERROR, INVALID_PROGRAM_NAME, PROGRAM_ALREADY_DEFINED</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_MODE_COMBINATION, INVALID_TYPE_ATTRIB_COMBIN</td>
</tr>
</tbody>
</table>

**LDLD gate, INQUIRE_PROGRAM function**

The INQUIRE_PROGRAM function of the LDLD gate is used to return the details of a specific program.

**Input parameters**

PROGRAM_NAME
- specifies the name of the program whose attributes are being requested.

PROGRAM_TOKEN
- is a valid program token as returned by a previous DEFINE or ACQUIRE request, or obtained from the PPT entry, for the program.

**Output parameters**

[NEW_PROGRAM_TOKEN]
- is an identifying token that can be used on subsequent ACQUIRE or RELEASE calls for this program name.

[PROGRAM_TYPE]
- is the current program copy type.

[PROGRAM_USAGE]
- is the current usage definition.

[EXECUTION_KEY]
- is the execution key for the program.

[PROGRAM_ATTRIBUTE]
- is the current residency attribute of the program.

[SPECIFIED_AMODE]
- is the addressing mode required by CICS for the program. A program that does not have the required residency mode is not loaded. If REQUIRED_AMODE was omitted when the program was defined, AMODE_NOT_SPECIFIED is returned.

[SPECIFIED_RMODE]
- is the residency mode required by CICS for the program. A program that does not have the required residency mode is not loaded. If REQUIRED_RMODE was omitted when the program was defined, RMODE_NOT_SPECIFIED is returned.

[PROGRAM_LENGTH]
- is the length of the program in bytes. If the program has not been used, this is zero.

[PROGRAM_USE_COUNT]
- is the cumulative use count of the program.

[PROGRAM_USER_COUNT]
- is the current number of users of the program.

[LOAD_POINT]
- is the address of the load point of the last program instance created for this program name.
[ENTRY_POINT]  
is the address of the entry point of the last program instance created for  
this program name.

[LOCATION]    
indicates where the program for which the LOAD_POINT and  
ENTRY_POINT have been returned resides.

[ACCESS]  
is the type of storage that the program resides in.

RESPONSE  
is the domain’s response to the call. It can have any of these values:  
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]  
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID.  
Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>PROGRAM_NOT_DEFINED</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_PROGRAM_TOKEN</td>
</tr>
</tbody>
</table>

NEW_PROGRAM_TOKEN  
is the identifying token that may be used on subsequent INQUIRE calls for  
this program.

LDLD gate, DELETE_PROGRAM function  
The DELETE_PROGRAM function of the LDLD gate is used to remove a program  
from the CICS system. All subsequent ACQUIRE requests for the named program  
fail with a reason of PROGRAM_NOT_DEFINED. Any instance of the program in  
use at the time the DELETE is received continue to exist until a RELEASE request  
reduces the use count to zero, at which time the instance is removed from memory.

Input parameters  
PROGRAM_NAME  
specifies the name of the program to be removed.

Output parameters  
RESPONSE  
is the domain’s response to the call. It can have any of these values:  
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]  
is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>PROGRAM_NOT_DEFINED</td>
</tr>
</tbody>
</table>

LDLD gate, START_BROWSE function  
The START_BROWSE function of the LDLD gate is used to start a browse session.
Loader domain (LD)

Input parameters

[PROGRAM_NAME]
specifies the name of the program whose attributes are to be returned.

[ENTRY_POINT]
is the address of the entry point of the last program instance created for this program name.

Output parameters

BROWSE_TOKEN
is a token used to refer to this browse session on subsequent browse requests.

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
</tbody>
</table>

LDLD gate, GET_NEXT_PROGRAM function

The GET_NEXT_PROGRAM function of the LDLD gate is used to perform an INQUIRE function for the next program in the alphabetic sequence of programs in the current browse session.

Input parameters

BROWSE_TOKEN
is a valid browse token as returned by the preceding START_BROWSE request.

Output parameters

[PROGRAM_NAME]
is the name of the program whose attributes have been returned.

[PROGRAM_TYPE]
is the current program copy type.

[PROGRAM_USAGE]
is the current usage definition.

[EXECUTION_KEY]
is the execution key for the program.

[PROGRAM_ATTRIBUTE]
is the current residency attribute of the program.

[SPECIFIED AMODE]
is the current addressing mode required by CICS for the program. If REQUIRED_AMODE was omitted when the program was defined, AMODE_NOT_SPECIFIED is returned.

[SPECIFIED RMODE]
is the current residency mode required by CICS for the program. If REQUIRED_RMODE was omitted when the program was defined, RMODE_NOT_SPECIFIED is returned.
[PROGRAM_LENGTH] is the length of the program in bytes. If the program has not been used, this is zero.

[PROGRAM_USE_COUNT] is the cumulative use count of the program.

[PROGRAM_USER_COUNT] is the current number of users of the program.

[LOAD_POINT] is the address of the load point of the last program instance created for this program name.

[ENTRY_POINT] is the address of the entry point of the last program instance created for this program name.

[LOCATION] indicates where the program for which the LOAD_POINT and ENTRY_POINT have been returned resides.

[ACCESS] is the type of storage that the program resides in.

RESPONSE is the domain’s response to the call. It can have any of these values: OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>END_LIST</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_BROWSE_TOKEN</td>
</tr>
</tbody>
</table>

LDLD gate, GET_NEXT_INSTANCE function

The GET_NEXT_INSTANCE function of the LDLD gate is used to browse the current program instances in ascending load point address sequence.

Input parameters

BROWSE_TOKEN is a valid browse token as returned by the preceding START_BROWSE request.

Output parameters

[PROGRAM_NAME] is the name of the program of which this is an instance.

[PROGRAM_TYPE] is the current program copy type.

[PROGRAM_USAGE] is the current usage definition.

[EXECUTION_KEY] is the execution key for the program.
Loader domain (LD)

**[PROGRAM_ATTRIBUTE]**
is the current residency attribute of the program.

**[SPECIFIED_AMODE]**
is the current addressing mode required by CICS for the program. If
REQUIRED_AMODE was omitted when the program was defined,
AMODE_NOT_SPECIFIED is returned.

**[SPECIFIED_RMODE]**
is the current residency mode required by CICS for the program. If
REQUIRED_RMODE was omitted when the program was defined,
RMODE_NOT_SPECIFIED is returned.

**[PROGRAM_LENGTH]**
is the length of the program in bytes. If the program has not been used,
this is zero.

**[ENTRY_POINT]**
is the address of the entry point of the last program instance created for
this program name.

**[LOAD_POINT]**
is the address of the load point of the last program instance created for this
program name.

**[LOCATION]**
indicates where the program instance for which the LOAD_POINT and
ENTRY_POINT have been returned resides.

**[ACCESS]**
is the type of storage that the program resides in.

**[INSTANCE_USE_COUNT]**
is the current number of users of this instance.

**RESPONSE**
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**[REASON]**
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID.
Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>END_LIST</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_BROWSE_TOKEN</td>
</tr>
</tbody>
</table>

LDLD gate, END_BROWSE function

The END_BROWSE function of the LDLD gate is used to end a browse session.

**Input parameters**

BROWSE_TOKEN
is the token identifying this browse session.

**Output parameters**

RESPONSE
is the domain’s response to the call. It can have any of these values:
[REASON] is returned when RESPONSE is DISASTER or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_BROWSE_TOKEN</td>
</tr>
</tbody>
</table>

**LDLD gate, IDENTIFY_PROGRAM function**

The IDENTIFY_PROGRAM function of the LDLD gate is used to locate the program instance which contains the specified address.

**Input parameters**

ADDRESS

is a storage address.

**Output parameters**

[PROGRAM_NAME]

is the name of the program of which this is an instance.

[PROGRAM_TYPE]

is the current program copy type.

[PROGRAM_USAGE]

is the current usage definition.

[EXECUTION_KEY]

is the execution key for the program.

[PROGRAM_ATTRIBUTE]

is the current residency attribute of the program.

[SPECIFIED_AMODE]

is the addressing mode required by CICS for the program. A program that does not have the required residency mode is not loaded. If REQUIRED_AMODE was omitted when the program was defined, AMODE_NOT_SPECIFIED is returned.

[SPECIFIED_RMODE]

is the residency mode required by CICS for the program. A program that does not have the required residency mode is not loaded. If REQUIRED_RMODE was omitted when the program was defined, RMODE_NOT_SPECIFIED is returned.

[PROGRAM_LENGTH]

is the length of the program in bytes. If the program has not been used, this is zero.

[ENTRY_POINT]

is the address of the entry point of the last program instance created for this program name.

[LOAD_POINT]

is the address of the load point of the last program instance created for this program name.
Loader domain (LD)

**LOCATION**
indicates where the program instance for which the LOAD_POINT and ENTRY_POINT have been returned resides.

**ACCESS**
is the type of storage that the program resides in.

**INSTANCE_USE_COUNT**
is the current number of users of this instance.

**CSECT_NAME**
is the name of the CSECT within the module which contains the address. If no CSECT is available, the module name is returned.

**OFFSET_INTO_CSECT**
is the offset of the address within the CSECT. If no CSECT is available, the module name is returned.

**RESPONSE**
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**REASON**
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>INSTANCE_NOT_FOUND</td>
</tr>
</tbody>
</table>

**LDLD gate, SET_OPTIONS function**
The SET_OPTIONS function of the LDLD gate is used to set loader global options.

**Input parameters**

**LLACOPY**
indicates whether the loader is to use the MVS macro LLACOPY or BLDL to locate programs. It can have any of these values:
YES|NO|NEWCOPY

**[SHARED_PROGRAMS]**
indicates whether the loader is to use LPA-resident programs to satisfy ACQUIRE requests. It can have either of these values:
YES|NO

**[STORAGE_FACTOR]**
indicates the percentage of system free storage that may be occupied by program instances that have a zero use count.

**[PRVMOD]**
is a list of the names of modules that are not to be used from the MVS link pack area (LPA), but instead are to be loaded as private copies from the DFHRPL library.

**Output parameters**

**RESPONSE**
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED
Loader domain (LD)

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID.
Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>CATALOG_NOT_OPERATIONAL, CATALOG_ERROR</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_STORAGE_FACTOR</td>
</tr>
</tbody>
</table>

LDLD gate, INQUIRE_OPTIONS function

The INQUIRE_OPTIONS function of the LDLD gate is used to return loader global options.

**Input parameters**
None.

**Output parameters**

- **[SHARED_PROGRAMS]** indicates whether the loader is utilizing LPA-resident programs to satisfy ACQUIRE requests.
- **[STORAGE_FACTOR]** indicates the percentage of system free storage that may be occupied by program instances that have a zero use count.

**RESPONSE**
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
</tbody>
</table>

LDLD gate, CATALOG_PROGRAMS function

The CATALOG_PROGRAMS function of the LDLD gate is used at the end of CICS initialization to request the loader domain to catalog all the program definitions that need cataloging. The call is issued by the DFHSIJ1 module.

**Input parameters**
None.

**Output parameters**

**RESPONSE**
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:
Loader domain (LD)

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>CATALOG_NOT_OPERATIONAL, CATALOG_ERROR</td>
</tr>
</tbody>
</table>

Loader domain’s generic gates

Table 64 summarizes the loader domain’s generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 64. Loader domain’s generic gates

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMDM</td>
<td>LD 6001</td>
<td>PRE_INITIALIZE</td>
<td>DMDM</td>
</tr>
<tr>
<td></td>
<td>LD 6002</td>
<td>INITIALIZE_DOMAIN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>QUIESCE_DOMAIN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TERMINATE_DOMAIN</td>
<td></td>
</tr>
<tr>
<td>STST</td>
<td>LD 5001</td>
<td>COLLECT_STATISTICS</td>
<td>STST</td>
</tr>
<tr>
<td></td>
<td>LD 5002</td>
<td>COLLECT_RESOURCE_STATS</td>
<td></td>
</tr>
<tr>
<td>SMNT</td>
<td>LD 4001</td>
<td>STORAGE_NOTIFY</td>
<td>SMNT</td>
</tr>
<tr>
<td></td>
<td>LD 4002</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For descriptions of these functions and their input and output parameters, refer to the sections dealing with the corresponding generic formats:

- Functions and parameters
  - Format DMDM—["Domain manager domain’s generic formats" on page 377]
  - Format STST—["Statistics domain’s generic format" on page 1087]
  - Format SMNT—["Storage manager domain’s generic format" on page 1116]

In preinitialization processing, the LDLD gate is added, enabling programs to be loaded.

In initialization processing, on a cold start, the loader domain purges the loader program definitions (for user application programs and non-nucleus CICS modules) from the CICS global catalog. The loader domain then reads program definitions from the local catalog, and makes them available to CICS.

On a warm or emergency start, the loader domain reads program definitions from the global and local CICS catalogs, and makes them available to CICS.

For any type of start, the loader domain loads the subset of CICS nucleus programs that are defined as resident.

In quiesce and termination processing, the loader domain performs only internal routines.
## Modules

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHLDDM</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>PREInicialize</td>
</tr>
<tr>
<td></td>
<td>INITIALIZE_DOMAIN</td>
</tr>
<tr>
<td></td>
<td>QUIESCE_DOMAIN</td>
</tr>
<tr>
<td></td>
<td>TERMINTATE_DOMAIN</td>
</tr>
<tr>
<td>DFHLDDMI</td>
<td>Reinstates any program resources defined during previous runs of CICS.</td>
</tr>
<tr>
<td></td>
<td>It is called by DFHLDDM.</td>
</tr>
<tr>
<td>DFHLDDUF</td>
<td>Formats the loader domain control blocks in a CICS system.</td>
</tr>
<tr>
<td>DFHLDDL0</td>
<td>Directs the following requests to DFHLDDL1, DFHLDDL2, or DFHLDDL3, as</td>
</tr>
<tr>
<td></td>
<td>appropriate:</td>
</tr>
<tr>
<td></td>
<td>ACQUIRE PROGRAM</td>
</tr>
<tr>
<td></td>
<td>RELEASE_PROGRAM</td>
</tr>
<tr>
<td></td>
<td>REFRESH_PROGRAM</td>
</tr>
<tr>
<td>DFHLDDL1</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>ACQUIRE_PROGRAM</td>
</tr>
<tr>
<td></td>
<td>RELEASE_PROGRAM</td>
</tr>
<tr>
<td></td>
<td>REFRESH_PROGRAM</td>
</tr>
<tr>
<td>DFHLDDL2</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>DEFINE PROGRAM</td>
</tr>
<tr>
<td></td>
<td>DELETE_PROGRAM</td>
</tr>
<tr>
<td>DFHLDDL3</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>INQUIRE_PROGRAM</td>
</tr>
<tr>
<td></td>
<td>START_BROWSE</td>
</tr>
<tr>
<td></td>
<td>GET_NEXT_PROGRAM</td>
</tr>
<tr>
<td></td>
<td>GET_NEXT_INSTANCE</td>
</tr>
<tr>
<td></td>
<td>END_BROWSE</td>
</tr>
<tr>
<td></td>
<td>IDENTIFY_PROGRAM</td>
</tr>
<tr>
<td></td>
<td>SET_OPTIONS</td>
</tr>
<tr>
<td></td>
<td>INQUIRE_OPTIONS</td>
</tr>
<tr>
<td></td>
<td>CATALOG_OPTIONS</td>
</tr>
<tr>
<td>DFHLDDLNT</td>
<td>Handles the following request:</td>
</tr>
<tr>
<td></td>
<td>STORAGE_NOTIFY</td>
</tr>
<tr>
<td>DFHLDST</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>COLLECT_STATISTICS</td>
</tr>
<tr>
<td></td>
<td>COLLECT_RESOURCE_STATS</td>
</tr>
<tr>
<td>DFHLDSVC</td>
<td>Provides authorized services for loader domain functions that involve</td>
</tr>
<tr>
<td></td>
<td>MVS load facilities.</td>
</tr>
<tr>
<td>DFHLDTRI</td>
<td>Provides a trace interpretation routine for CICS dumps and traces.</td>
</tr>
</tbody>
</table>
Loader domain (LD)

Exits

No global user exit points are provided in this domain.

Trace

The point IDs for the loader domain are of the form LD xxxx; the corresponding trace levels are LD 1, LD 2, and Exc.

For more information about the trace points, see the CICS Trace Entries. For more information about using traces in problem determination, see the CICS Problem Determination Guide.
Chapter 51. Lock manager domain (LM)

The lock manager domain (also sometimes known simply as “lock manager”) provides both locking and associated queuing facilities for CICS resources. Before using these facilities, a resource must add a named lock for itself. This lock can then be requested as either exclusive or shared. If an exclusive lock is obtained, no other task may obtain the lock with that name; if a shared lock is obtained, multiple tasks may obtain that lock, and the exclusive lock with that name cannot be acquired.

Lock manager domain’s specific gate

Table 65 summarizes the lock manager domain’s specific gate. It shows the level-1 trace point IDs of the modules providing the functions for the gate, the functions provided by the gate, and whether or not the functions are available through the exit programming interface (XPI).

Table 65. Lock manager domain’s specific gate

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>XPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>LMLM</td>
<td>LM 0003</td>
<td>ADD_LOCK</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>LM 0004</td>
<td>DELETE_LOCK</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LOCK</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UNLOCK</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TEST_LOCK_OWNER</td>
<td>NO</td>
</tr>
</tbody>
</table>

LMLM gate, ADD_LOCK function

The ADD_LOCK function of the LMLM gate is used to add a named lock to LM’s state.

Input parameters

LOCK_NAME

is an 8-character name.

Output parameters

LOCK_TOKEN

is the 8-character token that uniquely identifies the lock, returned to the caller on the this call.

RESPONSE

is the domain’s response to the call. It can have any of these values:

OK| INVALID| DISASTER| KERNERROR

[REASON]

is returned when RESPONSE is DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>INSUFFICIENT_STORAGE, ABEND, LOOP</td>
</tr>
</tbody>
</table>

LMLM gate, LOCK function

The LOCK function of the LMLM gate is used to request the lock.
Lock manager domain (LM)

**Input parameters**

**LOCK_TOKEN**

is the token returned to the caller on the ADD_LOCK call.

**MODE**

defines the type of lock. It can have either of these values:

EXCLUSIVE|SHARED

**[WAIT]**

indicates whether a task is suspended (CICS) or a LOCK_BUSY is to be returned as a reason output parameter (NO). It can have either of these values:

CICS|NO

**Output parameters**

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|PURGED|INVALID|DISASTER|KERNERROR

**[REASON]**

is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>INSUFFICIENT_STORAGE, ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>LOCK_TOKEN_NOT_FOUND, DUPLICATE_LOCK_OWNER, LOCK_BUSY</td>
</tr>
</tbody>
</table>

*Note: DUPLICATE_LOCK_OWNER is returned when a resource requests a lock twice without unlocking during the same task: this is often treated in the same way as OK by the requesting resource.*

LMLM gate, UNLOCK function

The UNLOCK function of the LMLM gate is used to release the lock.

**Input parameters**

**LOCK_TOKEN**

is the token returned to the caller on the ADD_LOCK call.

**MODE**

defines the type of lock to be released. It can have either of these values:

EXCLUSIVE|SHARED

**[OWNER_TOKEN]**

defines the owner of the lock.

**Output parameters**

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR

**[REASON]**

is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
</tbody>
</table>
LMLM gate, TEST_LOCK_OWNER function

The TEST_LOCK_OWNER function of the LMLM gate is used to test the owner of a lock for self.

Input parameters

LOCK_TOKEN
is the token returned to the caller on the ADD_LOCK call.

Output parameters

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK | EXCEPTION | INVALID | DISASTER | KERNERROR

[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>LOCK_TOKEN_NOT_FOUND, NOT_LOCK_OWNER</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
</tbody>
</table>

LMLM gate, DELETE_LOCK function

The DELETE_LOCK function of the LMLM gate is used to delete the named lock from LM’s state.

Input parameters

LOCK_TOKEN
is the token returned to the caller on the ADD_LOCK call.

[OWNER_TOKEN]
defines the owner of the lock.

Output parameters

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK | EXCEPTION | INVALID | DISASTER | KERNERROR

[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>LOCK_TOKEN_NOT_FOUND, NOT_LOCK_OWNER</td>
</tr>
</tbody>
</table>
Lock manager domain (LM)

Lock manager domain’s generic gates

Table 66 summarizes the lock manager domain’s generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 66. Lock manager domain’s generic gates

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMDM</td>
<td>LM 0001</td>
<td>PRE_INITIALIZE</td>
<td>DMDM</td>
</tr>
<tr>
<td></td>
<td>LM 0002</td>
<td>INITIALIZE_DOMAIN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>QUIESCE_DOMAIN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TERMINATE_DOMAIN</td>
<td></td>
</tr>
<tr>
<td>DSNT</td>
<td>LM 0005</td>
<td>DISPATCHER_NOTIFY</td>
<td>DSNT</td>
</tr>
<tr>
<td></td>
<td>LM 0006</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For descriptions of these functions and their input and output parameters, refer to the sections dealing with the corresponding generic formats:

Functions and parameters

Format DMDM—“Domain manager domain’s generic formats” on page 377

Format DSNT—“Dispatcher domain’s generic formats” on page 324

In preinitialization processing, gates are added to make lock manager services available to other domains.

In initialization, quiesce, and termination processing, the lock manager domain performs only internal routines.

Modules

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHLMMDM</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>PRE_INITIALIZE</td>
</tr>
<tr>
<td></td>
<td>INITIALIZE_DOMAIN</td>
</tr>
<tr>
<td></td>
<td>QUIESCE_DOMAIN</td>
</tr>
<tr>
<td></td>
<td>TERMINATE_DOMAIN</td>
</tr>
<tr>
<td>DFHLMDS</td>
<td>Handles transaction manager domain MXT_CHANGE_NOTIFY requests.</td>
</tr>
<tr>
<td>DFHLMDF</td>
<td>Formats the LM domain control blocks</td>
</tr>
<tr>
<td>DFHLMMLM</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>ADD_LOCK</td>
</tr>
<tr>
<td></td>
<td>DELETE_LOCK</td>
</tr>
<tr>
<td></td>
<td>LOCK</td>
</tr>
<tr>
<td></td>
<td>TEST_LOCK_OWNER</td>
</tr>
<tr>
<td></td>
<td>UNLOCK</td>
</tr>
<tr>
<td>DFHLMTRI</td>
<td>Interprets LM domain trace entries</td>
</tr>
</tbody>
</table>
Exits

No global user exit points are provided in this domain.

Trace

The point IDs for the lock manager domain are of the form LM xxxx; the corresponding trace levels are LM 1, LM 2, and Exc.

For more information about the trace points, see the CICS Trace Entries. For more information about using traces in problem determination, see the CICS Problem Determination Guide.
Chapter 52. Log manager domain (LG)

The log manager domain (also sometimes known simply as “log manager” or “logger”) provides facilities for Recovery Manager to:

- Write records to the CICS system log
- Read records from the CICS system log
- Maintain the system log deleting obsolete records and shunting old, but still needed, records to a secondary system log.

It also provides facilities to:

- Write user journal, forward recovery and auto journals records to MVS system logger logstreams or the MVS SMF log.
- Install, discard and inquire for Journalmodel resource definitions
- Auto-install, discard, inquire and set for Journal definitions
- Connect, disconnect and define for MVS system logger logstreams
- Collect statistics for Journal and Logstream usage.

Log manager domain’s specific gates

Table 67 summarizes the log manager domain’s specific gates. It shows the level-1 trace point IDs of the modules providing the functions for the gate, the functions provided by the gate, and whether or not the functions are available through the exit programming interface (XPI).

Table 67. Log manager domain’s specific gate

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>XPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>LGGL</td>
<td>LG 0201</td>
<td>OPEN</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>LG 0202</td>
<td>WRITE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FORCE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CLOSE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WRITE_JNL</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FORCE_JNL</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UOW_TIME</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INITIALIZE</td>
<td>NO</td>
</tr>
<tr>
<td>LGJN</td>
<td>LG 0301</td>
<td>INQUIRE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>LG 0302</td>
<td>START_BROWSE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>LG 0314</td>
<td>GET_NEXT</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>LG 0325</td>
<td>END_BROWSE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SET</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DISCARD</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EXPLICIT_OPEN</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IMPLICIT_OPEN</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INITIALIZE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>STREAM_FAIL</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PROCESS_STATISTICS</td>
<td>NO</td>
</tr>
</tbody>
</table>
### Log manager domain (LG)

**Table 67. Log manager domain’s specific gate (continued)**

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>XPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>LGLD</td>
<td>LG 0401</td>
<td>INQUIRE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>LG 0402</td>
<td>START_BROWSE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>LG 0411</td>
<td>GET_NEXT</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>LG 0412</td>
<td>END_BROWSE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>LG 0415</td>
<td>MATCH</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INSTALL</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DISCARD</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INITIALIZE</td>
<td>NO</td>
</tr>
<tr>
<td>LGST</td>
<td>LG 0501</td>
<td>INQUIRE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>LG 0502</td>
<td>START_BROWSE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>LG 0514</td>
<td>GET_NEXT</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>LG 0517</td>
<td>END_BROWSE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>LG 0526</td>
<td>CONNECT</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DISCONNECT</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INITIALIZE</td>
<td>NO</td>
</tr>
<tr>
<td>LGPA</td>
<td>LG 0601</td>
<td>INQUIRE_PARAMETERS</td>
<td>YES</td>
</tr>
<tr>
<td></td>
<td>LG 0602</td>
<td>SET_PARAMETERS</td>
<td>YES</td>
</tr>
<tr>
<td>LGLB</td>
<td>LG 2001</td>
<td>CONNECT</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>LG 2002</td>
<td>DISCONNECT</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GL_WRITE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GL_FORCE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DISCONNECT_ALL</td>
<td>NO</td>
</tr>
<tr>
<td>LGCC</td>
<td>LG 2101</td>
<td>SYSINI</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>LG 2102</td>
<td>CREATE_CHAIN_TOKEN</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RELEASE_CHAIN_TOKEN</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RESTORE_CHAIN_TOKEN</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>START_BROWSE_CHAINS</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BROWSE_CHAINS_GET_NEXT</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>END_BROWSE_CHAINS</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DELETE_ALL</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SET_HISTORY</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DELETE_HISTORY</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SET_KEYPOINT_FREQUENCY</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INQUIRE_KEYPOINT_FREQUENCY</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SET_INHIBIT_DELETE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INQUIRE_INHIBIT_DELETE</td>
<td>NO</td>
</tr>
<tr>
<td>LGWF</td>
<td>LG 2201</td>
<td>WRITE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>LG 2202</td>
<td>FORCE_DATA</td>
<td>NO</td>
</tr>
<tr>
<td>LGCB</td>
<td>LG 2301</td>
<td>START_CHAIN_BROWSE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>LG 2302</td>
<td>CHAIN_BROWSE_GET_NEXT</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>END_CHAIN_BROWSE</td>
<td>NO</td>
</tr>
<tr>
<td>LGBA</td>
<td>LG 2401</td>
<td>START_BROWSE_ALL</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>LG 2402</td>
<td>BROWSE_ALL_GET_NEXT</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>END_BROWSE_ALL</td>
<td>NO</td>
</tr>
<tr>
<td>LGMV</td>
<td>LG 2501</td>
<td>MOVE_CHAIN</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>LG 2502</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LGSR</td>
<td>LG 2601</td>
<td>WRITE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>LG 2602</td>
<td>FORCE_DATA</td>
<td>NO</td>
</tr>
</tbody>
</table>

**LGBA gate, BROWSE_ALL_GET_NEXT function**

Returns the next record in the browse all object.
Input parameters
None

Output parameters

USER_TOKEN
is a user token that was passed in by RESTORE_CHAIN_TOKEN.

USER_DATA
is the address of the CICS record just read from the CICS system log.

USER_DATA_LENGTH
is the length of the CICS record just read from the chain.

RESPONSE
is the log manager domain’s response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>END_OF_DATA</td>
</tr>
</tbody>
</table>

LGBA gate, END_BROWSE_ALL function
Destroys the browse all object.

Input parameters
None

Output parameters

RESPONSE
is the log manager domain’s response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
No reason codes are defined for this call.

LGBA gate, START_BROWSE_ALL function
Creates a browse all object for the CICS system log.

Input parameters
None

Output parameters

RESPONSE
is the log manager domain’s response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
No reason codes are defined for this call.
LGCB gate, CHAIN_BROWSE_GET_NEXT function

Creates a browse object for the chain denoted by CHAIN_TOKEN.

**Input parameters**

CHAIN_TOKEN

is a chain token.

**Output parameters**

USER_DATA

is the address of the CICS record just read from the chain.

USER_DATA_LENGTH

is the length of the CICS record just read from the chain.

RESPONSE

is the log manager domain’s response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]

is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>END_OF_DATA</td>
</tr>
</tbody>
</table>

LGCB gate, END_CHAIN_BROWSE function

Destroys the chain browse object denoted by CHAIN_TOKEN.

**Input parameters**

CHAIN_TOKEN

is a chain token.

**Output parameters**

RESPONSE

is the log manager domain’s response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]

No reason codes are defined for this call.

LGCB gate, START_CHAIN_BROWSE function

Creates a browse object for the chain denoted by CHAIN_TOKEN.

**Input parameters**

CHAIN_TOKEN

is a chain token.

**Output parameters**

RESPONSE

is the log manager domain’s response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

LGCC gate, SYSINI function

Creates a primary and secondary log stream objects of type MVS that comprises the CICS system log.

Input parameters
None

Output parameters

RESPONSE
is the log manager domain’s response to the call. It can have any one of these values:
OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

LGCC gate, CREATE_CHAIN_TOKEN function

Creates a CHAIN TOKEN.

Input parameters
None

Output parameters

CHAIN_TOKEN
is a new chain token token, which can be used as input to
RELEASE_CHAIN_TOKEN, RESTORE_CHAIN_TOKEN,
START_CHAIN_BROWSE, CHAIN_BROWSE_GET_NEXT,
END_CHAIN_BROWSE, MOVE_CHAIN

RESPONSE
is the log manager domain’s response to the call. It can have any one of these values:
OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

LGCC gate, RELEASE_STORE_CHAIN_TOKEN function

Destroys the chain token in CHAIN_TOKEN

Input parameters

CHAIN_TOKEN
is a chain token that must have been created by CREATE_CHAIN_TOKEN
or RESTORE_CHAIN_TOKEN

Output parameters

RESPONSE
is the log manager domain’s response to the call. It can have any one of these values:
OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED
Log manager domain (LG)

[REASON]
No reason codes are defined for this call.

LGCC gate, RESTORE_CHAIN_TOKEN function
Creates a chain token and adds the last record (viewed as a chain element) read from the system log during a BROWSE_ALL_GET_NEXT

Input parameters
USER_TOKEN
is a user token that is returned by BROWSE_CHAINS_GET_NEXT and BROWSE_ALL_GET_NEXT.

Output parameters
CHAIN_TOKEN
is a new chain token token, which can be used as input to RELEASE_CHAIN_TOKEN, RESTORE_CHAIN_TOKEN, START_CHAIN_BROWSE, CHAIN_BROWSE_GET_NEXT, END_CHAIN_BROWSE, MOVE_CHAIN.

RESPONSE
is the log manager domain’s response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
No reason codes are defined for this call.

LGCC gate, START_BROWSE_CHAINS function
Creates a chains browse object and initializes the browse cursor position.

Input parameters
None

Output parameters
RESPONSE
is the log manager domain’s response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
No reason codes are defined for this call.

LGCC gate, BROWSE_CHAINS_GET_NEXT function
Returns the next chain token and moves the browse cursor position to the next chain.

Input parameters
None

Output parameters
CHAIN_TOKEN
is the chain token of the next chain in the chains browse list.

USER_TOKEN
is a user token that was passed in by RESTORE_CHAIN_TOKEN.
RESPONSE
is the log manager domain’s response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>END_OF_CHAINS</td>
</tr>
</tbody>
</table>

LGCC gate, END_BROWSE_CHAINS function
Destroys the browse chains object.

Input parameters
None

Output parameters
RESPONSE
is the log manager domain’s response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
No reason codes are defined for this call.

LGCC gate, DELETE_ALL function
Deletes all of the data on both log streams of the CICS system log.

Input parameters
None

Output parameters
RESPONSE
is the log manager domain’s response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
No reason codes are defined for this call.

LGCC gate, SET_HISTORY function
Evaluates and saves the current history point for both log streams of the CICS system log. The history point of a log stream is the oldest block id that CICS knows of on the log stream.

Input parameters
None

Output parameters
RESPONSE
is the log manager domain’s response to the call. It can have any one of these values:
LGCC gate, DELETE_HISTORY function
Deletes all blocks of data, for both log streams of the CICS system log, that are older than the corresponding history point saved during a call of SET_HISTORY.

Input parameters
None

Output parameters
RESPONSE
is the log manager domain’s response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
No reason codes are defined for this call.

LGCC gate, SET_KEYPOINT_FREQUENCY function
Sets the activity frequency to KEYPOINT_FREQUENCY.

Input parameters
None

Output parameters
RESPONSE
is the log manager domain’s response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>OUT_OF_RANGE</td>
</tr>
</tbody>
</table>

LGCC gate, INQUIRE_KEYPOINT_FREQUENCY function
Returns the activity keypoint frequency value in KEYPOINT_FREQUENCY.

Input parameters
None

Output parameters
KEYPOINT_FREQUENCY
is the current keypoint frequency value.

RESPONSE
is the log manager domain’s response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED
LGCC gate, **SET_INHIBIT_DELETE function**

Sets a flag to the value of INHIBIT_DELETE. This flag controls the deletion of data from the log streams of the CICS system log.

**Input parameters**

None

**Output parameters**

- **KEYPOINT_FREQUENCY**
  
is the current keypoint frequency value.

- **RESPONSE**
  
is the log manager domain’s response to the call. It can have any one of these values:
  
  OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

LGGL gate, **OPEN function**

Opens a general log and returns a log token. The log token is used by the WRITE, FORCE and CLOSE operations.

**Input parameters**

- **STREAM_NAME**
  
The 26-byte log stream name to be opened

- **JNL_NAME**
  
The 8-byte journal name to be opened

Either **STREAM_NAME** or **JNL_NAME** must be specified

- **COMPONENT**
  
Identifies the component (e.g. FC) opening this stream

- **[USER_TOKEN]**
  
A token that identifies to the calling component why this log stream was opened. It will be passed to the ERROR gate in the event that an error is detected on the log stream. For example for file control it might contain a pointer to the DSNBx

- **[ERROR_GATE]**
  
The domain gate number that the logger should call using ERROR if an error occurs accessing the log stream.

**Output parameters**

- **LOG_TOKEN**
  
The token to be used on subsequent WRITE, FORCE, CLOSE requests.

- **LOG_TYPE**
  
The associated log stream type: It can have any one of these values:

  - **MVS**  MVS logger stream
  - **SMF**  SMF logging
Log manager domain (LG)

**DUMMY**
No logging

**JNL_STREAM**
The MVS logstream name associated with the journal being opened

**RESPONSE**
is the log manager domain’s response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**[REASON]**
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>ERROR_OPENING_LOG, LOG_IS_SYSTEM_LOG, LOG_IS_DISABLED, LOG_NOT_DEFINED, LOG_HAS_FAILED, INVALID_JNL_NAME</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_PARAMETERS</td>
</tr>
</tbody>
</table>

**LGGL gate, WRITE function**
Write a record to a general log identified by a token from a previous OPEN.

**Input parameters**

**LOG_TOKEN**
The token returned by OPEN

**DATA** The address of a reusable Iliffe vector describing the items of data to be written to the log stream.

**[FORCE_NOW]**
Indicates that the caller wishes to wait until the data has been successfully written to the log stream. It can have either of these values:
YES|NO

Default is NO

**[FORCE_AT_SYNC]**
Indicates that the caller wishes the log stream to be forced when the associated transaction reaches Syncpoint. It can have either of these values:
YES|NO

Default is NO

**Note:** Force_at_Sync can be used in conjunction with FORCE_NOW. This is needed by File control for ESDS writes which have to be forced immediately but which also need the UOW structure to allow the calculation of Fuzzy backup recovery times.

**Output parameters**

**[FORCE_TOKEN]**
A token to be used on a subsequent FORCE to ensure that a specific records and any prior records have been hardened

**RESPONSE**
is the log manager domain’s response to the call. It can have any one of these values:
Log manager domain (LG)

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or INVALID Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>WRITE_ERROR, BUFFER_LENGTH_ERROR</td>
</tr>
<tr>
<td>INVALID</td>
<td>UNKNOWN_LOG_TOKEN</td>
</tr>
</tbody>
</table>

LGGL gate, FORCE function

Ensures that the previously written records have been flushed from the buffer and hardened on the chosen log stream

Input parameters

LOG_TOKEN
The token returned by OPEN

[FORCE_TOKEN]
Token returned by WRITE to indicate a specific record to be written. If omitted all records are forced.

Output parameters

RESPONSE
is the log manager domain’s response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>WRITE_ERROR</td>
</tr>
<tr>
<td>INVALID</td>
<td>UNKNOWN_LOG_TOKEN</td>
</tr>
</tbody>
</table>

LGGL gate, CLOSE function

Invalidates the LOG_TOKEN, on the last usage of a log stream disconnects from the log stream

Input parameters

LOG_TOKEN
The token returned by OPEN

Output parameters

RESPONSE
is the log manager domain’s response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:
**LGGL gate, WRITE_JNL function**

Write a record to a general log identified by a journal name

**Input parameters**

**JNL_NAME**

The 8-byte journal name to be written to

**DATA**

The address of a reusable Iliffe vector describing the items of data to be written to the log stream.

**[FORCE_NOW]**

Indicates that the caller wishes to wait until the data has been successfully written to the log stream. It can have either of these values:

YES | NO

Default is NO

**[FORCE_AT_SYNC]**

Indicates that the caller wishes the log stream to be forced when the associated transaction reaches Syncpoint. It can have either of these values:

YES | NO

Default is NO

**Note:** Force_at_Sync can be used in conjunction with FORCE_NOW. This is needed by File control for ESDS writes which have to be forced immediately but which also need the UOW structure to allow the calculation of Fuzzy backup recovery times.

**COMPONENT**

Identifies the component (e.g. TC) writing this stream

**SUSPEND**

Supported for compatibility with old EXEC interface. Causes BUFFER_FULL exception to be raised if there is no space rather than waiting for space. The task may still be suspended for many other reasons! It can have either of these values:

YES | NO

Default is YES

**Output parameters**

**[FORCE_TOKEN]**

A token to be used on a subsequent FORCE_JNL to ensure that a specific record and any prior records have been hardened

**RESPONSE**

is the log manager domain’s response to the call. It can have any one of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED
[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>WRITE_ERROR, ERROR_OPENING_LOG, LOG_IS_SYSTEM_LOG, LOG_IS_DISABLED, LOG_HAS FAILED, LOG_NOT_DEFINED, BUFFER_FULL, INVALID_JNL_NAME, BUFFER_LENGTH_ERROR</td>
</tr>
</tbody>
</table>

**LGGL gate, FORCE_JNL function**

Ensures that the previously written records have been hardened on the chosen log.

**Input parameters**

**JNL_NAME**

The 8-byte journal name to be forced

**[FORCE_TOKEN]**

Token returned by WRITE_JNL to indicate a specific record to be written. If omitted all records are forced.

**Output parameters**

**RESPONSE**

is the log manager domain’s response to the call. It can have any one of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>WRITE_ERROR, LOG_IS_NOT_ACTIVE, LOG_IS_SYSTEM_LOG, LOG_IS_DISABLED, LOG_HAS FAILED</td>
</tr>
</tbody>
</table>

**LGGL gate, UOW_TIME function**

Returns the oldest active transactions first log write time for use in calculating the recovery time for Backup while open.

Usually called by AKP processing when calculating the recovery time for non-RLS BWO files

**Input parameters**

**UOW_TIME_STAMP**

The 8-byte STCK format time of the oldest active transaction that has written log records with the FORCE_AT_SYNC option, or current time if there are no active transactions.

**Output parameters**

**RESPONSE**

is the log manager domain’s response to the call. It can have any one of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON]

No reason codes are defined for this call
Log manager domain (LG)

**LGGL gate, INITIALIZE function**
Establish subpools, locks, and anchor control blocks

Called as subroutine during domain initialization.

**Input parameters**
None

**Output parameters**
RESPONSE
is the log manager domain’s response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
No reason codes are defined for this call

**LGJN gate, INQUIRE function**
Returns information about the current state of a user journal

Also causes the stats information for a particular journal to be updated, when called as part of a FORCE_JNL request from LGGL.

**Input parameters**
JNL_NAME
The 8-byte Journal name to be inquired upon

[FORCE]
Indicates that a force of the data in the buffer has been requested.

This is used to indicate when the stats field in the journal info, which records the number of flushes, needs incrementing.

**Output parameters**
[LOG_TYPE]
The associated log stream type:
MVS  MVS logger stream
SMF  SMF logging
DUMMY
No logging

[JNL_STATUS]
The associated log stream status:

**Note:** Status will always appear as disconnected for journals that have not been used as user journals (i.e. system logs, forward recovery logs, fc auto journals) even though they may be in use

CONNECTED
Stream is currently connected

DISCONNECTED
Stream is not currently connected

DISABLED
Stream has been disabled by SPI/CEMT function

FAILED
The MVS log stream has failed
[STREAM_NAME]
The associated MVS log stream name. Blank for SMF or DUMMY

[System Log]
Whether or not the journal is a system log. It can have either of these values:
YES | NO

[STREAM_TOKEN]
The log stream token if the journal is currently connected to an MVS log stream or the logbuf token for an SMF journal.

If specified the stream shared lock will be acquired and it its the callers responsibility to free the lock when they have finished with the stream token.

RESPONSE
is the log manager domain’s response to the call. It can have any one of these values:
OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>UNKNOWN_JNL_NAME</td>
</tr>
</tbody>
</table>

**LGJN gate, START_BROWSE function**
Initialize browse token for subsequent GET_NEXT requests

**Input parameters**
None.

**Output parameters**
BROWSE_TOKEN
Token for use on subsequent GET_NEXT requests

RESPONSE
is the log manager domain’s response to the call. It can have any one of these values:
OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON]
None defined for this call

**LGJN gate, GET_NEXT function**
Return information for next Journal.

**Input parameters**
BROWSE_TOKEN
Token returned by START_BROWSE

**Output parameters**
JNL_NAME
The next 8-byte Journal name found
Log manager domain (LG)

[LOG_TYPE]
The associated log stream type:
MVS MVS logger stream
SMF SMF logging
DUMMY No logging

[JNL_STATUS]
The associated log stream status:

Note: Status will always appear as disconnected for journals that have not been used as user journals (i.e. system logs, forward recovery logs, fc auto journals) even though they may be in use
CONNECTED Stream is currently connected
DISCONNECTED Stream is not currently connected
DISABLED Stream has been disabled by SPI/CEMT function
FAILED The MVS log stream has failed

[STREAM_NAME]
The associated MVS log stream name. Blank for SMF or DUMMY

[SYSTEM_LOG]
Whether or not the journal is a system log. It can have either of these values:
YES|NO

RESPONSE is the log manager domain’s response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NO_MORE_DATA_AVAILABLE</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_BROWSE_TOKEN</td>
</tr>
</tbody>
</table>

LGJN gate, END_BROWSE function
Terminate browse and invalidate browse token

Input parameters

BROWSE_TOKEN
Token returned by START_BROWSE

Output parameters

RESPONSE is the log manager domain’s response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED
LGJN gate, SET function

Update the status of the Journal.

Will create journal if it does not currently exist (except for FLUSH)

**Input parameters**

**JNL_NAME**

The 8-byte Journal name to be updated

**JNL_STATUS**

The new status for the journal:

- **CONNECTED**
  - Stream is to be connected
- **DISCONNECTED**
  - Stream is to be disconnected
- **DISABLED**
  - Stream is to be disabled from further use
- **FLUSH**
  - The current log buffers are to be written to the log stream

**Output parameters**

**RESPONSE**

is the log manager domain’s response to the call. It can have any one of these values:

- OK
- EXCEPTION
- DISASTER
- INVALID
- KERNERROR
- PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>JNL_ALREADY_IN_REQ_STATE, JNL_IS_NOT_ACTIVE, LOG_IS_SYSTEM_LOG, SYSTEM_LOG_CONFLICT,</td>
</tr>
<tr>
<td></td>
<td>UNKNOWN_JNL_NAME, UNABLE_TO_CREATE_JNL, ERROR_OPENING_LOG, JNL_HAS_FAILED, INVALID_JNL</td>
</tr>
<tr>
<td></td>
<td>NAME, WRITE_ERROR, ...</td>
</tr>
</tbody>
</table>

LGJN gate, DISCARD function

Remove a journal from the set of known journals to clean up the catalog or to allow it to be reinstalled with a new set of attributes.

**Input parameters**

**JNL_NAME**

The 8-byte Journal name to be discarded

**Output parameters**

**RESPONSE**

is the log manager domain’s response to the call. It can have any one of these values:

- OK
- EXCEPTION
- DISASTER
- INVALID
- KERNERROR
- PURGED
Log manager domain (LG)

[REASON]

is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>LOG_IS_SYSTEM_LOG, UNKNOWN_JNL_NAME</td>
</tr>
</tbody>
</table>

LGJN gate, EXPLICIT_OPEN function

Inquire on a journal and if the journal does not already exist in the set of known journals perform the autoinstall process to define it.

The stream is explicitly opened for each call and so must eventually be explicitly closed using the LGST DISCONNECT function

Input parameters

JNL_NAME

The 8-byte Journal name to be Explicit_Opened

SYSTEM_LOG

Whether or not this journal is to be used as a system log. It can have either of these values:

YES|NO

Output parameters

[LOG_TYPE]

The associated log stream type:

MVS MVS logger stream

SMF SMF logging

DUMMY No logging

[JNL_STATUS]

The associated log stream status:

Note: Status will always appear as disconnected for journals that have not been used as user journals (i.e. system logs, forward recovery logs, fc auto journals) even though they may be in use

CONNECTED Stream is currently connected

DISCONNECTED Stream is not currently connected

DISABLED Stream has been disabled by SPI/CEMT function

FAILED The MVS log stream has failed

STREAM_TOKEN

The log stream token if the journal is currently connected to an MVS log stream or the logbuf token for an SMF journal.

[STREAM_NAME]

The associated MVS log stream name. Blank for SMF or DUMMY

[LOG_TOKEN]

The buffer manager’s log token for the log stream

[STRUCTURE_NAME]

is the 16 byte name of the coupling facility structure of the log stream.
RESPONSE
is the log manager domain’s response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>UNABLE_TO_CREATE_JNL, SYSTEM_LOG_CONFLICT, JNL_IS_DISABLED, JNL_HAS_FAILED, ERROR_OPENING_LOG, INVALID_JNL_NAME</td>
</tr>
</tbody>
</table>

**LGJN gate, IMPLICIT_OPEN function**
Inquire on a journal and if the journal does not already exist in the set of known journals perform the autoinstall process to define it. If the associated log stream has not been opened then it is opened and the stream token returned.

**Input parameters**

**JNL_NAME**
The 8-byte Journal name to be Explicit_Opened

**SYSTEM_LOG**
Whether or not this journal is to be used as a system log It can have either of these values:
YES|NO

**[FORCE]**
Indicates that a force of the data in the buffer has been requested.
This is used to indicate when the stats field in the journal info, which records the number of flushes, needs incrementing. It can have either of these values:
YES|NO

**[WRITE_BYTES]**
The number of bytes of data being written, as a 64 bit value.
This field is used to update the bytes counter in the stats information for a journal, and also indicates that the writes counter also needs incrementing.

**Output parameters**

**[LOG_TYPE]**
The associated log stream type:
MVS MVS logger stream
SMF SMF logging
DUMMY No logging

**[JNL_STATUS]**
The associated log stream status:

**Note:** Status will always appear as disconnected for journals that have not been used as user journals (i.e. system logs, forward recovery logs, fc auto journals) even though they may be in use

CONNECTEDStream is currently connected
Log manager domain (LG)

DISCONNECTED
Stream is not currently connected

DISABLED
Stream has been disabled by SPI/CEMT function

FAILED
The MVS log stream has failed

[STREAM_NAME]
The associated MVS log stream name. Blank for SMF or DUMMY

STREAM_TOKEN
The log stream token if the journal is currently connected to an MVS log stream or logbuf token for SMF.

If specified the stream shared lock will be acquired and it its the callers responsibility to free the lock when they have finished with the stream token.

RESPONSE
is the log manager domain’s response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>UNABLE_TO_CREATE_JNL, SYSTEM_LOG_CONFLICT, JNL_IS_DISABLED, JNL_HAS_FAILED, ERROR_OPENING_LOG, INVALID_JNL_NAME</td>
</tr>
</tbody>
</table>

LGJN gate, INITIALIZE function
Establish subpools, locks, and anchor control blocks

Called as subroutine during domain initialization.

Input parameters
None

Output parameters
RESPONSE
is the log manager domain’s response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
None defined for this call

LGJN gate, STREAM_FAIL function
Marks all journals that have used the failing log stream as failed, issues a message, and closes the stream connection. This ensures that all subsequent activity for the log stream is rejected until either CICS is restarted or the operator explicitly reactivates the journal

Input parameters
STREAM_TOKEN
The token of the log stream that has failed
STREAM_NAME
The name of the log stream that has failed

Output parameters
RESPONSE
is the log manager domain’s response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
None defined for this call

LGJN gate, PROCESS_STATISTICS function
Deal with the various types of requests for journal statistics using the information in the STST parameter list.

Input parameters
STATS_PARMS
The address of the STST parameter list.

Output parameters
RESPONSE
is the log manager domain’s response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>UNKNOWN_JNL_NAME, NO_JOURNALS_DEFINED</td>
</tr>
</tbody>
</table>

LGLB gate, CONNECT function
Creates a log stream object and if of type MVS, a connection is made to the log stream, denoted by its name, through the MVS logger.

Input parameters
STREAM_NAME
is the name of the log stream to be connected. Only valid if the log type is MVS.

SYSTEM_LOG
is the system log indicator, which can assume the following values:
YES The log stream being connected is part of the CICS system log.
NO The log stream being connected is general log.

LOG_TYPE
is the log stream type, which can assume the following values:
MVS A MVS logger log stream
SMF The MVS SMF log
DUMMY A dummy log

JOURNAL_NAME
is the journal name associated with the log stream on this request.
Log manager domain (LG)

[STRUCTURE_NAME]
is the 16 byte name of the coupling facility structure of the log stream.

**Output parameters**

LOGBUF_TOKEN
is the token denoting the connected log stream, which can be used as input to GL_WRITE, GL_FORCE and DISCONNECT.

RESPONSE
is the response code, possible values are:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>LOG_NOT_DEFINED, CONNECT_FAILURE</td>
</tr>
</tbody>
</table>

**LGLB gate, DISCONNECT function**

Destroys the log stream object and if it is of type MVS, disconnects from the MVS logger.

**Input parameters**

LOGBUF_TOKEN
is the token of the log stream created during a call of CONNECT.

**Output parameters**

RESPONSE
is the response code, possible values are:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>LOG_NOT_DEFINED, CONNECT_FAILURE</td>
</tr>
</tbody>
</table>

**LGLB gate, GL_WRITE function**

Writes a record to a general log denoted by LOGBUF_TOKEN.

**Input parameters**

LOGBUF_TOKEN
is the token of the log stream created during a call of CONNECT.

DATA  is the address of the data to be written.

COMPONENT identifies the original CICS component making this request.

SUSPEND
is a task suspend indicator, which can assume the following values:
YES  The task may be suspended if necessary.
NO   If there is no buffer space immediately available without suspending the current task then return an exception with a reason of BUFFER_FULL.
JOURNAL_NAME

is the journal name associated with the log stream on this request.

Output parameters

FORCE_TOKEN

is the token denoting the output buffer which includes the data of this request. This token can be used as input to GL_FORCE.

RESPONSE

is the response code, possible values are:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]

is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>BUFFER_FULL, BUFFER_LENGTH_ERROR, WRITE_FAILURE</td>
</tr>
</tbody>
</table>

LGLB gate, GL_FORCE function

Ensures that the output buffer denoted by FORCE_TOKEN for the log stream denoted by LOGBUF_TOKEN has been written to the physical media.

Input parameters

LOGBUF_TOKEN

is the token of the log stream created during a call of CONNECT.

FORCE_TOKEN

is the token denoting the output buffer containing the data written during a call of GL_WRITE. A null token denotes the current output buffer.

Output parameters

RESPONSE

is the response code, possible values are:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]

is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>WRITE_FAILURE</td>
</tr>
</tbody>
</table>

LGLB gate, DISCONNECT_ALL function

Ensures that any data in the output buffer has been written to the physical media before the stream connection is destroyed for all connected streams.

Input parameters

None.

Output parameters

RESPONSE

is the response code, possible values are:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED
Log manager domain (LG)

[REASON]
No reason codes are defined for this call

**LGLD gate, INQUIRE function**
Returns information about the current state of a JournalModel

**Input parameters**

**JOURNALMODEL_NAME**
The 8-byte JournalModel name to be inquired upon

**Output parameters**

**[JNL_TEMPLATE]**
The associated journal name template

**[LOG_TYPE]**
The associated log stream type:
- **MVS**  MVS logger stream
- **SMF**  SMF logging
- **DUMMY**  No logging

**[STREAM_PROTOTYPE]**
The associated MVS log stream name prototype

**RESPONSE**
is the log manager domain’s response to the call. It can have any one of these values:
- **OK**
- **EXCEPTION**
- **DISASTER**
- **INVALID**
- **KERNERROR**
- **PURGED**

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>UNKNOWN_JOURNALMODEL_NAME</td>
</tr>
</tbody>
</table>

**LGLD gate, START_BROWSE function**
Initialize browse token for subsequent GET_NEXT requests

**Input parameters**
None

**Output parameters**

**BROWSE_TOKEN**
Token for use on subsequent GET_NEXT requests

**RESPONSE**
is the log manager domain’s response to the call. It can have any one of these values:
- **OK**
- **EXCEPTION**
- **DISASTER**
- **INVALID**
- **KERNERROR**
- **PURGED**

[REASON]
None defined for this function.

**LGLD gate, GET_NEXT function**
Return information for next JournalModel entry
Input parameters

BROWSE_TOKEN
Token returned by START_BROWSE

Output parameters

JOURNALMODEL_NAME
The next 8-byte JournalModel name

[JNL_TEMPLATE]
The associated journal name template

[LOG_TYPE]
The associated log stream type:
MVS MVS logger stream
SMF SMF logging
DUMMY No logging

[STREAM_PROTOTYPE]
The associated MVS log stream name prototype

RESPONSE is the log manager domain’s response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NO_MORE_DATA_AVAILABLE</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_BROWSE_TOKEN</td>
</tr>
</tbody>
</table>

LGLD gate, END_BROWSE function
Terminate browse and invalidate browse token

Input parameters

BROWSE_TOKEN
Token returned by START_BROWSE

Output parameters

RESPONSE is the log manager domain’s response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>INVALID_BROWSE_TOKEN</td>
</tr>
</tbody>
</table>
Log manager domain (LG)

**LGLD gate, MATCH function**

Find JournalModel entry that best matches a journal name. Variables in the stream name prototype are resolved and the resultant stream name is returned.

**Input parameters**

JNL_NAME
The journal name to be matched

**Output parameters**

LOG_TYPE
The associated log stream type:
- MVS  MVS logger stream
- SMF  SMF logging
- DUMMY  No logging

STREAM_NAME
The MVS log stream name

RESPONSE
is the log manager domain’s response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**[REASON]**
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>INVALID_JNL_NAME</td>
</tr>
</tbody>
</table>

**LGLD gate, INSTALL function**

Create/replace JournalModel entry

**Input parameters**

JOURNALMODEL_NAME
The 8-byte JournalModel name

JNL_TEMPLATE
The associated journal name template

LOG_TYPE
The associated log stream type:
- MVS  MVS logger stream
- SMF  SMF logging
- DUMMY  No logging

STREAM_PROTOTYPE
The associated MVS log stream name prototype

**Output parameters**

RESPONSE
is the log manager domain’s response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED
[REASON]

is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>INVALID_JNL_TEMPLATE, INVALID_STREAM_PROTOTYPE</td>
</tr>
</tbody>
</table>

**LGLD gate, DISCARD function**

Remove a JournalModel from the set of defined JournalModels

**Input parameters**

**JOURNALMODEL_NAME**

The 8-byte JournalModel name to be discarded

**Output parameters**

**RESPONSE**

is the log manager domain’s response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]

is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>UNKNOWN_JOURNALMODEL_NAME</td>
</tr>
</tbody>
</table>

**LGLD gate, INITIALIZE function**

Establish subpools, locks, and anchor control blocks

Called as subroutine during domain initialization.

**Input parameters**

None

**Output parameters**

**RESPONSE**

is the log manager domain’s response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]

None defined for this function.

**LGMV gate, MOVE_CHAIN function**

Destroys the chain browse object denoted by CHAIN_TOKEN.

**Input parameters**

**CHAIN_TOKEN**

is a chain token denoting the chain to be moved.
Log manager domain (LG)

**Output parameters**

**RESPONSE**

is the log manager domain’s response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**[REASON]**

None defined for this function.

**LGPA gate, INQUIRE_PARAMETERS function**

Inquire logger domain parameters.

**Input parameters**

None

**Output parameters**

**[KEYPOINT_FREQUENCY]**

How often, in terms of physical writes to the system log, activity keypoints are initiated. A value of zero indicates that activity keypoints are not initiated.

**RESPONSE**

is the log manager domain’s response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**[REASON]**

None defined for this function.

**LGPA gate, SET_PARAMETERS function**

Set logger domain parameters.

**Input parameters**

**[KEYPOINT_FREQUENCY]**

How often, in terms of physical writes to the system log, activity keypoints should be initiated. A value of zero indicates that activity keypoints should not be initiated.

Non-zero values outside the range from 200 to 65535 inclusive are invalid and cause the OUT_OF_RANGE exception to be returned.

**Output parameters**

**RESPONSE**

is the log manager domain’s response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**[REASON]**

is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>OUT_OF_RANGE</td>
</tr>
</tbody>
</table>
LGSR gate, LOGSTREAM_STATS function
Collects, and resets if required, the log stream statistics of either the log stream denoted by LOGSTREAM_NAME or of all log streams known to the log manager.

Input parameters
[ALL]  if specified then the request is for all log streams of type MVS known to the log manager.

[LOGSTREAM_NAME]
if specified then this is a log stream name, which must be of type MVS.

STATS_BUFFER_ADDR
is the address of a buffer to put the log stream statistics record(s).

STATS_BUFFER_LENGTH
is the length of the buffer.

[RESET]
is a request qualifier that assumes the following values:
YES The log stream statistics data are to be reset after collection.
NO The log stream statistics data are not to be reset.

Output parameters
RESPONSE
is the log manager domain’s response to the call. It can have any one of these values:
OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>LOG_NOT_DEFINED</td>
</tr>
</tbody>
</table>

LGST gate, INQUIRE function
Returns information about the current state of a stream name

Input parameters
STREAM_NAME
The 26-byte stream name

Output parameters
[USE_CT]
The current number of users of the stream

[System_Log]
Whether or not this is a CICS system log It can have either of these values:
YES | NO

[FAILED]
Whether or not the stream has failed It can have either of these values:
YES | NO

RESPONSE
is the log manager domain’s response to the call. It can have any one of these values:
Log manager domain (LG)

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>UNKNOWN_STREAM_NAME</td>
</tr>
</tbody>
</table>

LGST gate, START_BROWSE function
Initialize browse token for subsequent GET_NEXT requests

**Input parameters**
None

**Output parameters**

BROWSE_TOKEN
Token for use on subsequent GET_NEXT requests

RESPONSE
is the log manager domain’s response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
None defined for this function.

LGST gate, GET_NEXT function
Return information for next stream entry

**Input parameters**

BROWSE_TOKEN
Token returned by START_BROWSE

**Output parameters**

STREAM_NAME
The 26-byte stream name

[USE_CT]
The current number of users of the stream

[System_log]
Whether or not this is a CICS system log. It can have either of these values:
YES|NO

[FAILED]
Whether or not the stream has failed. It can have either of these values:
YES|NO

RESPONSE
is the log manager domain’s response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:
**LGST gate, END_BROWSE function**

Terminate browse and invalidate browse token

**Input parameters**

**BROWSE_TOKEN**

Token returned by START_BROWSE

**Output parameters**

**RESPONSE**

is the log manager domain’s response to the call. It can have any one of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON] is returned when RESPONSE is INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>INVALID_BROWSE_TOKEN</td>
</tr>
</tbody>
</table>

**LGST gate, CONNECT function**

Connect to an MVS log stream, or increment use count on subsequent call.

**Input parameters**

**STREAM_NAME**

The 26-byte stream name

**SYSTEM_LOG**

Whether or not this is a CICS system log. It can have either of these values:

YES | NO

**Output parameters**

**STREAM_TOKEN**

A token to represent this stream

**[STRUCTURE_NAME]**

is the 16-byte name of the coupling facility structure of the log stream.

**RESPONSE**

is the log manager domain’s response to the call. It can have any one of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>SYSTEM_LOG_CONFLICT, LOG_HAS_FAILED, DEFINE_FAILURE, CONNECT_FAILURE,</td>
</tr>
</tbody>
</table>
Log manager domain (LG)

**LGST gate, DISCONNECT function**

Decrease the stream use count and disconnect from the MVS logger on last use.

**Input parameters**

STREAM_TOKEN

The token returned by CONNECT

**Output parameters**

RESPONSE

is the log manager domain’s response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**[REASON]**

None defined for this function.

**LGST gate, INITIALIZE function**

Establish subpools, locks, and anchor control blocks.

Called as subroutine during domain initialization.

**Input parameters**

None

**Output parameters**

RESPONSE

is the log manager domain’s response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**[REASON]**

None defined for this function.

**LGWF gate, FORCE_DATA function**

Ensures that the output buffer denoted by FORCE_TOKEN has been written to the physical media.

**Input parameters**

FORCE_TOKEN

is a token denoting the output buffer containing the data written during a call of GL_WRITE. A null token denotes the current output buffer.

**Output parameters**

RESPONSE

is the log manager domain’s response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**[REASON]**

is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>AKP_KICK_OFF</td>
</tr>
</tbody>
</table>
LGWF gate, WRITE function

Writes a record to the CICS system log.

Input parameters

DATA is the address of the data to be written.

CHAIN_TOKEN is a token denoting the chain that this record belongs. A chain token is created by CREATE_CHAIN_TOKEN and RESTORE_CHAIN_TOKEN.

SUSPEND is a task suspend indicator, which can assume the following values:

YES  The task may be suspended if necessary.
NO   If there is no buffer space immediately available without suspending the current task then return an exception with a reason of BUFFER_FULL.

FORCE is a request qualifier, which can assume the following values:

YES  The data of this request including any other data already in the output buffer is to be written to the physical media before returning.
NO   The data of this request need only be written to the output buffer, but may get written to the physical media.

RAISE_LENGTH is a request qualifier, which can assume the following values:

YES  If the data length is too large to fit into the output buffer then an EXCEPTION condition is returned to the caller.
NO   If the data length is too large to fit into the output buffer then the log manager terminates CICS.

Output parameters

FORCE_TOKEN is the token denoting the output buffer which includes the data of the request. This token can be used as input to GL_FORCE.

RESPONSE is the log manager domain’s response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>BUFFER_FULL, AKP_KICK_OFF, BUFFER_LENGTH_ERROR</td>
</tr>
</tbody>
</table>
Log manager domain (LG)

Log manager domain’s generic gates

Table 68 summarizes the log manager domain’s generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gate, the functions provided by the gate, and the generic format for calls to the gate.

Table 68. Log manager domain’s generic gate

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>APUE</td>
<td>LG 0101</td>
<td>SET_EXIT_STATUS</td>
<td>APUE</td>
</tr>
<tr>
<td></td>
<td>LG 0102</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DMDM</td>
<td>LG 0101</td>
<td>INITIALISE_DOMAIN</td>
<td>DMDM</td>
</tr>
<tr>
<td></td>
<td>LG 0102</td>
<td>QUIESCE_DOMAIN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TERMINATE_DOMAIN</td>
<td></td>
</tr>
<tr>
<td>STST</td>
<td>LG 0701</td>
<td>COLLECT_STATISTICS</td>
<td>STST</td>
</tr>
<tr>
<td></td>
<td>LG 0702</td>
<td>COLLECTRESOURCE_STATISTICS</td>
<td></td>
</tr>
</tbody>
</table>

You can find descriptions of these functions and their input and output parameters, in the section dealing with the corresponding generic format, in “Domain manager domain’s generic formats” on page 377.

In Initialization processing, the log manager domain retrieves Journal and Journalmodel information from the catalog and initializes the system log except on a cold start when system log initialization occurs after group list install has completed.

In Quiesce processing, the log manager disconnects from MVS log streams after all transactions have completed.

Log manager domain’s call back gates

Table 68 summarizes the log manager domain’s call back gates. It shows the level-1 trace point IDs of the modules providing the functions for the gate, the functions provided by the gate, and the format for calls to the gate.

Table 69. Log manager domain’s call back gate

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>RMRO</td>
<td>LG 0201</td>
<td>PERFORM_COMMIT</td>
<td>RMRO</td>
</tr>
<tr>
<td></td>
<td>LG 0202</td>
<td>PERFORM_PREPARE</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>START_BACKOUT</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>DELIVER_BACKOUT_DATA</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>END_BACKOUT</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PERFORM_SHUNT</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PERFORM_UNSHUNT</td>
<td></td>
</tr>
</tbody>
</table>

You can find descriptions of these functions and their input and output parameters, in the section dealing with the recovery manager formats, in “Chapter 68. Recovery Manager Domain (RM)” on page 919.

For PERFORM_PREPARE, PERFORM_COMMIT, END_BACKOUT the log manager forces any log buffers written using the FORCE_AT_SYNCH option of the LGGL WRITE gate to the MVS system logger. For the other RMRO gate functions the log manager does nothing.
Log manager domain’s call back format

Table 70 describes the call back format owned by the log manager domain and shows the function performed on the calls.

Table 70. Call back format owned by the log manager domain

<table>
<thead>
<tr>
<th>Format</th>
<th>Calling module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>LGGL</td>
<td>DFHLGGL</td>
<td>ERROR</td>
</tr>
</tbody>
</table>

In the descriptions of the formats that follow, the “input” parameters are input not to log manager domain, but to the domain being called by the log manager. Similarly, the “output” parameters are output by the domain that was called by log manager domain, in response to the call.

LGGL gate, ERROR function

This is a back-to-front or outbound function. The logger will call the domain that issued OPEN, using the gate number specified in ERROR_GATE, when a long term error condition is detected on the opened log stream.

The called domain should take any recovery action needed and close the log stream (if appropriate).

Called by the logger during log stream error processing.

Note: An error call back could occur while an Open or Close request for the associated log-token is still in progress.

Input parameters

ERROR_TYPE
Indicates the severity of the error: It can have either of these values:
LONG_TERM|RECOVERED

STREAM_NAME
The 26-byte name of the failing log stream name

[JNL_NAME]
The 8-byte journal name if the open was by journal name

COMPONENT
The 2-byte component id supplied on OPEN

USER_TOKEN
The 8-byte token supplied on OPEN, this allows the opening domain to determine what resource (eg DSNB) this open is associated with.

LOG_TOKEN
The token returned by OPEN

Output parameters

RESPONSE
is the log manager domain’s response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
No reason codes are defined for this call
### Log manager domain (LG)

#### Modules

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHLGDM</td>
<td>Log manager domain initialization and termination. Also handles exit activation for XLGSTRM and XRSINDI. Handles the DMDM and APUE gate functions</td>
</tr>
<tr>
<td>DFHLGDUF</td>
<td>A routine to format system dump information</td>
</tr>
<tr>
<td>DFHLGGGL</td>
<td>Handles the LGGL and RMRO gate functions</td>
</tr>
<tr>
<td>DFHLGHB</td>
<td>Assesses the availability of the MVS system logger</td>
</tr>
<tr>
<td>DFHLGICV</td>
<td>Log record conversion for SSI exit</td>
</tr>
<tr>
<td>DFHLGI GT</td>
<td>Log record get routine for SSI exit</td>
</tr>
<tr>
<td>DFHLGILA</td>
<td>Lexical analysis for SSI exit</td>
</tr>
<tr>
<td>DFHLGIMS</td>
<td>Message composer for SSI exit</td>
</tr>
<tr>
<td>DFHLGIPA</td>
<td>Parser for SSI exit</td>
</tr>
<tr>
<td>DFHLGI PI</td>
<td>Parse interface for SSI exit</td>
</tr>
<tr>
<td>DFHLGI SM</td>
<td>Parse message exit for SSI exit</td>
</tr>
<tr>
<td>DFHLGJN</td>
<td>Handles the LGJN gate functions</td>
</tr>
<tr>
<td>DFHLGLD</td>
<td>Handles the LGLD gate functions</td>
</tr>
<tr>
<td>DFHLGPA</td>
<td>Handles the LGPA gate functions</td>
</tr>
<tr>
<td>DFHLGSC</td>
<td>Handles the STST gate functions</td>
</tr>
<tr>
<td>DFHLGST</td>
<td>Handles the LGST gate functions</td>
</tr>
<tr>
<td>DFHLGSSI</td>
<td>Handles the batch QSAM access to CICS logstreams via the DD SUBSYS=(LOGR...) SSI interface</td>
</tr>
<tr>
<td>DFHLGTRI</td>
<td>A routine to format trace points</td>
</tr>
<tr>
<td>DFHL2DM</td>
<td>Initializes the ‘L2’ part of the Log Manager Domain</td>
</tr>
<tr>
<td>DFHL2 TRI</td>
<td>A routine to format the ‘L2’ trace points</td>
</tr>
<tr>
<td>DFHL2LB</td>
<td>Handles the LGLB gate functions</td>
</tr>
<tr>
<td>DFHL2SR</td>
<td>Handles the LGSR gate functions</td>
</tr>
<tr>
<td>DFHL2WF</td>
<td>Handles the LGWF gate functions</td>
</tr>
<tr>
<td>DFHL2CC</td>
<td>Handles the LGCC gate functions</td>
</tr>
<tr>
<td>DFHL2CB</td>
<td>Handles the LGCB gate functions</td>
</tr>
<tr>
<td>DFHL2BA</td>
<td>Handles the LGBA gate functions</td>
</tr>
<tr>
<td>DFHL2MV</td>
<td>Handles the LGMV gate functions</td>
</tr>
<tr>
<td>DFHL2BL1</td>
<td>Initializes the Block class data</td>
</tr>
<tr>
<td>DFHL2BL2</td>
<td>Retrieves the current block on the CICS system log</td>
</tr>
<tr>
<td>DFHL2BS1</td>
<td>Initializes the BrowseableStream class data</td>
</tr>
<tr>
<td>DFHL2BS2</td>
<td>Creates a BrowseableStream class instance</td>
</tr>
<tr>
<td>DFHL2BS3</td>
<td>Destroys a BrowseableStream class instance</td>
</tr>
<tr>
<td>DFHL2BS4</td>
<td>Destroys all BrowseableStream class instance</td>
</tr>
<tr>
<td>DFHL2CH1</td>
<td>Initializes the Chain class data</td>
</tr>
<tr>
<td>DFHL2CH2</td>
<td>Creates a Chain class instance</td>
</tr>
<tr>
<td>DFHL2CH3</td>
<td>Handles start chain browse</td>
</tr>
</tbody>
</table>
Module | Function
---|---
DFHL2CH4 | Handles chain browse get next
DFHL2CH5 | Handles end chain browse
DFHL2CHA | Handles start browse all
DFHL2CHN | Handles browse all get next
DFHL2CHL | Handles end browse all
DFHL2CHH | Handles start browse chains
DFHL2CHG | Handles browse chains get next
DFHL2CHI | Handles end browse chains
DFHL2CHR | Handles chain restore
DFHL2CHS | handles set history point
DFHL2CHE | Handles delete at history point
DFHL2CHM | Handles move chain
DFHL2HS2 | Handles the log stream connect request to the MVS logger
DFHL2HS3 | Handles the log stream disconnect request to the MVS logger
DFHL2HS4 | Handles the log stream delete all request to the MVS logger
DFHL2HS5 | Handles the log stream delete history request to the MVS logger
DFHL2HS6 | Handles the log stream start browse block request to the MVS logger
DFHL2HS7 | Handles the log stream start browse cursor request to the MVS logger
DFHL2HS8 | Handles the log stream read browse cursor request to the MVS logger
DFHL2HS9 | Handles the log stream end browse cursor request to the MVS logger
DFHL2HSG | Handles the log stream read browse block request to the MVS logger
DFHL2HSJ | Handles the log stream end browse block request to the MVS logger
DFHL2OFI | Initializes the ObjectFactory instance data
DFHL2SL1 | Initializes the SystemLog class data
DFHL2SLN | Handles system log log stream open request
DFHL2SLE | Handles system log log stream failure notification
DFHL2SR1 | Initializes the Stream class data
DFHL2SR2 | Creates a Stream class instance
DFHL2SR3 | Destroys a Stream class instance
DFHL2SR4 | Collect and resets Stream statistics
DFHL2SR5 | Destroys all Stream class instances
DFHL2VPX | Initializes the VariablePool class data

Exits

Two global user exit points are provided in this domain.

**XLGSTRM**

This exit is called prior to defining a new log stream to the MVS system logger
Log manager domain (LG)

XRSINDI

This exit is called when a Journal or Journalmodel is installed or discarded. It is also called when CICS connects or disconnects an MVS system logger logstream.

See CICS Customization Guide for further information.

Trace

The point IDs for the log manager domain are of the form LG xxxx; the corresponding trace levels are LG 1, LG 2, and Exc.

For more information about the trace points, see the CICS Trace Entries. For more information about using traces in problem determination, see the CICS Problem Determination Guide.
Chapter 53. Master terminal program

The master terminal program enables dynamic control of the system. Using this function an operator can change the values of parameters used by CICS, alter the status of system resources, terminate tasks, and shut down the CICS system.

Design overview

The master terminal program is invoked by the CEMT transaction. The user enters a command to INQUIRE about or SET the status of a set of resources, and the command outputs a display that shows the resultant status of the resources. For a CEMT SET command, this display can be overtyped to alter the status of most of the resources displayed.

Commands are analyzed using the same techniques as the command interpreter described in Chapter 18. Command interpreter on page 275. A language table is used to define the syntax of commands and the contents of parameter lists which must be passed to DFHEIP to allow execution. In effect, each CEMT command results in the execution of a series of EXEC CICS INQUIRE and SET commands.

The master terminal program is also used by the CEST and CEOT transactions, which provide subsets of the functions available with CEMT. CEST is for supervisory operators and allows access to a limited set of resources. CEOT only allows changes to the status of the operator’s own terminal.

Modules

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHEMTP</td>
<td>Invoked by CEMT. Checks that the terminal is suitable. Obtains and initializes working storage. Loads the language table DFHEITMT. Links to DFHEMTD.</td>
</tr>
<tr>
<td>DFHEOTP</td>
<td>Same as DFHEMTP but invoked by CEOT and loads the language table DFHEITOT.</td>
</tr>
<tr>
<td>DFHESTP</td>
<td>Same as DFHEMTP but invoked by CEST and loads the language table DFHEITST.</td>
</tr>
<tr>
<td>DFHEMTD</td>
<td>Receives data from the terminal and sends back a display. Analyzes commands and overtypes. Constructs parameter lists for DFHEIP, which it calls. Deals with PF keys.</td>
</tr>
<tr>
<td>DFHEITMT</td>
<td>Command language table for CEMT.</td>
</tr>
<tr>
<td>DFHEITOT</td>
<td>Command language table for CEOT.</td>
</tr>
<tr>
<td>DFHEITST</td>
<td>Command language table for CEST.</td>
</tr>
</tbody>
</table>

Exits

No global user exit points are provided for this function.

Trace

No trace points are provided for this function.
Chapter 54. Message domain (ME)

The message domain acts as a repository for CICS messages, and handles the sending of messages to transient data destinations or to the console. It also provides an interface for returning the text of a message to the caller.

Message domain’s specific gates

Table 71 summarizes the message domain’s specific gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and whether or not the functions are available through the exit programming interface (XPI).

Table 71. Message domain’s specific gates

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>XPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEBM</td>
<td>None</td>
<td>INQUIRE_MESSAGE_DEFINITION</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INQUIRE_MESSAGE_LENGTH</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RETRIEVE_MESSAGE</td>
<td>NO</td>
</tr>
<tr>
<td>MEME</td>
<td>ME 0301</td>
<td>CONVERSE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>ME 0302</td>
<td>INQUIRE_MESSAGE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INQUIRE_MESSAGE_LENGTH</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RETRIEVE_MESSAGE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SEND_MESSAGE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VALIDATE_LANGUAGE_CODE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VALIDATE_LANGUAGE_SUFFIX</td>
<td>NO</td>
</tr>
<tr>
<td>MESR</td>
<td>ME 0201</td>
<td>SET_MESSAGE_OPTIONS</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>ME 0202</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

MEBM gate, RETRIEVE_MESSAGE function

The RETRIEVE_MESSAGE function of the MEBM gate is used to retrieve the message text and build the message into a buffer.

Input parameters

MESSAGE_TABLE
is a table containing all the message definitions for the message domain.

[COMPONENT_ID]
is the component identifier for the message.

MESSAGE_NUMBER
is the numeric message identifier.

MESSAGE_BUFFER
is the buffer to receive the message text.

[INSERT1] through [INSERT10]
are user-supplied inserts, if these are required by the message definition.

[SYMPTOM_BUFFER]
is the buffer to receive a symptom string for the message.

[SUPPRESS_SRBUILD]
indicates whether or not a symptom record build is suppressed. It can have either of these values:
Message domain (ME)

YES|NO

[MODULE_NAME]
is the name of the module in error, supplied as data for the symptom string.

[MODULE_PTF]
is the PTF level of the module in error, supplied as data for the symptom string.

[UPPERCASE]
determines whether or not messages should be converted to uppercase. It can have either of these values:
YES|NO

Output parameters
RESPONSE
is the domain’s response to the call. It can have either of these values:
OK|EXCEPTION

[REASON]
is returned when RESPONSE is EXCEPTION. It has this value:
MESSAGE_CANNOT_BE_PRODUCED

MEBM gate, INQUIRE_MESSAGE_LENGTH function
The INQUIRE_MESSAGE_LENGTH function of the MEBM gate is used to find the length of the message in order to obtain the appropriate sized buffer to retrieve the message.

Input parameters
MESSAGE_TABLE
is a table containing all the message definitions for messages output by the message domain.

[COMPONENT_ID]
is the component identifier for the message.

MESSAGE_NUMBER
is the numeric message identifier.

[INSERT1] through [INSERT10]
are user-supplied inserts, if these are required by the message definition.

Output parameters
MESSAGE_LENGTH
is the length of the message being inquired on.

RESPONSE
is the domain’s response to the call. It can have either of these values:
OK|EXCEPTION

[REASON]
is returned when RESPONSE is EXCEPTION. It has this value:
MESSAGE_CANNOT_BE_FOUND

MEBM gate, INQUIRE_MESSAGE_DEFINITION function
The INQUIRE_MESSAGE_DEFINITION function of the MEBM gate is used to return the action and severity codes of a message.
Input parameters

MESSAGE_TABLE
is a table containing all the message definitions for the message domain.

[COMPONENT_ID]
is the component identifier for the message.

MESSAGE_NUMBER
is the numeric message identifier.

Output parameters

SEVERITY_CODE
is the severity of the message.

ACTION_CODE
is the action code for the message.

RESPONSE
is the domain’s response to the call. It can have either of these values:

OK | EXCEPTION

[REASON]
is returned when RESPONSE is EXCEPTION. It has this value:
MESSAGECANNOTBE_FOUND

MEME gate, SEND_MESSAGE function

The SEND_MESSAGE function of the MEME gate is used to send a message to one or more destinations.

Input parameters

[COMPONENT_ID]
is the component identifier for the message.

MESSAGE_NUMBER
is the numeric message identifier.

[PRODUCT]
is an optional product identifier.

[MSGTABLE]
indicates that the feature message table is to be used.

[System_DumpCode]
is the dump code to be used when the message domain requests a dump on behalf of its caller.

[TERMINATE_CICS]
specifies whether the caller requests CICS to be terminated.

[INSERT1] through [INSERT10]
are user-supplied inserts, if these are required by the message definition.

[TRANID]
is the transaction identifier to be used to override the tranid obtained by the message domain.

[TERMID]
is the terminal identifier to be used to override the termid obtained by the message domain.
Message domain (ME)

[NETNAME]
is the network name to be used to override the netname obtained by the message domain.

Output parameters

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, INVALID, or PURGED. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>INVALID_MODULE_PTR, INVALID_TEMPLATE, ABEND, INSUFFICIENT_STORAGE</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_COMPONENT_TYPE, INVALID_DBCS_FORMAT, INVALID_DESTINATION, INVALID_FUNCTION, INVALID_INSERT, INVALID_MEFO_RESPONSE, MESSAGE_NOT_FOUND, MESSAGE_SET_NOT_FOUND, MISSING_INSERT, OPT_INSERT_NOT_FOUND, RETRY_MSG_LOCATE</td>
</tr>
<tr>
<td>PURGED</td>
<td>TDQ_PURGED</td>
</tr>
</tbody>
</table>

MEME gate, CONVERSE function

The CONVERSE function of the MEME gate is used to send a message and receive a reply.

Input parameters

[COMPONENT_ID]
is the component identifier for the message.

MESSAGE_NUMBER
is the numeric message identifier.

[PRODUCT]
is an optional product identifier.

[INSERT1] through [INSERT10]
are user-supplied inserts, if these are required by the message definition.

[TRANID]
is the transaction identifier to be used to override the tranid obtained by the message domain.

[TERMID]
is the terminal identifier to be used to override the termid obtained by the message domain.

[NETNAME]
is the network name to be used to override the netname obtained by the message domain.

[REPLY_BUFFER]
is the buffer into which the text reply is to be returned.

REPLY_FORMAT (VALUE|TEXT_OR_VALUE|TEXT)
indicates the format of the reply. It can be one of these formats:
VALUE|TEXT_OR_VALUE|TEXT
Output parameters

[REPLY_INDEX] is the number of the template reply option that matches the user’s reply text.

RESPONSE is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>INVALID_MODULE_PTR, INVALID_TEMPLATE, MAX_REPLIES_EXCEEDED, ABEND, INSUFFICIENT_STORAGE</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>REPLY_BUFFER_TOO_SMALL</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_COMPONENT_TYPE, INVALID_DESTINATION, INVALID_FUNCTION, INVALID_INSERT, INVALID_REPLY_BUFFER, MESSAGE_NOT_FOUND, MESSAGE_SET_NOT_FOUND, MISSING_INSERT, OPT_INSERT_NOT_FOUND, REPLY_BUFFER_REQUIRED, REPLY_INDEX_REQUIRED, RETRY_MSG_LOCATE</td>
</tr>
</tbody>
</table>

MEME gate, RETRIEVE_MESSAGE function

The RETRIEVE_MESSAGE function of the MEME gate is used to retrieve a message text.

Input parameters

[COMPONENT_ID] is the component identifier for the message.

MESSAGE_NUMBER is the numeric message identifier.

MESSAGE_BUFFER is the buffer to receive the message text.

[PRODUCT] is an optional product identifier.

[MSGTABLE] indicates that the feature message table is to be used.

[LANGUAGE] is an optional language code.

[INSERT1] through [INSERT10] are user-supplied inserts, if these are required by the message definition.

[TRANID] is the transaction identifier to be used to override the tranid obtained by the message domain.

[TERMD] is the terminal identifier to be used to override the termid obtained by the message domain.

[NETNAME] is the network name to be used to override the netname obtained by the message domain.
Message domain (ME)

**Output parameters**

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

**[REASON]**

is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, INSUFFICIENT_STORAGE, INVALID_MODULE_PTR, INVALID_TEMPLATE</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>MSG_BUFFER_TOO_SMALL, REPLY_BUFFER_TOO_SMALL</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_COMPONENT_TYPE, INVALID_FUNCTION, INVALID_INSERT,</td>
</tr>
<tr>
<td></td>
<td>INVALID_MESSAGE_BUFFER, MESSAGE_NOT_FOUND, MESSAGE_SET_NOT_FOUND,</td>
</tr>
<tr>
<td></td>
<td>MISSING_INSERT, OPT_INSERT_NOT_FOUND, RETRY_MSG_LOCATE</td>
</tr>
</tbody>
</table>

**MEME gate, INQUIRE_MESSAGE_LENGTH function**

The INQUIRE_MESSAGE_LENGTH function of the MEME gate is used to find the length of the message in order to obtain the appropriate size buffer to retrieve the message.

**Input parameters**

**[COMPONENT_ID]**

is the component identifier for the message.

**MESSAGE_NUMBER**

is the numeric message identifier.

**[PRODUCT]**

is an optional product identifier.

**[MSGTABLE]**

indicates that the feature message table is to be used.

**[LANGUAGE]**

is an optional language code.

**[INSERT1] through [INSERT10]**

are user-supplied inserts, if these are required by the message definition.

**Output parameters**

**MESSAGE_LENGTH**

is the length of the message being inquired on.

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

**[REASON]**

is returned when RESPONSE is DISASTER or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, INSUFFICIENT_STORAGE, INVALID_MODULE_PTR, INVALID_TEMPLATE</td>
</tr>
</tbody>
</table>
MEME gate, VALIDATE_LANGUAGE_CODE function

The VALIDATE_LANGUAGE_CODE function of the MEME gate is used to determine whether a specific three-letter IBM standard national language code is valid. If it is valid, this function returns the equivalent one-character CICS language suffix. The IBM standard three-character codes, and their corresponding one-character CICS language suffices, are listed in Table 72 on page 826.

Input parameters

LANGUAGE_CODE

is the three-character national language code to be validated.

Output parameters

[LANGUAGE_SUFFIX]

is the one-character CICS language suffix that corresponds to the input LANGUAGE_CODE.

RESPONSE

is the domain’s response to the call. It can have any of these values:

OK | DISASTER | EXCEPTION | INVALID

[REASON]

is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID.

Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>INVALID_COMPONENT_TYPE, INVALID_FUNCTION, INVALID_INSERT, MESSAGE_NOT_FOUND, MESSAGE_SET_NOT_FOUND, MISSING_INSERT, OPT_INSERT_NOT_FOUND, RETRY_MSG_LOCATE</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>LANGUAGE_CODE_INVALID, LANGUAGE_NOT_SUPPORTED</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION</td>
</tr>
</tbody>
</table>
### Table 72. Languages and their codes

<table>
<thead>
<tr>
<th>NATLANG code</th>
<th>NLS code</th>
<th>Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>ENG</td>
<td>Alternative English</td>
</tr>
<tr>
<td>Q</td>
<td>ARA</td>
<td>Arabic</td>
</tr>
<tr>
<td>1</td>
<td>BEL</td>
<td>Byelorussian</td>
</tr>
<tr>
<td>L</td>
<td>BGR</td>
<td>Bulgarian</td>
</tr>
<tr>
<td>B</td>
<td>PTB</td>
<td>Brazilian Portuguese</td>
</tr>
<tr>
<td>T</td>
<td>DBCS</td>
<td>Traditional Chinese</td>
</tr>
<tr>
<td>C</td>
<td>DBCS</td>
<td>Simplified Chinese</td>
</tr>
<tr>
<td>2</td>
<td>CSY</td>
<td>Czech</td>
</tr>
<tr>
<td>D</td>
<td>DAN</td>
<td>Danish</td>
</tr>
<tr>
<td>G</td>
<td>DEU</td>
<td>German</td>
</tr>
<tr>
<td>O</td>
<td>ELL</td>
<td>Greek</td>
</tr>
<tr>
<td>S</td>
<td>ESP</td>
<td>Spanish</td>
</tr>
<tr>
<td>W</td>
<td>FIN</td>
<td>Finnish</td>
</tr>
<tr>
<td>F</td>
<td>FRA</td>
<td>French</td>
</tr>
<tr>
<td>X</td>
<td>HEB</td>
<td>Hebrew</td>
</tr>
<tr>
<td>3</td>
<td>HRV</td>
<td>Croatian</td>
</tr>
<tr>
<td>4</td>
<td>HUN</td>
<td>Hungarian</td>
</tr>
<tr>
<td>J</td>
<td>ISL</td>
<td>Icelandic</td>
</tr>
<tr>
<td>I</td>
<td>ITA</td>
<td>Italian</td>
</tr>
<tr>
<td>H</td>
<td>DBCS</td>
<td>Korean</td>
</tr>
<tr>
<td>M</td>
<td>MKD</td>
<td>Macedonian</td>
</tr>
<tr>
<td>9</td>
<td>NLD</td>
<td>Dutch</td>
</tr>
<tr>
<td>N</td>
<td>NOR</td>
<td>Norwegian</td>
</tr>
<tr>
<td>5</td>
<td>PLK</td>
<td>Polish</td>
</tr>
<tr>
<td>P</td>
<td>PTG</td>
<td>Portuguese</td>
</tr>
<tr>
<td>6</td>
<td>ROM</td>
<td>Romanian</td>
</tr>
<tr>
<td>R</td>
<td>RUS</td>
<td>Russian</td>
</tr>
<tr>
<td>Y</td>
<td>SHC</td>
<td>Serbo-Croatian (Cyrillic)</td>
</tr>
<tr>
<td>7</td>
<td>SHL</td>
<td>Serbo-Croatian (Latin)</td>
</tr>
<tr>
<td>V</td>
<td>SVE</td>
<td>Swedish</td>
</tr>
<tr>
<td>Z</td>
<td>THA</td>
<td>Thai</td>
</tr>
<tr>
<td>8</td>
<td>TRK</td>
<td>Turkish</td>
</tr>
<tr>
<td>U</td>
<td>UKR</td>
<td>Ukrainian</td>
</tr>
</tbody>
</table>

**Notes:**

1. **DBCS** denotes Double-Byte Character Set languages.
2. A for alternative English. Code letter A means “alternative English” to distinguish your edited English message tables from the default US English message tables supplied by CICS. The default US English tables are designated by the language code letter E.
3. The NATLANG code for the selected language is used as the suffix of your edited message data sets that you can create using the message editing utility. For more information about the message editing utility, see [CICS Operations and Utilities Guide](https://www.ibm.com/docs/en/cics?topic=validate-language-suffix).

---

**MEME gate, VALIDATE_LANGUAGE SUFFIX function**

The VALIDATE_LANGUAGE SUFFIX function of the MEME gate is used to determine whether a specific one-character CICS language suffix is valid. If it is valid, this function returns the equivalent three-character IBM standard national language code. The IBM standard three-character codes, and their corresponding one-character CICS language suffixes, are listed in **Table 72**.
Input parameters
LANGUAGE_SUFFIX
is the one-character CICS language code to be validated.

Output parameters
[LANGUAGE_CODE]
is the three-character CICS language suffix that corresponds to the input LANGUAGE_SUFFIX.

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|DISASTER|EXCEPTION|INVALID

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>LANGUAGE_NOT_SUPPORTED, LANGUAGE_SUFFIX_INVALID</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

MEME gate, INQUIRE_MESSAGE function
The INQUIRE_MESSAGE function of the MEME gate is used to find the system default language as a one-character CICS language suffix and a three-character IBM standard national language code.

Input parameters
None.

Output parameters
DEFAULT_LANGUAGE_CODE
is the three-character code for the default language.

DEFAULT_LANGUAGE_SUFFIX
is the one-character suffix for the default language.

RESPONSE
is the domain’s response to the call. It can have either of these values:
OK|DISASTER|INVALID

[REASON]
is returned when RESPONSE is DISASTER or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

MESR gate, SET_MESSAGE_OPTIONS function
The SET_MESSAGE_OPTIONS function of the MESR gate is used to set the various message options specified by the system initialization parameters MSGCASE, MSGLVL, and NATLANG.
Message domain (ME)

Input parameters

[Languages_used] is a list of the languages used in the system.

[message_level] can be 0 or 1. 0 means that information messages do not appear (are suppressed) at the console.

[message_case] is either mixed-case messages, or upper for messages to be folded to uppercase.

Output parameters

[response] is the domain’s response to the call. It can have any of these values:

OK|INVALID|KERNERROR|PURGED

[reason] is returned when [response] is INVALID. It has this value:

INVALID_FUNCTION

Message domain’s generic gate

[Table 73] summarizes the message domain’s generic gate. It shows the level-1 trace point IDs of the modules providing the functions for the gate, the functions provided by the gate, and the generic format for calls to the gate.

Table 73. Message domain’s generic gate

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMDM</td>
<td>ME 0101</td>
<td>PRE_INITIALIZIZE</td>
<td>DMDM</td>
</tr>
<tr>
<td></td>
<td>ME 0102</td>
<td>INITIALIZE_DOMAIN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>QUIESCE_DOMAIN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TERMINATE_DOMAIN</td>
<td></td>
</tr>
</tbody>
</table>

You can find descriptions of these functions and their input and output parameters, in the section dealing with the corresponding generic formats, in "Domain manager domain’s generic formats" on page 377.

In preinitialization processing, the message domain sets the following message options:

• The national languages to be supported during this CICS run
• The message level for initialization messages
• The message case.

For a cold start, the information comes from the system initialization parameters; for any other type of start, the information comes from the local catalog, but is then modified by any relevant system initialization parameters.

The message domain does no quiesce processing or termination processing.

Modules

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHCMAC</td>
<td>Displays messages and codes online for the CMAC transaction</td>
</tr>
</tbody>
</table>
Module | Function
--- | ---
DFHMEBM | Is executed in an offline environment, and is provided for use by batch utility programs
DFHMEBU | Builds a message into a buffer, and also builds a symptom string when required
DFHMEDM | Performs the necessary domain manager functions; that is, preinitialize, initialize, quiesce, and terminate for the message domain
DFHMEDUF | ME domain offline dump formatting routine
DFHMEFO | Formats a long message into lines of specified length
DFHMEIN | Provides all the data required to build a message
DFHMEME | Handles the following functions:
SEND_MESSAGE
  sends a message to any individual or combination of MVS/MCS consoles, or CICS TD queues.
CONVERSE
  sends a message to any individual or combination of MVS/MCS consoles and receives a reply from one of them.
RETRIEVE_MESSAGE
  builds a message and places it in a buffer passed by the caller.
INQUIRE_MESSAGE_LENGTH
  returns the length of a terminal end user message.
INQUIRE_MESSAGE
  returns the requested data, held by the ME domain (for example, Default_Language).
VALIDATE_LANGUAGE_CODE
  checks whether a three-character language code is valid.
VALIDATE_LANGUAGE_SUFFIX
  checks whether a one-character language suffix is valid.
DFHMESR | Collects the system initialization parameter overrides for a particular CICS start
DFHMETRI | ME domain offline trace interpretation routine
DFHMEWS | Writes a symptom record containing a symptom string to SYS1.LOGREC by using the MVS SYMRBLD macro
DFHMEW | Provides support to execute the MVS WTOR SVC

Exits

There is one global user exit point in the message domain: XMEOUT. See the CICS Customization Guide for further information.

Trace

The point IDs for the message domain are of the form ME xxxx; the corresponding trace levels are ME 1, ME 2, and Exc.

For more information about the trace points, see the CICS Trace Entries. For more information about using traces in problem determination, see the CICS Problem Determination Guide.
Chapter 55. Message generation program

The message generation program provides an interface for sending CICS messages to the terminal user only.

Design overview

The input to the message generation program (DFHMGP) consists of the binary number of the message to be produced, the identifier of the component issuing the message, and any information to be inserted in the message. DFHMGP builds the complete message using a prototype held in the message prototype control table, also known as the message generation table (DFHMGT). The message text itself is held not in DFHMGT but in the message domain, from which it is retrieved by the DFHMGPME routine (a component of the DFHMGP load module) when required. DFHMGP finally sends the message to the appropriate terminal.

The prototype statements are invocations of the DFHMGM TYPE=TEXT macro, and are contained in copybooks held in DFHMGT.

The message prototype control table consists of a series of copybooks, DFHMGTnn, each of which contains 1 through 100 messages. They are arranged in such a way that each DFHMGTnn copybook contains prototypes for messages that have identifiers of the form DFHccnnxx, where cc is the 2-character identifier of the component issuing the message, nn is the numerical part of the copybook name, and xx is in the range 00 through 99. For example, the prototype for message DFHAC2214 (belonging to the AC component) is in copybook DFHMGT22.

Within each copybook are invocations of DFHMGM in ascending message number order. All messages sent to the terminal end user have both OPTION=NLS and COMPID specified on their DFHMGM invocations.

The main operands of the DFHMGM TYPE=TEXT macro are:

- MSGNO = actual message number
- COMPID = 2-character identifier of component issuing the message (this forms part of the message identifier)
- OPTION = any special options, for example, (NLS) for messages that require NLS enabling.

Other operands are provided on the DFHMGM invocations, but in general these are no longer used.

Modules

DFHMGP, DFHMGT

Exits

No global user exit points are provided for this function.
Trace

The following point ID is provided for this function:
• AP 00E0, for which the trace level is AP 1.

For more information about the trace points, see the CICS Trace Entries. For more information about using traces in problem determination, see the CICS Problem Determination Guide.
Chapter 56. Message switching

This function provides the user with a general-purpose message-switching capability while CICS is running.

The facility, which can route messages to one or more destinations, is initiated by the CMSG transaction, or a user-chosen replacement, read from the terminal. For further information about this transaction, see the CICS Supplied Transactions manual.

Design overview

Message switching runs as a task under CICS. A terminal operator requests activation of this task by entry of the CMSG transaction identifier (or another installation-defined 4-character transaction identifier), followed by appropriate parameters. After it has been initiated, message switching interfaces with CICS basic mapping support (BMS) and CICS control functions.

Although message switching appears conversational to the terminal operator, the message switching task is terminated with each terminal response. Conversation is forced, if continuation is possible, by effectively terminating the transaction with an EXEC CICS RETURN TRANSID(xxxx) command, where xxxx is the transaction identifier taken from the task’s PCT entry.

Figure 80 shows the message-switching interfaces.
Message switching

Notes:

1. If the first 4 characters of the terminal input/output area (TIOA) (not including a possible set buffer address (SBA) sequence from an IBM 3270 Information Display System) do not match the transaction identifier in the task’s PCT entry, this task must have started as part of a conversation in which a previous task has set up the next transaction identifier. A "C" immediately following the transaction identifier is also a forced continuation. In such a case, information has been stored in, and has to be retrieved from, temporary storage (using a record key of 1-byte X'FC', 4-byte terminal identifier, and 3-byte C"MSG") to allow the task to resume where it left off.

2. The operands in the input TIOA are processed and their values and status are stored in the TWA.

3. If a ROUTE operand specifies terminal list tables (TLTs) for a standard routing list, the program manager domain is called to load the requested TLTs.

4. Message switching requests storage areas for:
   - Building route lists (one or more segments, each of which has room for the number of destinations specified by MSRTELNG, an EQU within the program).
   - Constructing a record to be placed in temporary storage.
   - Providing the message text to BMS in any of the following situations:
     - Message parts from previous inputs exceed the current TIOA size
     - A message is completed in the current TIOA but has parts from previous inputs
     - A heading has been requested but the message in the current TIOA is too close to TIOADBA to allow the header to be inserted.

5. Message switching requests BMS routing functions by means of the DFHBMS TYPE=ROUTE macro. The message text is sent using DFHBMS TYPE=TEXTBLD, and completion of the message is indicated by DFHBMS TYPE=PAGEOUT. BMS returns the status of destinations and any error indications in response to the DFHBMS TYPE=CHECK macro.

6. Message switching interfaces with BMS using DFHBMS TYPE=(EDIT,OUT) and with CICS terminal control using DFHTC TYPE=WRITE for the IBM 3270 Information Display System only, in providing responses to terminals. These can indicate normal completion, signal that input is to continue, or provide notification of input error.

7. Like any other task, message switching has a task control area (TCA) in which values may be placed prior to issuing CICS macros, and from which any returned values can be retrieved after an operation. All values for the DFHBMS TYPE=ROUTE macro are placed in the TCA because they are created at execution time. The TWA is used for storing status information (partly saved in temporary storage across conversations) and space for work. The DFHMSP module is reentrant.

Control blocks

See the list of control blocks in Chapter 10. Basic mapping support on page 133.

Modules

DFHMSP (the message switching program) is invoked by the CMSG transaction. DFHMSP’s purpose is to route a message entered at the terminal to one or more operator-defined terminals or to other operators.
Exits

No global user exit points are provided for this function.

Trace

No trace points are provided for this function.

External interfaces

See Figure 80 on page 833 for external calls made to other areas or domains.
Chapter 57. Monitoring domain (MN)

The monitoring domain is responsible for all monitoring functions within CICS. These functions enable the user to measure the amount of CPU, storage, temporary-storage requests, and so on used per task, and hence charge customers for computing services and help review the performance of a CICS system.

Monitoring domain’s specific gates

Table 74 summarizes the monitoring domain’s specific gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and whether or not the functions are available through the exit programming interface (XPI).

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>XPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNMN</td>
<td>MN 0201</td>
<td>EXCEPTION_DATA_PUT</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>MN 0202</td>
<td>PERFORMANCE_DATA_PUT</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INQUIRE_MONITORING_DATA</td>
<td>YES*</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MONITOR</td>
<td>YES</td>
</tr>
<tr>
<td>MNSR</td>
<td>MN 0301</td>
<td>SET_MCT_SUFFIX</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>MN 0302</td>
<td>SET_MONITORING</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INQ_MONITORING</td>
<td>NO</td>
</tr>
<tr>
<td>MNXM</td>
<td>MN 0A01</td>
<td>TRANSACTION_INITIALIZATION</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>MN 0A02</td>
<td>TRANSACTION_TERMINATION</td>
<td>NO</td>
</tr>
</tbody>
</table>

* In a modified form, without a transaction number.

MNMN gate, EXCEPTION_DATA_PUT function

The EXCEPTION_DATA_PUT function of the MNMN gate is used to produce an exception record at the completion of an EXCEPTION condition.

Input parameters

EXCEPTION_START
is the start time of the exception in stored clock (STCK) format.

EXCEPTION_STOP
is the stop time of the exception in STCK format.

RESOURCE_TYPE
is the type of resource for which the exception data is to be recorded.

RESOURCE_ID
is the identifier of the resource for which the exception data is to be recorded.

EXCEPTION_TYPE
is the type of exception to be recorded.

Output parameters

RESPONSE
is the domain’s response to the call. It can have any of these values:
Monitoring domain (MN)

**OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED**

[**REASON**] is returned when RESPONSE is DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, INVALID_MONITORING_TOKEN, LOOP</td>
</tr>
</tbody>
</table>

**MNMN gate, PERFORMANCE_DATA_PUT function**

The PERFORMANCE_DATA_PUT function of the MNMN gate is used to produce a performance record and reset task monitoring information for a conversational task or a syncpoint.

**Input parameters**

**RECORD_TYPE**

is the reason for the record to be output.

**Output parameters**

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[**REASON**] is returned when RESPONSE is DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, INVALID_MONITORING_TOKEN, LOOP</td>
</tr>
</tbody>
</table>

**MNMN gate, INQUIRE_MONITORING_DATA function**

The INQUIRE_MONITORING_DATA function of the MNMN gate is used to access a transaction’s monitoring information.

**Input parameters**

**[TRANSACTION_NUMBER]**

is the transaction number for which monitoring data is required.

**DATA_BUFFER**

specifies the address and length of a buffer for the monitoring data.

**Output parameters**

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[**REASON**] is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>MONITOR_DATA_UNAVAILABLE, LENGTH_ERROR</td>
</tr>
</tbody>
</table>
**MNMN gate, MONITOR function**

The MONITOR function of the MNMN gate is called to process a user event-monitoring point (EMP).

**Input parameters**

POINT

is a value in the range 0 through 255 corresponding to a monitoring point identifier defined in the monitoring control table (MCT).

[ENTRYNAME]

is an ID qualifier, 1 through 8 bytes, corresponding to an entry name specified in the MCT.

[DATA1]

supplies 4 bytes of data to be used in the operations performed by this user’s EMP.

[DATA2]

supplies 4 bytes of data to be used in the operations performed by this user’s EMP.

**Output parameters**

RESPONSE

is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON]

is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, INVALID_MONITORING_TOKEN, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>POINT_NOT_DEFINED, DATA1_NOT_SPECIFIED, DATA2_NOT_SPECIFIED,</td>
</tr>
<tr>
<td></td>
<td>INVALID_DATA1_VALUE, INVALID_DATA2_VALUE</td>
</tr>
</tbody>
</table>

**MNSR gate, SET_MCT_SUFFIX function**

The SET_MCT_SUFFIX function of the MNSR gate is used to identify to the monitoring domain the suffix of the monitoring control table (MCT).

**Input parameters**

SUFFIX

is the 2-character MCT suffix.

**Output parameters**

RESPONSE

is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON]

is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
</tbody>
</table>
MNSR gate, SET_MONITORING function
The SET_MONITORING function of the MNSR gate is used to set the monitoring classes on or off and to change the monitoring options.

Input parameters
[CONVERSE]
indicates if a transaction performance class record is to be produced for conversational tasks for each pair of terminal control I/O requests. It can have either of these values:
YES|NO

[EXCEPTION_STATUS]
indicates the exception class monitoring setting. It can have either of these values:
ON|OFF

[FREQUENCY]
is the interval for which monitoring automatically produces a transaction performance class record for any long-running transaction. Frequency times are 0, or in the range 001500 through 240000. The default frequency value is 0, which means that frequency monitoring is inactive.

[MONITORING_STATUS]
indicates the monitoring status setting. It can have either of these values:
ON|OFF

[PERFORMANCE_STATUS]
indicates the performance class monitoring setting. It can have either of these values:
ON|OFF

[SUBSYSTEM_ID]
specifies the 4-character subsystem-id to be used in the sysevent records. The default is the first four character of the generic applid. implicit syncpoint (unit-of-work).

[SYNCPOINT]
indicates if a transaction performance class record is to be produced when a transaction takes an explicit or implicit syncpoint (unit-of-work). It can have either of these values:
YES|NO

[SYSEVENT_STATUS]
indicates the SYSEVENT class monitoring setting. It can have either of these values:
ON|OFF

[TIME]
indicates whether the monitoring timestamp fields returned on the INQUIRE_MONITORING_DATA function are to be in GMT or Local time. It can have either of these values:
GMT|LOCAL
Output parameters

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>INVALID,FREQUENCY</td>
</tr>
</tbody>
</table>

MNSR gate, INQ_MONITORING function

The INQ_MONITORING function of the MNSR gate is used to enquire on the monitoring classes and the monitoring options.

Input parameters
None.

Output parameters

CONVERSE
indicates if a transaction performance class record is to be produced for conversational tasks for each pair of terminal control I/O requests. It can have either of these values:
YES | NO

EXCEPTION_STATUS
indicates whether exception class monitoring is active. It can have either of these values:
ON | OFF

FREQUENCY
is the interval for which monitoring automatically produces a transaction performance class record for any long-running transaction. Frequency times are 0, or in the range 001500 through 240000. The default frequency value is 0, which means that frequency monitoring is inactive.

MONITORING_STATUS
indicates whether monitoring is active. It can have either of these values:
ON | OFF

PERFORMANCE_STATUS
indicates whether performance class monitoring is active. It can have either of these values:
ON | OFF

SUBSYSTEM_ID
specifies the 4-character subsystem-id to be used in the sysevent records. The default is the first four character of the generic applid. implicit syncpoint (unit-of-work).

SYNCPOINT
indicates if a transaction performance class record is to be produced when a transaction takes an explicit or implicit syncpoint (unit-of-work). It can have either of these values:
Monitoring domain (MN)

YES|NO

SYSEVENT_STATUS
indicates whether SYSEVENT class monitoring is active. It can have either of these values:
ON|OFF

TIME indicates whether the monitoring timestamp fields returned on the INQUIRE_MONITORING_DATA function are to be in GMT or Local time. It can have either of these values:
GMT|LOCAL

RESPONSE is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>SUBSYSTEM_ID_NOT_AVAILABLE</td>
</tr>
</tbody>
</table>

MNXM gate, TRANSACTION_INITIALIZATION function

The TRANSACTION_INITIALIZATION function of the MNXM gate is used to inform the monitoring domain of a transaction attach request so that the monitoring domain can allocate task monitoring storage.

Input parameters

TASK.Attach.T ime is the time when this task was attached.

TCLASS.DELAY.T ime is the time this task was delayed due to the transaction class (if any) limit for this transaction being reached.

MXT.DELAY.T ime is the time this task was delayed due to the maximum user task limit (MXT) being reached.

INITIAL_DISPATCH.T ime is the time when this task was first dispatched after attach.

Output parameters

RESPONSE is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, INVALID_MONITORING_TOKEN, LOOP</td>
</tr>
</tbody>
</table>
**MNXM gate, TRANSACTION_TERMINATION function**

The TRANSACTION_TERMINATION function of the MNXM gate is used to inform the monitoring domain of a transaction detach request, so that the monitoring domain can report on task monitoring information and then release the storage.

**Input parameters**

None.

**Output parameters**

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

- OK
- EXCEPTION
- DISASTER
- INVALID
- KERNERROR
- PURGED

**[REASON]**

is returned when RESPONSE is DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, INVALID_MONITORING_TOKEN, LOOP</td>
</tr>
</tbody>
</table>

### Monitoring domain’s generic gates

[Table 75](#) summarizes the monitoring domain’s generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

**Table 75. Monitoring domain’s generic gates**

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>APUE</td>
<td>MN 0601</td>
<td>SET_EXIT_STATUS</td>
<td>APUE</td>
</tr>
<tr>
<td></td>
<td>MN 0602</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DMDM</td>
<td>MN 0101</td>
<td>INITIALIZED_DOMAIN</td>
<td>DMDM</td>
</tr>
<tr>
<td></td>
<td>MN 0102</td>
<td>QUIESCE_DOMAIN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TERMINATE_DOMAIN</td>
<td></td>
</tr>
<tr>
<td>STST</td>
<td>MN 0401</td>
<td>COLLECT_STATISTICS</td>
<td>STST</td>
</tr>
<tr>
<td></td>
<td>MN 0402</td>
<td>COLLECTRESOURCE_STATS</td>
<td></td>
</tr>
<tr>
<td>TISR</td>
<td>MN 0801</td>
<td>NOTIFY</td>
<td>TISR</td>
</tr>
<tr>
<td></td>
<td>MN 0802</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XMNT</td>
<td>MN 0901</td>
<td>MXT_CHANGE_NOTIFY</td>
<td>XMNT</td>
</tr>
<tr>
<td></td>
<td>MN 0902</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For descriptions of these functions and their input and output parameters, refer to the sections dealing with the corresponding generic formats:
Monitoring domain (MN)

Functions and parameters

<table>
<thead>
<tr>
<th>Format</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>APUE</td>
<td>Application domain’s generic formats</td>
<td>100</td>
</tr>
<tr>
<td>DMDM</td>
<td>Domain manager domain’s generic formats</td>
<td>377</td>
</tr>
<tr>
<td>STST</td>
<td>Statistics domain’s generic format</td>
<td>1087</td>
</tr>
<tr>
<td>TISR</td>
<td>Timer domain’s generic format</td>
<td>1228</td>
</tr>
<tr>
<td>XMNT</td>
<td>Transaction manager domain’s generic format</td>
<td>1304</td>
</tr>
</tbody>
</table>

In initialization processing, the monitoring domain sets the initial monitoring options:
- Monitoring control table suffix
- Initial monitoring status
- Initial event monitoring status
- Initial exception class monitoring status
- Initial performance class monitoring status
- Initial converse option
- Initial syncpoint option
- Initial time option
- Initial frequency option
- Initial subsystem id.

For a cold start, the information comes from the system initialization parameters; for any other type of start, the information comes from the global catalog, but is then modified by any relevant system initialization parameters.

In addition:
- If necessary, the monitoring control table (MCT) is loaded and initialized.
- If performance class monitoring is active, CPU timing is started.
- The monitoring domain user exit gate is enabled.
- Messages are sent to the console to indicate whether monitoring is active, and what MCT suffix (if any) is being used.

In quiesce processing, the monitoring domain waits for all transactions that it is monitoring to terminate. Then the final data in the performance class buffer, if any, is written to SMF.

The monitoring domain does no termination processing.

Modules

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHMNDM</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>INITIALIZE_DOMAIN</td>
</tr>
<tr>
<td></td>
<td>QUIESCE_DOMAIN</td>
</tr>
<tr>
<td></td>
<td>TERMINATE_DOMAIN</td>
</tr>
<tr>
<td>DFHMNDUF</td>
<td>Formats the MN domain control blocks in a CICS system dump</td>
</tr>
</tbody>
</table>
### Module Function

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
</table>
| DFHMNMN | Handles the following requests:  
| &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;EXCEPTION_DATA_PUT  
| &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;PERFORMANCE_DATA_PUT  
| &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;INQUIRE_MONITORING_DATA  
| &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;MONITOR |
| DFHMNNNT | Handles the following request:  
| &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;MXT_CHANGE_NOTIFY |
| DFHMNSR | Handles the following requests:  
| &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;SET_MCT_SUFFIX  
| &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;SET_MONITORING  
| &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;INQ_MONITORING |
| DFHMNST | Handles the following requests:  
| &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;COLLECT_STATISTICS  
| &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;COLLECT_RESOURCE_STATS |
| DFHMNSU | Handles monitoring domain subroutine requests of format MNSU:  
| &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;UPDATE_CATALOGUE  
| &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;MONITORING_DATASET_PUT  
| &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;SYSEVENT_WRITE |
| DFHMNSVC | Provides SMFEWTM and SYSEVENT authorized services with GTF tracing (GTRACE) |
| DFHMNTI | Handles the following request:  
| &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;NOTIFY |
| DFHMNTRI | Provides a trace interpretation routine for CICS dumps and traces |
| DFHMNUUE | Provides a SET_EXIT_STATUS (services user exit) routine to enable or disable an exit |
| DFHMNXM | Handles the following requests:  
| &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;TRANSACTION_INITIALIZATION  
| &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;TRANSACTION_TERMINATION |

### Exits

There is one global user exit point in the monitoring domain: XMNOUT. See the [CICS Customization Guide](#) for further information.

### Trace

The point IDs for the monitoring domain are of the form MN xxxx; the corresponding trace levels are MN 1, MN 3, and Exc.

For more information about the trace points, see the [CICS Trace Entries](#). For more information about using traces in problem determination, see the [CICS Problem Determination Guide](#).
Chapter 58. Multiregion operation (MRO)

CICS multiregion operation (MRO) enables CICS regions that are running in the same MVS image, or in the same MVS sysplex, to communicate with each other. MRO does not support communication between a CICS system and a non-CICS system such as IMS.8

ACF/VTAM and SNA networking facilities are not required for MRO. The support within CICS that enables region-to-region communication is called **interregion communication**

The facilities provided by MRO include:
- Transaction routing
- Distributed transaction processing
- Function shipping
- Asynchronous processing
- Distributed program link.

For more information about the design and implementation of interregion communication facilities, see “Chapter 45. Interregion communication (IRC)” on page 708. For descriptions of the facilities provided by MRO, see:

- “Chapter 24. Distributed program link” on page 327
- “Chapter 25. Distributed transaction processing” on page 331
- “Chapter 42. Function shipping” on page 673
- “Chapter 102. Transaction routing” on page 1311.

8. The external CICS interface (EXCI) uses a specialized form of MRO link to support: communication between MVS batch programs and CICS; DCE remote procedure calls to CICS programs.
Multiregion operation (MRO)
Chapter 59. Node abnormal condition program

DFHZNAC is a CICS program used by terminal control to analyze abnormal terminal conditions that are logical unit or node errors detected by VTAM. VTAM notifies the CICS terminal control program that there is a terminal error, and the terminal control program places the terminal out of service. The terminal control program then invokes DFHZNAC, which writes any error messages to the CSNE transient data destination.

Design overview

The node abnormal condition program (DFHZNAC) can be called for any of several reasons:

- As a central point of control for most VTAM-related error situations, error actions can be standardized in table form, allowing for easy addition and alteration to the way conditions are processed.
- Some exception conditions that are not errors are also processed by DFHZNAC, but some exception conditions that are errors are not processed by DFHZNAC.
- It provides a single point of user interface to those who want to change the default actions for an error requiring at most one user program (NEP)—see "Chapter 60. Node error program" on page 853.

To process conditions that are not associated with a known terminal, the dummy TCTTE is used. It is invoked by placing a TCTTE on the system error queue with a 1-byte code relating to the condition. Placing it on the queue makes the TCTTE ‘temporary OUTSERV’ (TCTTESOS); that is, the decision is pending the outcome of DFHZNAC.

The activate scan routine (DFHZACT) is responsible for attaching the CSNE transaction to run DFHZNAC; this is done during CICS initialization. The CSNE transaction remains in the system until CICS or VTAM is quiesced. If DFHZNAC itself abends, or VTAM is closed and then restarted, DFHZACT attaches a new CSNE transaction when there is more work for DFHZNAC to do.

There is only ever one CSNE transaction in the system at any one time. (This should not be confused with the CSNE transaction that is attached by the remote delete processing of autoinstall.)

Once DFHZNAC has been called, it runs down the system error queue, processing each error for each TCTTE on the queue. When there is no more work to be done, DFHZNAC suspends itself, to be resumed by DFHZACT when further processing is required.

Note that the system error queue need not be empty before DFHZNAC terminates; errors can be left on the queue to be processed later. For example, in an XRF environment, some error codes cannot be handled until the alternate CICS system has taken over; that is, it has passed the ‘initialization complete’ stage. If DFHZNAC is passed a TCTTE indicating such an error, it leaves that entry on the queue.

Node abnormal condition program (NACP) processing involves mapping the error code (placed into the TCTTE by a DFHZERRM macro call) to a set of actions,
performing any specific processing for that error code, accumulating the actions for all the error codes in that TCTTE, and then performing the actions.

Figure 81 on page 851 shows the NACP error code processing. The numbers in Figure 81 refer to the following notes, which use the table entry for DFHZC3424 as the example:

DFHZNCM MSGNO=3424,
E1=S88,
E2=NULL,
E3=NULL,
E4=NULL,
ACT=(ABSEND, ABRECV, ABTASK, CLSDST, SIMLOG),
CODE=NSP02,
TYPE=ENTRY
For each TCTTE

1. Map error code to a DFHZNCA table entry

2. Call any pre-sense exits designated by the entry

3. If sense code associated, call DFHZNCS routine

4. If any RPL feedback code, call DFHZNCV routine

5. Call any pre-NEP exits

6. Call DFHZNEP

7. Output the error-code message

8. Process any 'unavailable printer' error

9. Accumulate actions so far

10. Output any sense message

11. Output any VTAM_3270 message

12. Call any post-NEP exit

13. Another error code for this TCTTE?

14. Retrieve accumulated actions

15. Call the action routines

16. Output the 'actions taken' message

17. Check again for added error codes and enter again at the top

18. If any work resulting from the actions, add TCTTE to the DFHZACT work queue

get next TCTTE
Node abnormal condition program

Notes:

1. The error codes in TCTEVRC* and default actions are defined in the VTAM-associated errors section of the CICS Trace Entries. In the example, TCTVRC5 contains X'5C', which equates to TCZNSP02 (ref CODE=NSP02).

2. Errors that involve SNA sense have it saved in TCTEVNSS. It is processed by code in copy book DFHZNCS.

3. Call any pre-NEP exits specified by the table entry; for example, E1=S88 references routine NAPES88.

4. Call the node error program (NEP), passing a parameter list via a COMMAREA. This call may or may not change the default actions. The operation of the NEP is described in the CICS Customization Guide and the "Chapter 60. Node error program" on page 855.

5. Output error-code message associated with the table entry (DFHZC3424 from MSGNO=3424) to the CSNE log.

6. Check for ‘unavailable printer error’—this caters for a screen copy request that is unable to find an eligible printer if the first choice is unavailable.

7. Because there can be multiple error codes, the actions are accumulated now and performed together later.

8. Output any sense message resulting from the DFHZZNCS call, to the CSNE log.

9. Output any VTAM_3270 message resulting from the DFHZZNCS call (if it was non-SNA) to the CSNE log.

10. Call the post-NEP exit, if any (E4=NULL, no routine).

11. Loop for each error code in TCTEVRC*.

12. When all the error codes for this TCTTE that can be processed at this time have been processed, retrieve the actions that have been accumulated, such as ACT=(ABSEND, ABRECV, ABTASK, CLSDST, SIMLOG).

13. Call the action routine to process each of the actions.

14. Output the ‘actions taken’ message DFHZC3437 to the CSNE log.

15. Check again for any error codes added asynchronously while the CSNE transaction was running.

16. Queue any work resulting from the actions to the activate scan routine.

Control blocks

DFHZNAC references CSA, its own TCA, JCA, TCT prefix, TIOA, NIB, PCT, SIT, TCTWE, VTAM RPL, VTAM ACB, and the NACP/NEP communication area.

As would be expected, however, the processing mainly concerns access to the TCTTE, and to the NACP/NEP communication area (COMMAREA), which is mapped by the DFHNEPCA DSECT.

See the CICS Data Areas manual or the CICS Customization Guide for a detailed description of the NEP communication area.
### Modules

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHZNAC</td>
<td>Processes the system error queue of TCTTEs and contains the central</td>
</tr>
<tr>
<td></td>
<td>structure of NACP, outlined in <a href="#">Figure 81 on page 85</a>. It contains the</td>
</tr>
<tr>
<td></td>
<td>following copy books:</td>
</tr>
<tr>
<td></td>
<td>DFHZNCA This copy book contains the exit routines for each error code and</td>
</tr>
<tr>
<td></td>
<td>the error code table itself built by DFHZNCM macros.</td>
</tr>
<tr>
<td></td>
<td>DFHZNCE Links to the user node error program (DFHZNEP) and responds to</td>
</tr>
<tr>
<td></td>
<td>the action flag settings in the NACP/NEP COMMAREA.</td>
</tr>
<tr>
<td></td>
<td>DFHZNCS Processes the SNA sense codes and contains the sense code tables</td>
</tr>
<tr>
<td></td>
<td>built using a combination of DFHZMJM and DFHZNCM macros.</td>
</tr>
<tr>
<td></td>
<td>DFHZNCV Contains the VTAM return code table.</td>
</tr>
<tr>
<td></td>
<td>DFHZNCM The macro to build the error code table.</td>
</tr>
<tr>
<td></td>
<td>DFHZMJM The macro to build the sense code table.</td>
</tr>
</tbody>
</table>

### Exits

No global user exit points are provided for this function.

### Trace

The following point IDs are provided for the node abnormal condition program, as part of terminal control:

- AP FCxx, for which the trace levels are TC 1, TC 2, and Exc
- AP FD7E, for which the trace level is TC 1.

For more information about the trace points, see the [CICS Trace Entries](#). For more information about using traces in problem determination, see the [CICS Problem Determination Guide](#).

### Statistics

The only statistical field that DFHZNAC updates is TCTTETE. Because DFHZNAC is the main module for terminal errors, it has primary responsibility for updating the node error count.
Chapter 60. Node error program

CICS provides a user-replaceable node error program, DFHZNEP, which assists the user in the following ways:

- It provides a general environment within which it is easy for users to add their own error processors.
- It provides the fundamental error recovery actions for a VTAM 3270 network.
- It serves as the default node error program (NEP), where the user selects a NEP at system initialization.

The DFHZNEP program can be one of the following:

- The CICS-supplied default NEP
- A skeleton sample NEP generated using the DFHSNEP macro
- A user-written NEP generated using the DFHSNEP macro.

Design overview

The purpose of the NEP is to allow user-dependent processing whenever a communication system event is reported to CICS. An example of the processing that can be done is to analyze the event and override the default action set by DFHZNAC. When NEP processing is complete, control returns to DFHZNAC.

The default node error program sets the ‘print TCTTE’ action flag (TWAOTCTE in the user option byte TWAOPT1, defined in DFHNEPCA) if a VTAM storage problem has been detected; otherwise, it performs no processing, and leaves the action flags set by DFHZNAC unchanged.

The skeleton sample NEP provided by CICS can provide extended error handling for VTAM terminals, and is generated by means of the DFHSNEP macro. This procedure is described in the CICS Customization Guide.

The DFHSNEP macro can also be used to generate a user-written NEP. Interactions between the applications and VTAM depend on characteristics of the transactions and the installation. Each system has different characteristics. The CICS-provided skeleton NEP is a framework for a user-written NEP to handle network error conditions that may be unique to a particular installation.

Guidance information about NEP coding is given in the CICS Recovery and Restart Guide. Reference information about NEP coding is given in the CICS Customization Guide.

Modules

DFHZNEP

Exits

No global user exit points are provided for this function.
Trace

No trace points are provided specifically for this function; however, trace entries are made from DFHZNAC immediately before and after calling the node error program.

Point IDs AP FC71 and AP FC72, with a trace level of TC 1, correspond to these trace entries.

For more information about the trace points, see the CICS Trace Entries. For more information about using traces in problem determination, see the CICS Problem Determination Guide.
Chapter 61. Object Transaction Service domain (OT)

The object transaction service domain provides services to manage OTS transactions.

Object Transaction Service domain’s specific gates

Table 76 summarizes the OT domain’s specific gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and whether or not the functions are available through the exit programming interface (XPI).

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>XPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>OTTR</td>
<td>OT 0401</td>
<td>IMPORT_TRAN</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>OT 0402</td>
<td>BEGIN_TRAN</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COMMIT_ONE_PHASE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PREPARE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COMMIT</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ROLLBACK</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SET_ROLLBACK_ONLY</td>
<td>NO</td>
</tr>
<tr>
<td>OTSU</td>
<td>XM 0501</td>
<td>ADD_SUBORDINATE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>XM 0502</td>
<td>SET_VOTE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FORGET</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RESYNC</td>
<td>NO</td>
</tr>
<tr>
<td>OTCO</td>
<td>XM 0601</td>
<td>SET_COORDINATOR</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>XM 0602</td>
<td>FORGET</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RESYNC</td>
<td>NO</td>
</tr>
<tr>
<td>OTRM</td>
<td>XM 0701</td>
<td>PERFORM_PRELOGGING</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>XM 0702</td>
<td>PERFORM_PREPARE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PERFORM_COMMIT</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>REPLY_DO_COMMIT</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SEND_DO_COMMIT</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PERFORM_SHUNT</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PERFORM_UNSHUNT</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>START_BACKOUT</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DELIVER_BACKOUT_DATA</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>END_BACKOUT</td>
<td>NO</td>
</tr>
</tbody>
</table>

OTTR gate, IMPORT_TRAN function

The IMPORT_TRAN function of the OTTR gate is used to import an OTS transaction to a task.

Input parameters

FORMAT_ID
The OTS transactions format identifier.

BQUAL_LEN
The batch qualifier length of the OTS transaction.

TID_BLOCK_IN
The OTS transaction identifier (TID) of the transaction being imported.

TIMEOUT
The OTS transaction timeout value.

LOGICAL_SERVER
The name of the logical server within which the transaction is executing.
Object Transaction Service domain (OT)

**PUBLIC_ID**

The Request Stream public identifier associated with the transaction.

**Output parameters**

UOW_ID

The identifier of the logical unit of work into which the OTS transaction was imported.

RESPONSE

is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]

is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>TID_TOO_LONG</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

**OTTR gate, BEGIN.TRAN function**

The BEGIN.TRAN function of the OTTR gate is used to create a new OTS transaction.

**Input parameters**

[TIMEOUT]

The timeout specified for the new OTS transaction.

TID BUFFER OUT

The OTS transaction identifier (TID) of the transaction created.

LOGICAL_SERVER

The name of the logical server within which the transaction is executing.

PUBLIC_ID

The Request Stream public identifier associated with the transaction.

**Output parameters**

FORMAT_ID

The OTS transactions format identifier.

BQUAL_LEN

The batch qualifier length of the OTS transaction.

[TIMEOUT]

The default OTS transaction timeout value.

RESPONSE

is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]

is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>TID_TOO_LONG, UOW_ROLLED_BACK</td>
</tr>
</tbody>
</table>
OTTR gate, COMMIT_ONE_PHASE function

The COMMIT_ONE_PHASE function of the OTTR gate is used to attempt to commit the current OTS transaction.

Input parameters
None

Output parameters

STATUS
The outcome of the OTS transaction. It can have either of these values:
COMMITTED|ROLLEDBACK

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
None

OTTR gate, PREPARE function

The PREPARE function of the OTTR gate is used to perform the first phase of the syncpoint of an OTS transaction.

Input parameters
None

Output parameters

VOTE
The vote from first phase of syncpoint. It can have any of these values:
YES|NO|READ_ONLY|HEURISTIC_MIXED

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
None

OTTR gate, COMMIT function

The COMMIT function of the OTTR gate is used to perform the second phase of the syncpoint of an OTS transaction.

Input parameters
None.

Output parameters

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>UOW_ROLLEDBACK</td>
</tr>
</tbody>
</table>

Chapter 61. Object Transaction Service domain (OT)  859
Object Transaction Service domain (OT)

**OTTR gate, SET_ROLLBACKONLY function**

The SET_ROLLBACKONLY function of the OTTR gate is used to ensure that the OTS transaction will rollback when it comes to syncpoint.

**Input parameters**
None.

**Output parameters**

- **RESPONSE**
  is the domain's response to the call. It can have any of these values: OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

- **[REASON]**
  None

**OTSU gate, ADD_SUBORDINATE function**

The ADD_SUBORDINATE function of the OTSU gate is used add a subordinate participant to the OTS transaction.

**Input parameters**

- **IOR_BLOCK**
  Block containing the CORBA IOR of the OTS Resource that is being added as a subordinate participant in the OTS transaction.

- **HOST_BLOCK**
  Block containing the name of the TCPIP host where the subordinate OTS resource resides.

**Output parameters**

- **SUBORDINATE_TOKEN**
  token representing the added Resource.

- **RESPONSE**
  is the domain's response to the call. It can have any of these values: OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

- **[REASON]**
  is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>IOR_TOO_LONG, HOST_TOO_LONG</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ADD_LINK_FAILED</td>
</tr>
</tbody>
</table>

**OTSU gate, SET_VOTE function**

The SET_VOTE function of the OTSU gate is used record the vote that results from a PREPARE method being invoked on the OTS Resource represented by the given SUBORDINATE_TOKEN.

**Input parameters**

- **SUBORDINATE_TOKEN**
  Token representing the subordinate OTS resource.

- **VOTE**
  The vote resulting from the first phase of syncpoint on the subordinate resource. It can have one of the following values:
Output parameters

RESPONSE

is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]

is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>UNKNOWN_SUBORDINATE,INVALID_VOTE</td>
</tr>
<tr>
<td>DISASTER</td>
<td>RECORD_VOTE_FAILED</td>
</tr>
</tbody>
</table>

OTSU gate, FORGET function

The FORGET function of the OTSU gate is used signal the fact that the obligation to the subordinate resource has been discharged.

Input parameters

SUBORDINATE_TOKEN

Token representing the subordinate OTS resource.

Output parameters

RESPONSE

is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]

is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>UNKNOWN_SUBORDINATE</td>
</tr>
<tr>
<td>DISASTER</td>
<td>INBOUND_FLOW_FAILED</td>
</tr>
</tbody>
</table>

OTSU gate, RESYNC function

The RESYNC function of the OTSU gate is used to initiate the resynchronisation protocol with the subordinate resource identified by the given IOR.

Input parameters

IOR_BLOCK

Block containing the CORBA IOR of the OTS Resource with which to resynchronise.

UOWID

Identification of the local logical unit of work managing the OTS transaction.

Output parameters

RESPONSE

is the domain’s response to the call. It can have any of these values:
**OTCO gate, SET_COORDINATOR function**

The SET_COORDINATOR function of the OTCO gate is used to make known the CORBA object that will coordinate this part of the OTS transaction.

**Input parameters**

**IOR_BLOCK**
Block containing the CORBA IOR of the OTS Coordinator.

**HOST_BLOCK**
Block containing the name of the TCPIP host where the coordinator resides.

**Output parameters**

**COORDINATOR_TOKEN**
token representing the coordinator.

**RESPONSE**
is the domain's response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

**[REASON]**
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>IOR_TOO_LONG, HOST_TO_LONG</td>
</tr>
</tbody>
</table>

**OTCO gate, FORGET function**

The FORGET function of the OTCO gate is used to signal the fact that the obligation to the coordinator has been discharged.

**Input parameters**

**SUBORDINATE_TOKEN**
Token representing the subordinate OTS resource.

**Output parameters**

**RESPONSE**
is the domain's response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

**[REASON]**
is returned when RESPONSE is EXCEPTION or DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>UNKNOWN_SUBORDINATE</td>
</tr>
<tr>
<td>DISASTER</td>
<td>INBOUND_FLOW_FAILED</td>
</tr>
</tbody>
</table>
Modules

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHOTDM</td>
<td>Domain initialisation and termination.</td>
</tr>
<tr>
<td></td>
<td>PRE_INITIALIZE</td>
</tr>
<tr>
<td></td>
<td>INITIALIZE_DOMAIN</td>
</tr>
<tr>
<td></td>
<td>QUIESCE_DOMAIN</td>
</tr>
<tr>
<td></td>
<td>TERMINATE_DOMAIN</td>
</tr>
<tr>
<td>DFHOTRM</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>ATTACH</td>
</tr>
<tr>
<td>DFHOTTR</td>
<td>Handles requests on the OTTR gate.</td>
</tr>
<tr>
<td>DFHOTSU</td>
<td>Handles requests on the OTSU gate.</td>
</tr>
<tr>
<td>DFHOTCO</td>
<td>Handles requests on the OTCO gate.</td>
</tr>
<tr>
<td>DFHOTDUF</td>
<td>OT domain offline dump formatting routine</td>
</tr>
<tr>
<td>DFHOTTRI</td>
<td>Interprets OT domain trace entries</td>
</tr>
</tbody>
</table>

Exits

None

Trace

The point IDs for the OT domain are of the form OTxxxx; the corresponding trace levels are OT 1, OT 2 and Exc.

For more information about the trace points, see the [CICS User’s Handbook](https://example.com). For more information about using traces in problem determination, see the [CICS Problem Determination Guide](https://example.com).
Chapter 62. Parameter manager domain (PA)

The parameter manager domain (also sometimes known simply as “parameter manager”) provides a facility to inform CICS domains of system parameters during CICS initialization. These system initialization parameters are specified in the system initialization table (SIT), and as temporary override parameters read from the SYSIN data stream or specified interactively at the system console.

The parameter manager domain also provides an operator correction facility for incorrectly specified system initialization parameter keywords early in CICS initialization. To use this facility, the user must specify the PARMERR system initialization parameter.

Parameter manager domain’s specific gate

Table 77 summarizes the parameter manager domain’s specific gate. It shows the level-1 trace point IDs of the modules providing the functions for the gate, the functions provided by the gate, and whether or not the functions are available through the exit programming interface (XPI).

Table 77. Parameter manager domain’s specific gate

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>XPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAGP</td>
<td>PA 0101</td>
<td>FORCE_START</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>PA 0102</td>
<td>GET_PARAMETERS</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INQUIRE_START</td>
<td>NO</td>
</tr>
</tbody>
</table>

PAGP gate, FORCE_START function

The FORCE_START function of the PAGP gate is used to override the type of start requested by the START system initialization parameter. It is currently used to force START=AUTO if the MVS automatic restart manager indicates that CICS is being automatically restarted with the original startup JCL (so that CICS does not get a COLD start that the original JCL might have asked for).

Input parameters

START_TYPE
specifies the type of CICS start to be forced. It can have either of these values:
- COLD
- AUTO

Output parameters

RESPONSE
is the parameter manager domain’s response to the call. It can have any one of these values:
- OK
- EXCEPTION
- DISASTER

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NOT_POSSIBLE</td>
</tr>
</tbody>
</table>
Parameter manager domain (PA)

**PAGP gate, GET_PARAMETERS function**

The GET_PARAMETERS function of the PAGP gate is used to get the initialization parameters for a requesting domain.

**Input parameters**

FORCE_ALL

specifies whether all parameters are required, even on a non-cold start. It can have either of these values:

YES|NO

**Output parameters**

PARAMETERS_TRANSFERRED

indicates to the calling domain whether any system parameters were transferred successfully by the parameter manager domain. It can have either of these values:

YES|NO

RESPONSE

is the parameter manager domain’s response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER

**PAGP gate, INQUIRE_START function**

The INQUIRE_START function of the PAGP gate is used to find out the type of start that CICS is to perform. This information is used to determine whether domains need to perform a cold or warm start.

**Input parameters**

None.

**Output parameters**

START

specifies the type of start CICS is to perform. It can have any one of these values:

COLD|WARM|LOGTERM

RESPONSE

is the parameter manager domain’s response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER

---

**Parameter manager domain’s generic gate**

Table 78 summarizes the parameter manager domain’s generic gate. It shows the level-1 trace point IDs of the modules providing the functions for the gate, the functions provided by the gate, and the generic format for calls to the gate.

**Table 78. Parameter manager domain’s generic gate**

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMDM</td>
<td>PA 0201</td>
<td>PRE_INITIALIZE</td>
<td>DMDM</td>
</tr>
<tr>
<td></td>
<td>PA 0202</td>
<td>INITIALIZERE_DOMAIN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>QUIESCE_DOMAIN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TERMINATE_DOMAIN</td>
<td></td>
</tr>
</tbody>
</table>
Parameter manager domain (PA)

You can find descriptions of these functions and their input and output parameters, in the section dealing with the corresponding generic format, in "Domain manager domain’s generic formats” on page 372.

In preinitialization processing, the parameter manager domain reads system initialization (override) parameters from the startup job stream and, if requested, from the SYSIN data set and the console.

If a system initialization table (SIT) has been specified, that is loaded into storage. Otherwise, the default SIT is loaded. The override parameters are applied to the SIT, and related parameters are checked for consistency. Errors are reported, but no action is taken.

The parameter manager domain also provides services to other domains as they preinitialize. It informs them of the type of start (cold or auto), and supplies information as required from the SIT.

In initialization processing, the parameter manager domain waits for all the other domains to complete their initialization, and then writes a warm start record to the catalog.

The parameter manager domain does no quiesce processing or termination processing.

**Modules**

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHPADM</td>
<td>Parameter manager domain initialization and termination</td>
</tr>
<tr>
<td>DFHPADUF</td>
<td>An offline routine to format system dump information</td>
</tr>
<tr>
<td>DFHPAGP</td>
<td>Passes initialization parameters to domains requesting GET_PARAMETERS</td>
</tr>
<tr>
<td>DFHPAIO</td>
<td>Communicates with the SYSIN data set and operator console</td>
</tr>
<tr>
<td>DFHPASY</td>
<td>System initialization override parameter checker and syntax parser</td>
</tr>
<tr>
<td>DFHPATRI</td>
<td>An offline routine to format trace points</td>
</tr>
</tbody>
</table>

**Exits**

No global user exit points are provided in this domain.

**Trace**

The point IDs for the parameter manager domain are of the form PA xxxx; the corresponding trace levels are PA 1, PA 2, and Exc.

For more information about the trace points, see the CICS Trace Entries. For more information about using traces in problem determination, see the CICS Problem Determination Guide.
Chapter 63. Partner resource manager

The partner resource manager (an OCO component of the AP domain) is responsible for managing all operations involving the partner resource table (PRT). A PARTNER definition is required for every remote partner referenced in SAA communications interface calls (see "Chapter 73. SAA Communications and Resource Recovery interfaces" on page 1001). Partner resources are installed either at system initialization or using CEDA INSTALL, and can be discarded using either the CEMT transaction or EXEC CICS commands.

The partner resource manager is implemented as a set of subroutine interfaces.

Functions provided by the partner resource manager

Table 79 summarizes the external subroutine interfaces provided by the partner resource manager. It shows the subroutine call formats, the level-1 trace point IDs of the modules providing the functions for these formats, and the functions provided.

<table>
<thead>
<tr>
<th>Format</th>
<th>Trace</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRCM</td>
<td>AP 0F36</td>
<td>INQUIRE_PARTNER</td>
</tr>
<tr>
<td></td>
<td>AP 0F37</td>
<td>START_PARTNER_BROWSE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GET_NEXT_PARTNER</td>
</tr>
<tr>
<td></td>
<td></td>
<td>END_PARTNER_BROWSE</td>
</tr>
<tr>
<td>PRFS</td>
<td>AP 0F34</td>
<td>LOCATE_AND_LOCK_PARTNER</td>
</tr>
<tr>
<td></td>
<td>AP 0F35</td>
<td></td>
</tr>
<tr>
<td>PRIN</td>
<td>AP 0F20</td>
<td>START_INIT</td>
</tr>
<tr>
<td></td>
<td>AP 0F21</td>
<td>COMPLETE_INIT</td>
</tr>
<tr>
<td>PRPT</td>
<td>AP 0F30</td>
<td>ADD_REPLACE_PARTNER</td>
</tr>
<tr>
<td></td>
<td>AP 0F31</td>
<td>DELETE_PARTNER</td>
</tr>
</tbody>
</table>

PRCM format, INQUIRE_PARTNER function

The INQUIRE_PARTNER function of the PRCM format is used to retrieve the installed definition of a specified partner, consisting of the remote transaction program name (TP name), network identifier, netname (network LU name), and profile name.

Input parameters

PARTNER_NAME
is the 8-character name of the entry whose contents are to be retrieved.

TP_NAME
is a buffer for the output TP name.

Output parameters

NETWORK
is the 8-character network identifier.

NETNAME
is the 8-character netname.
Partner resource manager

**PROFILE_NAME**
- is the 8-character CICS profile name.

**RESPONSE**
- is the subroutine’s response to the call. It can have any of these values:
  - OK
  - EXCEPTION
  - KERNERROR

**[REASON]**
- is returned when RESPONSE is EXCEPTION. It has this value:
  - PARTNER_NOT_FOUND

**PRCM format, START_PARTNER_BROWSE function**

The START_PARTNER_BROWSE function of the PRCM format is used to initiate a browse of the partner resource table. The browse starts at the beginning of the table.

**Input parameters**
- None.

**Output parameters**

**BROWSE_TOKEN**
- is the token identifying the browse session initiated by this call.

**RESPONSE**
- is the subroutine’s response to the call. It can have any of these values:
  - OK
  - DISASTER
  - KERNERROR
  - PURGED

**[REASON]**
- is returned when RESPONSE is DISASTER. It has this value:
  - GETMAIN_FAILED

**PRCM format, GET_NEXT_PARTNER function**

The GET_NEXT_PARTNER function of the PRCM format is used to retrieve the information stored in the next partner found in a sequential browse of the partner resource table.

**Input parameters**

**BROWSE_TOKEN**
- is the token identifying this browse session.

**TP_NAME**
- is a buffer for the output TP name.

**Output parameters**

**PARTNER_NAME**
- is the 8-character name of the entry retrieved.

**NETWORK**
- is the 8-character network identifier.

**NETNAME**
- is the 8-character netname.

**PROFILE_NAME**
- is the 8-character CICS profile name.

**RESPONSE**
- is the subroutine’s response to the call. It can have any of these values:
OK|EXCEPTION|KERNERROR

[REASON]
is returned when RESPONSE is EXCEPTION. It has this value:
END_OF_LIST

**PRCM format, END_PARTNER_BROWSE function**

The END_PARTNER_BROWSE function of the PRCM format is used to terminate a browse of the partner resource table.

**Input parameters**

BROWSE_TOKEN
is the token identifying this browse session.

**Output parameters**

RESPONSE
is the subroutine’s response to the call. It can have either of these values:
OK|KERNERROR

**PRFS format, LOCATE_AND_LOCK_PARTNER function**

The LOCATE_AND_LOCK_PARTNER function of the PRFS format is used to retrieve the information stored in a named entry in the partner resource table. A table manager read lock is applied to the entry.

**Input parameters**

PARTNER_NAME
is the 8-character name of the entry whose contents are to be retrieved.

TP_NAME
is a buffer for the output TP name.

**Output parameters**

NETWORK
is the 8-character network identifier.

NETNAME
is the 8-character netname.

PROFILE_NAME
is the 8-character CICS profile name.

RESPONSE
is the subroutine’s response to the call. It can have any of these values:
OK|EXCEPTION|KERNERROR

[REASON]
is returned when RESPONSE is EXCEPTION. It has this value:
PARTNER_NOT_FOUND

**PRIN format, START_INIT function**

The START_INIT function of the PRIN format is used to attach a CICS task to perform initialization of the partner resource manager.

**Input parameters**

None.
Partner resource manager

Output parameters

RESPONSE
is the subroutine’s response to the call. It can have any of these values:
OK|DISASTER|KERNERROR

[REASON]
is returned when RESPONSE is DISASTER. It can have either of these values:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>GETMAIN-FAILED, ADD_SUSPEND_FAILED</td>
</tr>
</tbody>
</table>

PRIN format, COMPLETE_INIT function

The COMPLETE_INIT function of the PRIN format is used to wait for the initialization task attached by the START_INIT function to complete processing.

Input parameters
None.

Output parameters

RESPONSE
is the subroutine’s response to the call. It can have any of these values:
OK|DISASTER|KERNERROR

[REASON]
is returned when RESPONSE is DISASTER. It has this value:
INIT_TASK_FAILED

PRPT format, ADD_REPLACE_PARTNER function

The ADD_REPLACE_PARTNER function of PRPT format is used to add a named entry to the partner resource table. The new entry replaces the existing entry (if any) with the specified name.

Input parameters

PARTNER_NAME
is the 8-character name of the entry whose contents are to be added or replaced.

NETWORK
is the 8-character network identifier.

NETNAME
is the 8-character netname.

PROFILE_NAME
is the 8-character CICS profile name.

TP_NAME
specifies the address and length of a buffer containing the TP name.

SYSTEM_STATUS
specifies the status of the CICS system at the time of the call. It can have any one of these values (ONLINE here means during execution):
COLD_START|WARM_START|ONLINE
Output parameters

RESPONSE is the subroutine’s response to the call. It can have any of these values:
OK | EXCEPTION | DISASTER | KERNERROR | PURGED

[REASON] is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>CATALOG_WRITE_FAILED, GETMAIN_FAILED</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>PARTNER_IN_USE</td>
</tr>
</tbody>
</table>

PRPT format, DELETE_PARTNER function

The DELETE_PARTNER function of the PRPT format is used to delete a named entry in the partner resource table.

Input parameters

PARTNER_NAME is the 8-character name of the entry to be deleted.

SYSTEM_STATUS is the status of the CICS system at the time of the call. It can have any one of these values (ONLINE here means during execution):
COLD_START | WARM_START | ONLINE

Output parameters

RESPONSE is the subroutine’s response to the call. It can have any of these values:
OK | EXCEPTION | DISASTER | KERNERROR

[REASON] is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>CATALOG_DELETE_FAILED</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>PARTNER_IN_USE, PARTNER_NOT_FOUND</td>
</tr>
</tbody>
</table>

Modules

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHAPTRR</td>
<td>Interprets partner resource manager trace entries</td>
</tr>
<tr>
<td>DFHPRCM</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>INQUIRE_PARTNER</td>
</tr>
<tr>
<td></td>
<td>START_PARTNER_BROWSE</td>
</tr>
<tr>
<td></td>
<td>GET_NEXT_PARTNER</td>
</tr>
<tr>
<td></td>
<td>END_PARTNER_BROWSE</td>
</tr>
<tr>
<td>DFHPRDUF</td>
<td>Formats the partner resource manager control blocks in a CICS system dump</td>
</tr>
</tbody>
</table>
Partner resource manager

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHPRFS</td>
<td>Handles the following request:</td>
</tr>
<tr>
<td></td>
<td>LOCATE_AND_LOCK_PARTNER</td>
</tr>
<tr>
<td>DFHPRIN1</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>START_INIT</td>
</tr>
<tr>
<td></td>
<td>COMPLETE_INIT</td>
</tr>
<tr>
<td>DFHPRIN2</td>
<td>Runs as a CICS task to perform initialization of the partner resource</td>
</tr>
<tr>
<td></td>
<td>manager</td>
</tr>
<tr>
<td>DFHPRPT</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>ADD_REPLACE_PARTNER</td>
</tr>
<tr>
<td></td>
<td>DELETE_PARTNER</td>
</tr>
<tr>
<td>DFHPRRP</td>
<td>Initializes the partner resource table at CICS startup</td>
</tr>
</tbody>
</table>

Exits

No global user exit points are provided for this component.

Trace

The following point ID is provided for the partner resource manager:
- AP 0F20 through AP 0F3F, for which the trace levels are AP 1 and Exc.

For more information about the trace points, see the CICS Trace Entries. For more information about using traces in problem determination, see the CICS Problem Determination Guide.
Chapter 64. Program control

The program control program, DFHPCP, is an interface routine which supports DFHPC LINK, ABEND, SETXIT and RESETXIT calls issued in other CICS modules and invokes the appropriate program manager domain function.

In previous releases DFHPCP provided the functions that are now provided by the Program Manager Domain, and other domains.

Design overview

Services in response to requests

The following services are performed by DFHPCP in response to DFHPC requests from other CICS functions, where those functions have not been converted to use domain interfaces:

Link (LINK)
Builds a parameter list and issues DFHPGLK FUNCTION(LINK) domain call.

Handle Abend (SETXIT)
If SETXIT macro specifies an abend routine address, then DFHPCP builds a parameter list and issues a DFHPGHM FUNCTION(SET_ABEND) OPERATION(HANDLE) call. If SETXIT macro does not specify an abend routine address, then DFHPCP builds a parameter list and issues a DFHPGHM FUNCTION(SET_ABEND) OPERATION(CANCEL) call.

RESETXIT
DFHPCP builds a parameter list and issues a DFHPGHM FUNCTION(SET_ABEND) OPERATION(RESET) call. If SETXIT macro does not specify an abend routine address, then DFHPCP builds a parameter list and issues a DFHPGHM CANCEL call.

Abend (ABEND)
If it is an ABEND request without an existing TACB, then the parameter list is built for this abend. A DFHABAB(CREATE_ABEND_RECORD) is issued to build the TACB. Else a DFHABAB(UPDATE_ABEND_RECORD) is issued with the name of the failing program is issued. A DFHABAB(START_ABEND) call is then made to issue the abend. If the DFHABAB(START_ABEND) call returns control to this module, it is because the exit XPCTA has been invoked and modified the return address. Control is passed to the modified address in the requested execution key.

Modules

DFHEPC

Call mechanism
Branched to from DFHEIP.

Entry address
DFHEPCNA. Stored in the CSA in a field named CSAEPC.
Program control

Purpose
DFHEPC is DFHEIP’s program control interface. It supports the following EXEC CICS requests
- LINK
- XCTL
- RETURN
- LOAD
- RELEASE
- ABEND
- HANDLE ABEND

It routes a local request to the PG domain, or to DFHABAB (EXEC CICS ABEND)
It routes a remote EXEC CICS LINK request to the intersystem module, DFHISP.

Called by
DFHEPC is called exclusively by DFHEIP.

Inputs
The application parameter list.

Outputs
Updated EIB.

Operation

LINK
If SYSID is remote, ships the link request through the DFHISP module.
If SYSID is local:
- Builds parameter list and calls DFHPGLE FUNCTION(LINK_EXEC)
- Checks the response.
- If response indicates the program is remote, ships the link request through the DFHISP module.
- Sets up EIBRESP (and, if needed, EIBRESP2).
- Returns control to DFHEIP.

XCTL
Builds parameter list and calls DFHPGXE FUNCTION(PREPARE_XCTL_EXEC)
Checks the response
Sets up EIBRESP (and, if needed, EIBRESP2).
If the PGXE request failed, then returns control to DFHEIP
If the PGXE request was successful, then return control to DFHAPLI as for EXEC CICS RETURN. (DFHAPLI will then invoke the program specified on EXEC CICS XCTL).

RETURN
Builds parameter list and calls DFHPGRE FUNCTION(PREPARE_RETURN_EXEC) (this call is bypassed if there are no options (COMMAREA, TRANSID, INPUTMSG) specified on EXEC CICS RETURN
- Checks the response
- Sets up EIBRESP (and, if needed, EIBRESP2).
- If the PGRE request failed, then returns control to DFHEIP
Program control

. If the PGRE request was successful (or was bypassed), then return control to DFHAPLI which completes the return to the calling program or to Transaction Manager.

LOAD
Builds parameter list and calls DFHPGLD FUNCTION(LOAD_EXEC)
Checks the response
Sets up EIBRESP (and, if needed, EIBRESP2).
If the PGLD request was successful, then set the return parameters in the application parameter list.
Returns control to DFHEIP.

RELEASE
Builds parameter list and calls DFHPGLD FUNCTION(RELEASE_EXEC)
Checks the response
Sets up EIBRESP (and, if needed, EIBRESP2).
Returns control to DFHEIP.

HANDLE ABEND
For HANDLE ABEND PROGRAM, perform resource security check and check whether program name is known.
Builds parameter list and calls DFHPGHM FUNCTION(SET_ABEND)
• OPERATION(HANDLE) for HANDLE ABEND PROGRAM or LABEL
• OPERATION(CANCEL) for HANDLE ABEND CANCEL
• OPERATION(RESET) for HANDLE ABEND
Checks the response
Sets up EIBRESP (and, if needed, EIBRESP2).
Returns control to DFHEIP.

ABEND
Builds parameter list and calls DFHABAB FUNCTION(CREATE_ABEND_RECORD) and FUNCTION(START_ABEND).
DFHABAB START_ABEND does not normally return, as control is passed to a program or label specified on a HANDLE ABEND, or the program is terminated abnormally.
The XPCTA user exit can request retry. In this case DFHABAB START_ABEND returns to DFHEPC passing back the retry parameters. DFHEPC sets the registers and other values and branches to the specified retry address.

How loaded
At CICS startup, as part of the building of the CICS nucleus. The nucleus is built by DFHSIB1, which uses its nucleus build list to determine the content and characteristics of the CICS nucleus.
Program control

Exits

There are two global user exit points in DFHEPC: XPCREQ and XPCREQC.

There are two global user exit points in DFHABAB: XPCABND and XPCTA.

There are two global user exit points in DFHAPLI1: XPCFTCH and XPCHAIR.

There is one global user exit point in DFHERM: XPCHAIR.

There is one global user exit point in DFHUEH: XPCHAIR.

For further information, see the CICS Customization Guide.

Trace

The following point IDs are provided for entry to and exit from DFHPCPG:
• AP 2000, for which the trace level is PC 1
• AP 2001, for which the trace level is PC 1.

For more information about the trace points, see the CICS Trace Entries. For more information about using traces in problem determination, see the CICS Problem Determination Guide.
Chapter 65. Program error program

CICS provides a dummy program error program (DFHPEP) that does nothing except give control back to the abnormal condition program (DFHACP), which is invoked during transaction abend processing.

You can provide some additional routines to handle programming errors. For instance, it is possible to disable the transaction code associated with the program in error, thus preventing the recurrence of the error until it can be corrected; send messages to the end-user terminal; initiate a new transaction; or record abend information in transient data.

Design overview

To provide corrective action in response to a programming error, you can code a program error program (DFHPEP). This program can then be assembled and link-edited to replace the dummy DFHPEP.

If provided, this program is invoked by the abnormal condition program (DFHACP) whenever a task terminates due to a task abnormal condition. However, it will NOT be called if a task is terminated due to an attach failure (for example the transaction is not defined) or when CICS deliberately terminates a task to alleviate a stall.

The user can perform any kind of corrective action within a program error program.

Guidance information about PEP coding is given in the CICS Recovery and Restart Guide. Reference information about PEP coding is given in the CICS Customization Guide.

Control blocks

The control block associated with the program error program is:
DFHPEP_COMMAREA, the commarea passed to DFHPEP.

See the CICS Data Areas manual for a detailed description of this control block.

Modules

DFHPEP

Exits

No global user exit points are provided for this function.

Trace

No trace points are provided for this function.
Chapter 66. Program manager domain (PG)

The program manager domain provides support for the following areas of CICS:

- Program control functions; EXEC CICS LINK, XCTL, LOAD, RELEASE, and RETURN
- Transaction ABEND and condition handling functions; EXEC CICS ABEND, HANDLE ABEND, HANDLE CONDITION and HANDLE AID
- Related functions such as invoking user-replaceable modules, global user exits, and task-related user exits
- Autoinstall for programs, mapsets, and partitionsets.

Program manager domain’s specific gates

Table 80 summarizes the program manager domain’s specific gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and whether or not the functions are available through the exit programming interface (XPI).

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>XPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>PGAI</td>
<td>PG 0E01</td>
<td>Kernel subroutine called internally from program manager</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>PG 0E02</td>
<td></td>
<td>NO</td>
</tr>
<tr>
<td>PGAQ</td>
<td>PG 0401</td>
<td>INQUIRE_AUTOINSTALL</td>
<td>YES</td>
</tr>
<tr>
<td></td>
<td>PG 0402</td>
<td>SET_AUTOINSTALL</td>
<td>YES</td>
</tr>
<tr>
<td>PGDD</td>
<td>PG 0301</td>
<td>DEFINE_PROGRAM</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>PG 0302</td>
<td>DELETE_PROGRAM</td>
<td>NO</td>
</tr>
<tr>
<td>PGEX</td>
<td>PG 0C01</td>
<td>INITIALIZE_EXIT</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>PG 0C02</td>
<td>TERMINATE_EXIT</td>
<td>NO</td>
</tr>
<tr>
<td>PGHM</td>
<td>PG 0700</td>
<td>SET_CONDITIONS</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>PG 0701</td>
<td>IGNORE_CONDITIONS</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INQ_CONDITION</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SET_AIDS</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INQ_AID</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SET_ABEND</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INQ_ABEND</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PUSH_HANDLE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>POP_HANDLE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FREE_HANDLE_TABLES</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CLEAR_LABELS</td>
<td>NO</td>
</tr>
<tr>
<td>PGIS</td>
<td>PG 0500</td>
<td>INQUIRE_PROGRAM</td>
<td>YES</td>
</tr>
<tr>
<td></td>
<td>PG 0501</td>
<td>INQUIRE_CURRENT_PROGRAM</td>
<td>YES</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SET_PROGRAM</td>
<td>YES</td>
</tr>
<tr>
<td></td>
<td></td>
<td>START_BROWSE_PROGRAM</td>
<td>YES</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GET_NEXT_PROGRAM</td>
<td>YES</td>
</tr>
<tr>
<td></td>
<td></td>
<td>END_BROWSE_PROGRAM</td>
<td>YES</td>
</tr>
<tr>
<td></td>
<td></td>
<td>REFRESH_PROGRAM</td>
<td>NO</td>
</tr>
<tr>
<td>PGLD</td>
<td>PG 0601</td>
<td>LOAD_EXEC</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>PG 0602</td>
<td>LOAD</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RELEASE_EXEC</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RELEASE</td>
<td>NO</td>
</tr>
</tbody>
</table>
Program manager domain (PG)

Table 80. Program manager domain’s specific gates (continued)

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>XPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>PGLE</td>
<td>PG 1101</td>
<td>LINK_EXEC</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>PG 1102</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PGLK</td>
<td>PG 0B01</td>
<td>LINK</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>PG 0B02</td>
<td>LINK_PLT</td>
<td>NO</td>
</tr>
<tr>
<td>PGLU</td>
<td>PG 0A01</td>
<td>LINK_URM</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>PG 0A02</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PGPG</td>
<td>PG 0901</td>
<td>INITIAL_LINK</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>PG 0902</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PGRE</td>
<td>PG 1201</td>
<td>PREPARE_RETURN_EXEC</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>PG 1202</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PGXE</td>
<td>PG 1301</td>
<td>PREPARE_XCTL_EXEC</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>PG 1302</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PGXM</td>
<td>PG 0901</td>
<td>INITIALIZE_TRANSACTION</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>PG 0902</td>
<td>TERMINATE_TRANSACTION</td>
<td>NO</td>
</tr>
</tbody>
</table>

Note: PGRE is only called for EXEC RETURN statements which have input parameters (COMMAREA, INPUTMSG, or TRANSID) specified. If no input parameters are specified, there is no trace of PGRE after the EIP trace of the RETURN statement.

PGAQ gate, INQUIRE_AUTOINSTALL function

The INQUIRE_AUTOINSTALL function of the PGAQ gate is used to inquire about attributes of the program autoinstall function.

Input parameters
None.

Output parameters

[AUTOINSTALL_STATE] is the state of the program autoinstall function. It can have either of these values:

ACTIVE|INACTIVE

[AUTOINSTALL_CATALOG] identifies if program autoinstall events are cataloged. It can have any of these values:

MODIFY|NONE|ALL

[AUTOINSTALL_EXIT_NAME] is the name of the program autoinstall exit program.

RESPONSE is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION</td>
</tr>
</tbody>
</table>
PGAQ gate, SET_AUTOINSTALL function

The SET_AUTOINSTALL function of the PGAQ gate is used to set attributes of the program autoinstall function.

Input parameters

[AUTOINSTALL_STATE]
  is the state of the program autoinstall function. It can have either of these values:
  ACTIVE | INACTIVE

[AUTOINSTALL_CATALOG]
  identifies if program autoinstall events are cataloged. It can have any of these values:
  MODIFY | NONE | ALL

[AUTOINSTALL_EXIT_NAME]
  is the name of the program autoinstall exit program.

Output parameters

RESPONSE
  is the domain’s response to the call. It can have any of these values:
  OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON]
  is returned when RESPONSE is INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

PGDD gate, DEFINE_PROGRAM function

The DEFINE_PROGRAM function of the PGDD gate is used to define a program resource.

Input parameters

Note: Specify either the PROGRAM_NAME parameter or the CATALOG_ADDRESS parameter, not both.

PROGRAM_NAME
  is the name of the program resource to be defined.

CATALOG_ADDRESS
  is the token identifying the program resource to be defined.

[CEDF_STATUS]
  indicates whether or not the EDF diagnostic screens are displayed when the program is running under the control of the execution diagnostic facility (EDF). It can have either of these values:
  CEDF | NOCEDF

[LANGUAGE_DEFINED]
  is the language to be defined for the program. It can have any of these values:
  ASSEMBLER | C370 | COBOL | LE370 | PL1 | NOT_DEFINED
[AVAIL_STATUS] defines whether (ENABLED) or not (DISABLED) the program can be used. It can have either of these values: ENABLED|DISABLED

[MODULE_TYPE] is the type of program resource to be defined: It can have any of these values: PROGRAM|MAPSET|PARTITIONSET

[DATA_LOCATION] defines whether the program can handle only 24-bit addresses (data located below the 16MB line) can handle 31-bit addresses (data located above or below the 16MB line). The DATALLOCATION options are independent from the addressing mode of the link-edited program. It can have either of these values: ANY|BELOW

[EXECUTION_SET] indicates whether you want CICS to link to and run the program as if it were running in a remote CICS region (with or without the API restrictions of a DPL program). It can have either of these values: FULLAPI|DPLSUBSET

[REMOTE_PROGID] is the name by which the program is known in the remote CICS region. If you specify REMOTE_SYSID and omit REMOTE_PROGID, the REMOTE_PROGID parameter defaults to the same name as the local name (that is, the PROGRAM_NAME value).

[REMOTE_SYSID] is the name of a remote CICS region if you want CICS to ship a distributed program link (DPL) request to another CICS region.

[REMOTE_TRANID] is the name of the transaction you want the remote CICS to attach, and under which it is to run the remote program.

[EXECUTION_KEY] is the key in which CICS gives control to the program, and determines whether the program can modify CICS-key storage. It can have either of these values: CICS|USER

Note: If the program is link-edited with the RENT attribute and the RMODE(ANY) mode statement, CICS loads the program into extended the read-only DSA(ERDSA), regardless of the EXECKEY option. The ERDSA is allocated from read-only extended storage only if RENTPGM=PROTECT is specified as a system initialization parameter.

[PROGRAM_TYPE] is the type of program. It can have any of these values: PRIVATE|SHARED|TYPE_ANY

[PROGRAM_USAGE] defines whether the program is to be used as a CICS nucleus program or as a user application program. It can have either of these values: NUCLEUS|APPLICATION
defines the residence status of the program, and when the storage for this program is released. It can have any of these values:
RESIDENT|REUSABLE|TRANSIENT|RELOAD|TEST

is the addressing mode of the program. It can have any of these values:
24|31|AMODE_ANY

is the residence mode of the program. It can have any of these values:
24|RMODE_ANY

Output parameters
NEW_PROGRAM_TOKEN is the token assigned to program.
RESPONSE is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, CATALOG_NOT_OPERATIONAL, CATALOG_ERROR, INSUFFICIENT_STORAGE, LOCK_ERROR</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>PROGRAM_ALREADY_DEFINED, PROGRAM_IN_USE</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_CATALOG_ADDRESS, INVALID_FUNCTION, INVALID_MODE_COMBINATION, INVALID_PROGRAM_NAME, INVALID_TYPE_ATTRIB_COMBIN</td>
</tr>
</tbody>
</table>

PGDD gate, DELETE_PROGRAM function
The DELETE_PROGRAM function of the PGDD gate is used to delete a program resource.

Input parameters
PROGRAM_NAME is the name of the program resource to be defined.

Output parameters
NEW_PROGRAM_TOKEN is the token assigned to program.
RESPONSE is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOCK_ERROR</td>
</tr>
</tbody>
</table>
PGEX gate, INITIALIZE_EXIT function

The INITIALIZE_EXIT function of the PGEX gate is used to initialize an exit program.

Input parameters

PROGRAM_NAME
is the name, 1 through 8 alphanumeric characters, of the program to be initialized.

LOAD_PROGRAM
defines whether or not the program is to be loaded when initialized. It can have either of these values:
YES|NO

SYSTEM_AUTOINSTALL
defines whether CICS is to autoinstall the program if there is no associated PROGRAM resource definition. It can have either of these values:
YES|NO

[LPA_ELIGIBLE]
defines whether or not the program can be loaded into the MVS link pack area (LPA). It can have either of these values:
YES|NO

Output parameters

PROGRAM_TOKEN
is the token assigned to program.

[ENTRY_POINT]
is the token defining the entry point of the program.

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESPONSE</td>
<td>Possible REASON values</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>AUTOINSTALL_URM_FAILED, AUTOINSTALL_MODEL_NOT_DEF, AUTOINSTALL_INVALID_DATA, AUTOINSTALL_FAILED, PROGRAM_NOT_AUTHORIZED, PROGRAM_NOT_DEFINED, PROGRAM_NOT_ENABLED, PROGRAM_NOT_LOADABLE, REMOTE_PROGRAM</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_INITIALIZE_REQUEST, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>
**PGEX gate, TERMINATE_EXIT function**

The TERMINATE_EXIT function of the PGEX gate is used to terminate an exit program.

**Input parameters**

- **PROGRAM_TOKEN**
  is the token identifying the program to be terminated.

- **RELEASE_PROGRAM**
  defines whether or not the program is to be released when terminated. It can have either of these values: YES | NO

**Output parameters**

- **RESPONSE**
  is the domain’s response to the call. It can have any of these values: OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

  **[REASON]**
  is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>PROGRAM_NOT_AUTHORIZED, PROGRAM_NOT_DEFINED, PROGRAM_NOT_IN_USE, PROGRAM_NOT_ENABLED, PROGRAM_NOT_LOADABLE</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_PROGRAM_TOKEN, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

**PGHM gate, SET_CONDITIONS function**

The SET_CONDITIONS function of the PGHM gate is used to process for user EXEC CICS HANDLE CONDITION commands, and to save the details of the condition into the current condition handle table.

**Input parameters**

- **IDENTIFIERS**
  is the token identifying the conditions to be handled.

- **LABELS_FLAGS**
  is the token identifying the number of conditions in this command that have associated labels.

- **[LABELS]**
  is the token identifying the condition labels (the locations within the program to be branched to if the condition occurs).

- **[LANGUAGE]**
  is the program language. It can have any of these values: ASSEMBLER | C370 | COBOL | LE370 | PLI

- **[CURRENT_EXECUTION_KEY]**
  is an 8-bit value indicating the current program execution key (at the time the EXEC CICS HANDLE CONDITION command was issued).

- **[USERS_RSA_POINTER]**
  is the address of the user program Register Save Area into which the program’s registers are saved at each EXEC CICS command execution.
Program manager domain (PG)

Output parameters

[FASTPATH_FLAGS] identifies the fastpath flag settings for the following conditions handled by the user: RDATT, WRBRK, EOF, NOSPACE, QBUSY, NOSTG, ENQBUSY, NOJBUFSP, SIGNAL, OVERFLOW, SYSBUSY, SESSBUSY.

RESPONSE is the domain’s response to the call. It can have any of these values: OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION, MISSING_PARAMETER</td>
</tr>
</tbody>
</table>

PGHM gate, INQ_CONDITIONS function

The INQ_CONDITIONS function of the PGHM gate is invoked when a condition has occurred, and returns to the caller about details of the condition for user EXEC CICS HANDLE CONDITION commands.

Input parameters

CONDITION is an 8-bit value identifying the condition.

Output parameters

STATUS identifies the status of the condition. It can have any of these values: DEFAULT|HANDLED|IGNORED

[LABEL] is the token identifying the condition label within the program to be branched to if the condition occurs.

[LANGUAGE] is the program language. It can have any of these values: ASSEMBLER|C370|COBOL|LE370|PLI

[CURRENT_EXECUTION_KEY] is an 8-bit value indicating the current program execution key (at the time the EXEC CICS HANDLE CONDITION command was issued).

[USERS_RSA_POINTER] is the address of the user program Register Save Area into which the program’s registers are saved at each EXEC CICS command execution.

[PROGRAM_MASK] identifies the program mask at the time the HANDLE CONDITION command was executed.

[GOTOL] is the token identifying the condition label within the program to be branched to if the condition is ignored.
**Program manager domain (PG)**

**[ABEND_CODE]**

is the four-character abend code to be issued if CICS drives the system default, which is to abend the transaction.

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**[REASON]**

is returned when RESPONSE is DISASTER or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION, MISSING_PARAMETER</td>
</tr>
</tbody>
</table>

**PGHM gate, IGNORE_CONDITIONS function**

The IGNORE_CONDITIONS function of the PGHM gate is used to ignore the conditions for user EXEC CICS IGNORE CONDITION commands.

**Input parameters**

**IDENTIFIERS**

is the token identifying the conditions to be ignored.

**Output parameters**

**[FASTPATH_FLAGS]**

identifies the fastpath flag settings for the conditions.

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**[REASON]**

is returned when RESPONSE is DISASTER or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION, MISSING_PARAMETER</td>
</tr>
</tbody>
</table>

**PGHM gate, SET_AID function**

The SET_AID function of the PGHM gate is invoked in response to a user EXEC CICS HANDLE AID command, and saves the details of the handle into the current aid Handle Table.

**Input parameters**

**IDENTIFIERS**

is the token identifying the aids to be handled.

**LABELS_FLAGS**

is the token identifying the number of aids in this command that have associated labels.
Program manager domain (PG)

[LABELS]
is the token identifying the condition labels (the locations within the program to be branched to if the aid occurs).

[LANGUAGE]
is the program language. It can have any of these values:
ASSEMBLER|C370|COBOL|LE370|PLI

[CURRENT_EXECUTION_KEY]
is an 8-bit value indicating the current program execution key (at the time the EXEC CICS HANDLE AID command was issued).

[USERS_RSA_POINTER]
is the address of the user program Register Save Area into which the program’s registers are saved at each EXEC CICS command execution.

Output parameters

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION, MISSING_PARAMETER</td>
</tr>
</tbody>
</table>

PGHM gate, INQ_AID function

The INQ_AID function of the PGHM gate is invoked when an aid has occurred, and returns to the caller details of the handle aid for user EXEC CICS HANDLE AID commands.

Input parameters

AID    is an 8-bit value identifying the aid.

Output parameters

[LABEL]
is the token identifying the condition label within the program to be branched to if the aid occurs.

[LANGUAGE]
is the program language. It can have any of these values:
ASSEMBLER|C370|COBOL|LE370|PLI

[CURRENT_EXECUTION_KEY]
is an 8-bit value indicating the current program execution key (at the time the EXEC CICS HANDLE AID command was issued).

[USERS_RSA_POINTER]
is the address of the user program Register Save Area into which the program’s registers are saved at each EXEC CICS command execution.

[PROGRAM_MASK]
identifies the program mask at the time the HANDLE CONDITION command was executed.
is the token identifying the condition label within the program to be branched to if the condition is ignored.

is the four-character abend code to be issued if CICS drives the system default, which is to abend the transaction.

RESPONSE is the domain’s response to the call. It can have any of these values: OK, EXCEPTION, DISASTER, INVALID, KERNERROR, PURGED

is returned when RESPONSE is DISASTER or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION, MISSING_PARAMETER</td>
</tr>
</tbody>
</table>

PGHM gate, SET_ABEND function
The SET_ABEND function of the PGHM gate is invoked in response to a user EXEC CICS HANDLE ABEND command, and saves the details of the handle into the current abend Handle Table.

Input parameters

OPERATION identifies what is to be done if the abend occurs. It can have any of these values: HANDLE, CANCEL, RESET

Note: Specify either the LABEL parameter or the PROGRAM parameter, not both.

is the token identifying the condition label within the program to be branched to if the abend occurs.

is the name of the program to which control will be passed if the abend occurs.

is the program language. It can have any of these values: ASSEMBLER, C370, COBOL, LE370, PLI

is an 8-bit value indicating the current program execution key (at the time the EXEC CICS HANDLE ABEND command was issued).

is the address of the user program Register Save Area into which the program’s registers are saved at each EXEC CICS command execution.

Output parameters

RESPONSE is the domain’s response to the call. It can have any of these values:

Program manager domain (PG)

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION, MISSING_PARAMETER</td>
</tr>
</tbody>
</table>

PGHM gate, INQ_ABEND function

The INQ_ABEND function of the PGHM gate is invoked when an abend has occurred, and returns to the caller details of the handle abend for user EXEC CICS HANDLE AID commands.

Input parameters
None.

Output parameters

STATUS
identifies the status of the condition. It can have either of these values:
DEFAULT|HANDELED

[LABEL]
is the token identifying the condition label within the program branched to when the abend occurred.

[PROGRAM]
is the name of the program to which control was passed when the abend occurred.

[LANGUAGE]
is the program language. It can have any of these values:
ASSEMBLER|C370|COBOL|LE370|PLI

[CURRENT_EXECUTION_KEY]
is an 8-bit value indicating the current program execution key (at the time the EXEC CICS HANDLE AID command was issued).

[USERS_RSA_POINTER]
is the address of the user program Register Save Area into which the program’s registers are saved at each EXEC CICS command execution.

[PROGRAM_MASK]
identifies the program mask at the time the HANDLE CONDITION command was executed.

[GOTOL]
is the token identifying the condition label within the program to be branched to if the condition is ignored.

[HANDLE_COUNT]
is the number of times that this abend code has been handled.

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED
Program manager domain (PG)

[REASON]

is returned when RESPONSE is DISASTER or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION, MISSING_PARAMETER</td>
</tr>
</tbody>
</table>

PGHM gate, PUSH_HANDLE function

The PUSH_HANDLE function of the PGHM gate is invoked for a user EXEC CICS PUSH command.

**Input parameters**
None.

**Output parameters**

[FASTPATH_FLAGS]
identifies the fastpath flag settings for the conditions.

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION, MISSING_PARAMETER</td>
</tr>
</tbody>
</table>

PGHM gate, POP_HANDLE function

The POP_HANDLE function of the PGHM gate is invoked for a user EXEC CICS POP command.

**Input parameters**
None.

**Output parameters**

[FASTPATH_FLAGS]
identifies the fastpath flag settings for the conditions.

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>NO_PREVIOUS_PUSH</td>
</tr>
</tbody>
</table>
Program manager domain (PG)

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION, MISSING_PARAMETER</td>
</tr>
</tbody>
</table>

**PGHM gate, FREE_HANDLE_TABLES function**

The FREE_HANDLE_TABLES function of the PGHM gate is invoked by CICS during program termination processing and frees all storage relating to the Handle State for that program level.

**Input parameters**
No parameters.

**Output parameters**

RESPONSE is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON] is returned when RESPONSE is DISASTER or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION, MISSING_PARAMETER</td>
</tr>
</tbody>
</table>

**PGHM gate, CLEAR_LABELS function**

The CLEAR_LABELS function of the PGHM gate is invoked by CICS during XCTL processing and frees all storage relating to the Handle State for that program (except for the initial default state) and removes all user-defined label handles.

**Input parameters**
No parameters.

**Output parameters**

[FASTPATH_FLAGS] identifies the fastpath flag settings for the conditions.

RESPONSE is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON] is returned when RESPONSE is DISASTER or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION, MISSING_PARAMETER</td>
</tr>
</tbody>
</table>

**PGIS gate, INQUIRE_PROGRAM function**

The INQUIRE_PROGRAM function of the PGIS gate is used to inquire about attributes of a program.
Input parameters

Note: Specify either the PROGRAM_NAME parameter or the PROGRAM_TOKEN parameter, not both.

**PROGRAM_NAME**

is the name of the program.

**PROGRAM_TOKEN**

is the token identifying the program.

Output parameters

**[CEDF_STATUS]**

indicates whether or not the EDF diagnostic screens are displayed when the program is running under the control of the execution diagnostic facility (EDF). It can have any of these values:

CEDF|NOCEDF|NOT_APPLIC

**[HOLD_STATUS]**

is the hold status of the program (that is, how long the program is to be loaded). It can have any of these values:

TASK_LIFE|CICS_LIFE|NOT_APPLIC

**[LOAD_STATUS]**

is the load status of the program (that is, whether or not the program can be loaded). It can have any of these values:

LOADABLE|NOT_LOADABLE|NOT_LOADED|NOT_APPLIC

**[INSTALL_TYPE]**

is the method used to install the PROGRAM resource definition. It can have any of these values:

RDO|CATALOG|GROUPLIST|AUTO|SYSAUTO|MANUAL

**[LANGUAGE_DEFINED]**

is the language defined for the program. It can have any of these values:

ASSEMBLER|C370|COBOL|LE370|PLI|

NOT_DEFINED|NOT_APPLIC

**[LANGUAGE_DEDUCED]**

is the language deduced by CICS for the program. It can have any of these values:

ASSEMBLER|C370|COBOL|COBOL2|LE370|PLI|

NOT_DEDUCED|NOT_APPLIC

**[AVAIL_STATUS]**

defines whether (ENABLED) or not (DISABLED) the program can be used. It can have either of these values:

ENABLED|DISABLED

**[MODULE_TYPE]**

is the type of program resource to be defined: It can have any of these values:

PROGRAM|MAPSET|PARTITIONSET

**[DATALOCATION]**

defines whether the program can handle only 24-bit addresses (data located below the 16MB line) can handle 31-bit addresses (data located above or below the 16MB line). The DATALOCATION options are independent from the addressing mode of the link-edited program. It can have either of these values:
Program manager domain (PG)

ANY|BELOW|NOT_APPLIC

**[EXECUTION_SET]**
indicates whether you want CICS to link to and run the program as if it were running in a remote CICS region (with or without the API restrictions of a DPL program). It can have either of these values:
FULLAPI|DPLSUBSET|NOT_APPLIC

**[REMOTE_PROGID]**
is the name by which the program is known in the remote CICS region. If you specify REMOTE_SYSID and omit REMOTE_PROGID, the REMOTE_PROGID parameter defaults to the same name as the local name (that is, the PROGRAM_NAME value).

**[REMOTE_SYSID]**
is the name of a remote CICS region if you want CICS to ship a distributed program link (DPL) request to another CICS region.

**[REMOTE_TRANID]**
is the name of the transaction you want the remote CICS to attach, and under which it is to run the remote program.

**[EXECUTION_KEY]**
is the key in which CICS gives control to the program, and determines whether the program can modify CICS-key storage. It can have either of these values:
CICS|USER|NOT_APPLIC

*Note:* If the program is link-edited with the RENT attribute and the RMODE(ANY) mode statement, CICS loads the program into extended the read-only DSA(ERDSA), regardless of the EXECKEY option. The ERDSA is allocated from read-only extended storage only if RENTPGM=PROTECT is specified as a system initialization parameter.

**[PROGRAM_TYPE]**
is the type of program. It can have any of these values:
PRIVATE|SHARED|TYPE_ANY|NOT_APPLIC

**[PROGRAM_USAGE]**
defines whether the program is to be used as a CICS nucleus program or as a user application program. It can have either of these values:
NUCLEUS|APPLICATION

**[PROGRAM_ATTRIBUTE]**
defines the residence status of the program, and when the storage for this program is released. It can have any of these values:
RESIDENT|REUSABLE|TRANSIENT|RELOAD|TEST

**[SPECIFIED_AMODE]**
is the addressing mode of the program. It can have any of these values:
24|31|AMODE_ANY|AMODE_NOT_SPECIFIED

**[SPECIFIED_RMODE]**
is the residence mode of the program. It can have any of these values:
24|RMODE_ANY|RMODE_NOT_SPECIFIED

**[PROGRAM_LENGTH]**
is the length of the program. returned by the loader domain on the ACQUIRE_PROGRAM call.
[PROGRAM_USE_COUNT] is the number of times that the program has been used.

[PROGRAM_USER_COUNT] is the number of different users that have invoked the program.

[LOAD_POINT] is the load point address of the program returned by the loader domain on the ACQUIRE_PROGRAM call.

[ENTRY_POINT] is the entry point address of the program returned by the loader domain on the ACQUIRE_PROGRAM call.

[LOCATION] defines where the program resides. It can have any of these values: CDSA|ECDSA|SDSA|ESDSA|RDSA|ERDSA|LPA|ELPA|NONE

[ACCESS] is the type of access for the program. It can have any of these values: USER|CICS|READ_ONLY|NONE

RESPONSE is the domain’s response to the call. It can have any of these values: OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOCK_ERROR</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>PROGRAM_NOT_DEFINED_TO_LD, PROGRAM_NOT_DEFINED_TO_PG,</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_PROGRAM_TOKEN</td>
</tr>
</tbody>
</table>

PGIS gate, INQUIRE_CURRENT_PROGRAM function

The INQUIRE_CURRENT_PROGRAM function of the PGIS gate is used to inquire about the current attributes of a program (for the current invocation of the program).

Input parameters
None.

Output parameters

[CEDF_STATUS] indicates whether or not the EDF diagnostic screens are displayed when the program is running under the control of the execution diagnostic facility (EDF). It can have any of these values: CEDF|NOCEDF|NOT_APPLIC

[HOLD_STATUS] is the hold status of the program (that is, for how long the program is to be loaded). It can have any of these values: TASK_LIFE|CICS_LIFE|NOT_APPLIC
Program manager domain (PG)

[LOAD_STATUS]
is the load status of the program (that is, whether or not the program can be loaded). It can have any of these values:
LOADABLE|NOT_LOADABLE|NOT_LOADED|NOT_APPLIC

[INSTALL_TYPE]
is the method used to install the PROGRAM resource definition. It can have any of these values:
RDO|CATALOG|GROUPLIST|AUTO|SYSAUTO|MANUAL

[LANGUAGE_DEFINED]
is the language defined for the program. It can have any of these values:
ASSEMBLER|C370|COBOL|LE370|PLI
NOT_DEFINED|NOT_APPLIC

[LANGUAGE_DEDUCED]
is the language deduced by CICS for the program. It can have any of these values:
ASSEMBLER|C370|COBOL|COBOL2|LE370|PLI
NOT_DEDUCED|NOT_APPLIC

[AVAIL_STATUS]
defines whether (ENABLED) or not (DISABLED) the program can be used. It can have either of these values:
ENABLED|DISABLED

[MODULE_TYPE]
is the type of program resource to be defined: It can have any of these values:
PROGRAM|MAPSET|PARTITIONSET

[DATA_LOCATION]
defines whether the program can handle only 24-bit addresses (data located below the 16MB line) can handle 31-bit addresses (data located above or below the 16MB line). The DATALLOCATION options are independent from the addressing mode of the link-edited program. It can have either of these values:
ANY|BELOW|NOT_APPLIC

[EXECUTION_SET]
indicates whether you want CICS to link to and run the program as if it were running in a remote CICS region (with or without the API restrictions of a DPL program). It can have any of these values:
FULLAPI|DPLSUBSET|NOT_APPLIC

[REMOTE_DEFINITION]
defines whether the program is local or remote. It can have either of these values:
LOCAL|REMOTE

[REMOTE_PROGID]
is the name by which the program is known in the remote CICS region. If you specify REMOTE_SYSID and omit REMOTE_PROGID, the REMOTE_PROGID parameter defaults to the same name as the local name (that is, the PROGRAM_NAME value).

[REMOTE_SYSID]
is the name of a remote CICS region if you want CICS to ship a distributed program link (DPL) request to another CICS region.
is the name of the transaction you want the remote CICS to attach, and under which it is to run the remote program.

is the key in which CICS gives control to the program, and determines whether the program can modify CICS-key storage. It can have any of these values:

Note: If the program is link-edited with the RENT attribute and the RMODE(ANY) mode statement, CICS loads the program into extended the read-only DSA(ERDSA), regardless of the EXECKEY option. The ERDSA is allocated from read-only extended storage only if RENTPGM=PROTECT is specified as a system initialization parameter.

is the token assigned to program.

is the current name of the program.

is the name of the program invoking this program.

is the name of the program to which control will be returned when this program has ended.

indicates whether or not the EDF diagnostic screens are displayed when the program is running under the control of the execution diagnostic facility (EDF) It can have either of these values:

indicates whether the program is running with or without the API restrictions of a DPL program. It can have any of these values:

indicates the current environment in which the program is running. It can have any of these values:

is the addressing mode of the program. It can have either of these values:

is the current load point address of the program returned by the loader domain on the ACQUIRE_PROGRAM call.

is the current entry point address of the program returned by the loader domain on the ACQUIRE_PROGRAM call.

is the domain’s response to the call. It can have any of these values:
Program manager domain (PG)

[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOCK_ERROR</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>NO_CURRENT_PROGRAM</td>
</tr>
</tbody>
</table>

PGIS gate, SET_PROGRAM function

The SET_PROGRAM function of the PGIS gate is used to set the characteristics of a program when it is loaded.

**Input parameters**

Note: Specify either the PROGRAM_NAME parameter or the PROGRAM_TOKEN parameter, not both.

**PROGRAM_NAME**
is the name of the program.

**PROGRAM_TOKEN**
is the token identifying the program.

**[CEDF_STATUS]**
indicates whether or not the EDF diagnostic screens are displayed when the program is running under the control of the execution diagnostic facility (EDF). It can have either of these values:

CEDF|NOCEDF

**[AVAIL_STATUS]**
defines whether (ENABLED) or not (DISABLED) the program can be used. It can have either of these values:

ENABLED|DISABLED

**[EXECUTION_SET]**
indicates whether you want CICS to link to and run the program as if it were running in a remote CICS region (with or without the API restrictions of a DPL program). It can have either of these values:

FULLAPI|DPLSUBSET

**[EXECUTION_KEY]**
is the key in which CICS gives control to the program, and determines whether the program can modify CICS-key storage. It can have either of these values:

CICS|USER

Note: If the program is link-edited with the RENT attribute and the RMODE(ANY) mode statement, CICS loads the program into extended the read-only DSA(ERDSA), regardless of the EXECKEY option. The ERDSA is allocated from read-only extended storage only if RENTPGM=PROTECT is specified as a system initialization parameter.

**[PROGRAM_TYPE]**
is the type of program. It can have any of these values:

PRIVATE|SHARED|TYPE_ANY
defines whether the program is to be used as a CICS nucleus program or as a user application program. It can have either of these values:
NUCLEUS|APPLICATION

defines the residence status of the program, and when the storage for this program is released. It can have any of these values:
RESIDENT|REUSABLE|TRANSIENT|RELOAD|TEST

is the addressing mode of the program. It can have any of these values:
24|31|AMODE_ANY

is the residence mode of the program. It can have any of these values:
24|RMODE_ANY

is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, CATALOG_NOT_OPERATIONAL, CATALOG_ERROR, LOCK_ERROR</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>PROGRAM_NOT_DEFINED_TO_PG, CEDF_STATUS_NOT_FOR_REMOTE, CEDF_STATUS_NOT_FOR_MAPSET, CEDF_STATUS_NOT_FOR_PTNSET, EXEC_SET_NOT_FOR_REMOTE, EXEC_SET_NOT_FOR_MAPSET, EXEC_SET_NOT_FOR_PTNSET, EXEC_KEY_NOT_FOR_REMOTE, EXEC_KEY_NOT_FOR_MAPSET, EXEC_KEY_NOT_FOR_PTNSET, PROG_TYPE_NOT_FOR_REMOTE</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_MODE_COMBINATION, INVALID_PROGRAM_NAME, INVALID_PROGRAM_TOKEN, INVALID_TYPE_ATTRIB_COMBIN</td>
</tr>
</tbody>
</table>

The START_BROWSE_PROGRAM function of the PGIS gate is used to start browsing through program definitions, optionally starting at the given program definition.

is the optional name of the program definition at which you want to start browsing.

is a token identifying the program definition being browsed.

is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED
Program manager domain (PG)

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, INVALID_DIRECTORY, LOCK_ERROR</td>
</tr>
</tbody>
</table>

PGIS gate, GET_NEXT_PROGRAM function

The GET_NEXT_PROGRAM function of the PGIS gate is used to get the next program definition to be browse.

Input parameters

BROWSE_TOKEN
is a token identifying the program definition to be browsed.

Output parameters

[CEDF_STATUS]
indicates whether or not the EDF diagnostic screens are displayed when the program is running under the control of the execution diagnostic facility (EDF) It can have any of these values:
CEDF|NOEDF|NOT_APPLIC

[HOLD_STATUS]
is the hold status of the program (that is, for how long the program is to be loaded). It can have any of these values:
TASK_LIFE|CICS_LIFE|NOT_APPLIC

[LOAD_STATUS]
is the load status of the program (that is, whether or not the program can be loaded). It can have any of these values:
LOADABLE|NOT_LOADABLE|NOT_LOADED|NOT_APPLIC

[INSTALL_TYPE]
is the method used to install the PROGRAM resource definition. It can have any of these values:
RDO|CATALOG|GROUP|AUTO|SYSAUTO|MANUAL

[LANGUAGE_DEFINED]
is the language defined for the program. It can have any of these values:
ASSEMBLER|C370|COBOL|LE370|PLI|
NOT_DEFINED|NOT_APPLIC

[LANGUAGE_DEDUCED]
is the language deduced by CICS for the program. It can have any of these values:
ASSEMBLER|C370|COBOL|COBOL2|LE370|PLI|
NOT_DEDUCED|NOT_APPLIC

[AVAIL_STATUS]
defines whether (ENABLED) or not (DISABLED) the program can be used. It can have either of these values:
ENABLED|DISABLED

[MODULE_TYPE]
is the type of program resource to be defined: It can have any of these values:
PROGRAM|MAPSET|PARTITIONSET
defines whether the program can handle only 24-bit addresses (data located below the 16MB line) can handle 31-bit addresses (data located above or below the 16MB line). The DATALOCATION options are independent from the addressing mode of the link-edited program. It can have either of these values:

ANY|BELOW|NOT_APPLIC

indicates whether you want CICS to link to and run the program as if it were running in a remote CICS region (with or without the API restrictions of a DPL program). It can have either of these values:

FULLAPI|DPLSUBSET|NOT_APPLIC

is the name by which the program is known in the remote CICS region. If you specify REMOTE_SYSID and omit REMOTE_PROGID, the REMOTE_PROGID parameter defaults to the same name as the local name (that is, the PROGRAM_NAME value.

is the name of a remote CICS region if you want CICS to ship a distributed program link (DPL) request to another CICS region.

is the name of the transaction you want the remote CICS to attach, and under which it is to run the remote program.

is the key in which CICS gives control to the program, and determines whether the program can modify CICS-key storage. It can have either of these values:

CICS|USER|NOT_APPLIC

Note: If the program is link-edited with the RENT attribute and the RMODE(ANY) mode statement, CICS loads the program into extended the read-only DSA(ERDSA), regardless of the EXECKEY option. The ERDSA is allocated from read-only extended storage only if RENTPGM=PROTECT is specified as a system initialization parameter.

is the type of program. It can have any of these values:

PRIVATE|SHARED|TYPE_ANY|NOT_APPLIC

defines whether the program is to be used as a CICS nucleus program or as a user application program. It can have either of these values:

NUCLEUS|APPLICATION

defines the residence status of the program, and when the storage for this program is released. It can have any of these values:

RESIDENT|REUSABLE|TRANSIENT|RELOAD|TEST

is the addressing mode of the program. It can have any of these values:

24|31|AMODE_ANY|AMODE_NOT_SPECIFIED
Program manager domain (PG)

[SPECIFIED_RMODE]
is the residence mode of the program. It can have any of these values:
24|RMODE_ANY|RMODE_NOT_SPECIFIED

(PROGRAM_LENGTH)
is the length of the program. returned by the loader domain on the
ACQUIRE_PROGRAM call.

(PROGRAM_USE_COUNT)
is the number of times that the program has been used.

(PROGRAM_USER_COUNT)
is the number of different users that have invoked the program.

(LOAD_POINT)
is the load point address of the program returned by the loader domain on
the ACQUIRE_PROGRAM call.

(ENTRY_POINT)
is the entry point address of the program returned by the loader domain
on the ACQUIRE_PROGRAM call.

,LOCATION defines where the program resides. It can have any of these values:
CDSA|ECDSA|SDSA|ESDSA|RDSA|ERDSA|LPA|ELPA|NONE

(ACCESS)
is the type of access for the program. It can have any of these values:
USER|CICS|READ_ONLY|NONE

RESPONSE is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOCK_ERROR</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>INVALID_BROWSE_TOKEN, END_LIST</td>
</tr>
</tbody>
</table>

PGIS gate, END_BROWSE_PROGRAM function

The END_BROWSE_PROGRAM function of the PGIS gate is used to end browsing
through program definitions.

Input parameters

BROWSE_TOKEN
is a token identifying the last program definition that was browsed.

Output parameters

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:
PGIS gate, REFRESH_PROGRAM function

The REFRESH_PROGRAM function of the PGIS gate is used to inform the loader domain that a new copy of a named program is now available for use in the relocatable program library.

**Input parameters**

- **PROGRAM_NAME**
  - is the name of the program being refreshed.

**Output parameters**

- **VERSION**
  - is the version of the program after the REFRESH_PROGRAM function call.
  - It can have either of these values:
    - NEW
    - OLD
- **RESPONSE**
  - is the domain’s response to the call. It can have any of these values:
    - OK
    - EXCEPTION
    - DISASTER
    - INVALID
    - KERNELERROR
    - PURGED
- **[REASON]**
  - is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOCK_ERROR</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>INVALID_BROWSE_TOKEN, END_LIST</td>
</tr>
</tbody>
</table>

PGLD gate, LOAD_EXEC function

The LOAD_EXEC function of the PGLD gate is used to load a program in response to an EXEC CICS LOAD command.

**Input parameters**

- **PROGRAM_NAME**
  - is the name of the program being refreshed.

- **HOLD_LIFETIME**
  - determines for how long the program is to be loaded; that is, for the life-time of CICS (or until explicitly deleted) or for the lifetime of the task (unless explicitly deleted by the task). It can have either of these values:
    - CICS_LIFE
    - TASK_LIFE

**Output parameters**

- **LOAD_POINT**
  - is the current load point address of the program returned by the loader domain on the ACQUIRE_PROGRAM call.
Program manager domain (PG)

ENTRY_POINT
is the current entry point address of the program returned by the loader domain on the ACQUIRE_PROGRAM call.

PROGRAM_LENGTH
is the length of the program returned by the loader domain on the ACQUIRE_PROGRAM call.

RESPONSE
is the domain’s response to the call. It can have any of these values: OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

REASON
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>PROGRAM_NOT_DEFINED, PROGRAM_NOT_ENABLED, PROGRAM_NOT_LOADABLE, REMOTE_PROGRAM, NOT_AUTHORIZED, AUTOINSTALL_URM_FAILED, AUTOINSTALL_INVALID_DATA, AUTOINSTALL_MODEL_NOT_DEF, AUTOINSTALL_FAILED, NOT_INITIALIZED</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

PGLD gate, LOAD function

The LOAD function of the PGLD gate is used to load a program in response to a CICS internal load request.

Input parameters

PROGRAM_NAME
is the name of the program being refreshed.

HOLD_LIFETIME
determines for how long the program is to be loaded; that is, for the life-time of CICS (or until explicitly deleted) or for the lifetime of the task (unless explicitly deleted by the task). It can have either of these values: CICS_LIFE|TASK_LIFE

MODULE_TYPE
is the type of program to be loaded: It can have any of these values: PROGRAM|MAPSET|PARTITIONSET

SYSTEM_AUTOINSTALL
defines whether CICS is to autoinstall the program if there is no associated PROGRAM resource definition. It can have either of these values: YES|NO

LPA_ELIGIBLE
defines whether or not the program can be loaded into the MVS link pack area (LPA). It can have either of these values: YES|NO

Output parameters

LOAD_POINT
is the current load point address of the program returned by the loader domain on the ACQUIRE_PROGRAM call.
ENTRY_POINT
is the current entry point address of the program returned by the loader
domain on the ACQUIRE_PROGRAM call.

PROGRAM_LENGTH
is the length of the program returned by the loader domain on the
ACQUIRE_PROGRAM call.

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or INVALID. Possible values
are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>PROGRAM_NOT_DEFINED, PROGRAM_NOT_ENABLED, PROGRAM_NOT_LOADABLE, REMOTE_PROGRAM, AUTOINSTALL_URM_FAILED, AUTOINSTALL_INVALID_DATA, AUTOINSTALL_MODEL_NOT_DEF, AUTOINSTALL_FAILED</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

PGLD gate, RELEASE_EXEC function
The RELEASE_EXEC function of the PGLD gate is used to release a program in
response to an EXEC CICS RELEASE command.

Input parameters
PROGRAM_NAME
is the name of the program being released.

Output parameters
RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or INVALID. Possible values
are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NOT_AUTHORIZED, NOT_INITIALIZED, PROGRAM_NOT_DEFINED, PROGRAM_NOT_ENABLED, PROGRAM_NOT_IN_USE, PROGRAM_NOT_LOADED, PROGRAM_RELOAD_YES, RELEASE_ISSUING_PROGRAM, REMOTE_PROGRAM</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

PGLD gate, RELEASE function
The RELEASE function of the PGLD gate is used by CICS internal modules to
release a program in response previously loaded by a PGLD LOAD request.

Input parameters
PROGRAM_NAME
is the name of the program being released.
Program manager domain (PG)

**Output parameters**

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**[REASON]**

is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>PROGRAM_NOT_DEFINED, PROGRAM_NOT_ENABLED, PROGRAM_NOT_IN_USE, PROGRAM_NOT_LOADED, PROGRAM_RELOAD_YES, REMOTE_PROGRAM</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

**PGLE gate, LINK_EXEC function**

The LINK_EXEC function of the PGLE gate is used to link to a program in response to a user EXEC CICS LINK command.

**Input parameters**

**PROGRAM_NAME**

is the name of the program to be linked.

**[COMMAREA]**

is the optional communications area to be made available to the linked program.

**[HANDLE_ABEND_PGM]**

defines whether or not the program is to run as an abend handler program. It can have either of these values:

YES|NO

**[INPUTMSG]**

is a data area to be supplied to the linked program on its first execution of an EXEC CICS RECEIVE command.

**[SYNCONRETURN]**

defines whether or not a syncpoint is to be taken on return from the linked program. It can have either of these values:

YES|NO

**Output parameters**

**[REMOTE_PROGRAM_NAME]**

is the name by which the program is known in the remote CICS region. If you specify REMOTE_SYSID and omit REMOTE_PROGID, the REMOTE_PROGID parameter defaults to the same name as the local name (that is, the PROGRAM_NAME value).

**[REMOTE_SYSID]**

is the name of a remote CICS region if you want CICS to ship a distributed program link (DPL) request to another CICS region.

**[REMOTE_TRANID]**

is the name of the transaction you want the remote CICS to attach, and under which it is to run the remote program.
Program manager domain (PG)

[ABEND_CODE]

is the four-character abend code to be issued if there is an exception response with reason TRANSACTION_ABEND.

RESPONSE

is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]

is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>AUTOINSTALL_FAILED, AUTOINSTALL_INVALID_DATA, AUTOINSTALL_MODEL_NOT_DEF, AUTOINSTALL_URM_FAILED, DESTRUCTIVE_OVERLAP, INVALID_COMMAREA_ADDR, INVALID_COMMAREA_LEN, INVALID_INPUTMSG_LEN, INVALID_TERMINAL_TYPE, NOT_INITIALIZED, PROGRAM_NOTAUTHORIZED, PROGRAM_NOT_DEFINED, PROGRAM_NOT_ENABLED, PROGRAM_NOT_LOADABLE, REMOTE_PROGRAM, TRANSACTION_ABEND</td>
</tr>
</tbody>
</table>

PGLK gate, LINK function

The LINK function of the PGLK gate is used by CICS internal modules to link to a program.

Input parameters

PROGRAM_NAME

is the name of the program being linked.

SYSTEM_AUTOINSTALL

defines whether CICS is to autoinstall the program if there is no associated PROGRAM resource definition. It can have either of these values:

YES|NO

[LPA_ELIGIBLE]

defines whether or not the program can be loaded into the MVS link pack area (LPA). It can have either of these values:

YES|NO

[PARMLIST_PTR]

is the address of a parameter list passed by the CICS program initiating the PGLK link to the new program.

Output parameters

[ABEND_CODE]

is the four-character abend code to be issued if there is an exception response with reason TRANSACTION_ABEND.

RESPONSE

is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]

is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:
Program manager domain (PG)

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>PROGRAM_NOT_DEFINED, PROGRAM_NOT_ENABLED, PROGRAM_NOT_LOADABLE,</td>
</tr>
<tr>
<td></td>
<td>REMOTE_PROGRAM, AUTOINSTALL_URM_FAILED, AUTOINSTALL_MODEL_NOT_DEF,</td>
</tr>
<tr>
<td></td>
<td>AUTOINSTALL_INVALID_DATA, AUTOINSTALL_FAILED, TRANSACTION_ABEND</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

PGLK gate, LINK_PLT function

The LINK_PLT function of the PGLK gate is used by CICS internal modules to link to a program in the program list table.

**Input parameters**

**PROGRAM_NAME**

is the name of the program being linked.

**SYSTEM_AUTOINSTALL**

defines whether CICS is to autoinstall the program if there is no associated PROGRAM resource definition. It can have either of these values:

YES|NO

**[LPA_ELIGIBLE]**

defines whether or not the program can be loaded into the MVS link pack area (LPA). It can have either of these values:

YES|NO

**Output parameters**

**[ABEND_CODE]**

is the four-character abend code to be issued if there is an exception response with reason TRANSACTION_ABEND.

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**[REASON]**

is returned when RESPONSE is EXCEPTION or INVALID Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>PROGRAM_NOT_DEFINED, PROGRAM_NOT_ENABLED, PROGRAM_NOT_LOADABLE,</td>
</tr>
<tr>
<td></td>
<td>REMOTE_PROGRAM, AUTOINSTALL_URM_FAILED, AUTOINSTALL_MODEL_NOT_DEF,</td>
</tr>
<tr>
<td></td>
<td>AUTOINSTALL_INVALID_DATA, AUTOINSTALL_FAILED, TRANSACTION_ABEND</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

PGLU gate, LINK_URM function

The LINK_URM function of the PGLU gate is used by CICS internal modules to link to a user-replaceable module (program).

**Input parameters**

**PROGRAM_NAME**

is the name of the program to be linked.
SYSTEM_AUTOINSTALL
defines whether CICS is to autoinstall the program if there is no associated
PROGRAM resource definition. It can have either of these values:
YES | NO

[LPA_ELIGIBLE]
defines whether or not the program can be loaded into the MVS link pack
area (LPA). It can have either of these values:
YES | NO

[COMMAREA]
is the optional communications area to be made available to the linked
program.

Output parameters

[ABEND_CODE]
is the four-character abend code to be issued if there is an exception
response with reason URM_ABEND.

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON]
is returned when RESPONSE is EXCEPTION or INVALID Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>PROGRAM_NOT_DEFINED, PROGRAM_NOT_ENABLED, PROGRAM_NOT_LOADABLE,</td>
</tr>
<tr>
<td></td>
<td>REMOTE_PROGRAM, AUTOINSTALL_URM_FAILED, AUTOINSTALL_MODEL_NOT_DEF,</td>
</tr>
<tr>
<td></td>
<td>AUTOINSTALL_INVALID_DATA AUTOINSTALL_FAILED, INVALID_COMMAREA_LEN,</td>
</tr>
<tr>
<td></td>
<td>INVALID_COMMAREA_ADDR, AMODE_ERROR, URM_ABEND, DESTRUCTIVE_OVERLAP</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

PGPG gate, INITIAL_LINK function
The INITIAL_LINK function of the PGPG gate is used to link to the first program
of a transaction.

Input parameters

PROGRAM_NAME
is the name of the program being linked.

Output parameters

[ABEND_CODE]
is the four-character abend code to be issued if there is an exception
response with reason TRANSACTION_ABEND.

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:
**PGRE gate, PREPARE_RETURN function**

The PREPARE_RETURN function of the PGRE gate is used to process the communications area, inputmsg data, and transaction identifier from a user EXEC CICS RETURN command.

**Input parameters**

**[TRANSID]**

is the four-character transaction identifier.

**[COMMAREA]**

is the optional communications area made available to the linked program.

**[INPUTMSG]**

is a data area to be supplied to the linked program on its first execution of an EXEC CICS RECEIVE command.

**[IMMEDIATE]**

Indicates whether or not the transaction specified in TRANSID is to be attached as the next transaction regardless of any other transactions enqueued by ATI for this terminal. It can have either of these values: YES|NO

**Output parameters**

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**[REASON]**

is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>INVALID_COMMAREA_ADDR, INVALID_COMMAREA_LEN, INVALID_INPUTMSG_LEN, INVALID_TERMINAL_TYPE, INVALID_REQUEST_FROM_EXIT, INVALID_RETURN_REQUEST, NOT_INITIALIZED, NO_TERMINAL, TRANSID_NO_TERMINAL</td>
</tr>
</tbody>
</table>

**PGXE gate, PREPARE_XCTL_EXEC function**

The PREPARE_XCTL_EXEC function of the PGXE gate is used to process the communications area, inputmsg data, and transaction identifier from a user EXEC CICS XCTL command.

**Input parameters**

**PROGRAM_NAME**

is the name of the program to which control is to be passed.

**[COMMAREA]**

is the optional communications area made available to the linked program.
Program manager domain (PG)

[INPUTMSG]

is a data area to be supplied to the linked program on its first execution of an EXEC CICS RECEIVE command.

Output parameters

[ABEND_CODE]

is the four-character abend code to be issued if CICS drives the system default, which is to abend the transaction.

RESPONSE

is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON]

is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>AUTOINSTALL_FAILED, AUTOINSTALL_INVALID_DATA, AUTOINSTALL_MODEL_NOT_DEF, AUTOINSTALL_URM_FAILED, DESTRUCTIVE_OVERLAP, INVALID_COMMAREA_ADDR, INVALID_COMMAREA_LEN, INVALID_INPUTMSG_LEN, INVALID_TERMINAL_TYPE, INVALID_REQUEST_FROM_EXIT, NOT_INITIALIZED, PROGRAM_NOT_AUTHORIZED, PROGRAM_NOT_DEFINED, PROGRAM_NOT_ENABLED, PROGRAM_NOT_LOADABLE, REMOTE_PROGRAM, TRANSACTION_ABEND</td>
</tr>
</tbody>
</table>

PGXM gate, INITIALIZE_TRANSACTION function

The INITIALIZE_TRANSACTION function of the PGXM gate is used to initialize a transaction, and set up storage for the transaction.

Input parameters

None.

Output parameters

RESPONSE

is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON]

is returned when RESPONSE is INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

PGXM gate, TERMINATE_TRANSACTION function

The TERMINATE_TRANSACTION function of the PGXM gate is used to terminate a transaction, and clean up the transaction-related storage at task termination.

Input parameters

None.

Output parameters

RESPONSE

is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED
Program manager domain (PG)

[REASON]
is returned when RESPONSE is INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

Program manager domain’s generic gates

Table 81 summarizes the program manager domain’s generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 81. Program manager domain’s generic gates

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>PGDM</td>
<td>PG 0101</td>
<td>INITIALIZE_DOMAIN</td>
<td>DMDM</td>
</tr>
<tr>
<td></td>
<td>PG 0102</td>
<td>QUIESCE_DOMAIN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TERMATE_DOMAIN</td>
<td></td>
</tr>
<tr>
<td>PGST</td>
<td>PG 0F01</td>
<td>COLLECT_STATISTICS</td>
<td>STST</td>
</tr>
<tr>
<td></td>
<td>PG 0F02</td>
<td>COLLECT_RESOURCE_STATS</td>
<td></td>
</tr>
<tr>
<td>PGUE</td>
<td>PG 1001</td>
<td>SET_EXIT_STATUS</td>
<td>APUE</td>
</tr>
<tr>
<td></td>
<td>PG 1002</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For descriptions of these functions and their input and output parameters, refer to the sections dealing with the corresponding generic formats:

**Functions and parameters**

Format APUE—“Application domain’s generic formats” on page 100

Format DMDM—“Domain manager domain’s generic formats” on page 377

Format STST—“Statistics domain’s generic format” on page 1087

Initialize domain

There are two phases to initialization of the program manager domain:

1. The DFHPGDM module creates the PG domain anchor block, the PPT directory, and the PG Lock. It also adds subpools and gates, determines whether a cold, warm, or emergency start is needed, and waits for the global catalog to be available.

2. For a warm or emergency start, the DFHPGDM module rebuilds the PPT and restores the program autoinstall system initialization parameters from the global catalog entries. (It calls the parameter manager to obtain other system initialization parameter values.)

   For a cold start, the DFHPGDM module purges all the PPT entries from the global catalog.

Quiesce domain

In quiesce processing, the program manager domain:

- Sets the PG state to quiescing.
- Ensures that the statistics domain has gathered the PG statistics by issuing a WAIT_PHASE for STATISTICS_UNAVAILABLE.
This also ensures synchronization with the AP domain quiesce activity.

- Does *not* delete PG gates; PG functions remain available. However, use of programs after this point does not appear in statistics. (DFHSTP issues a PC LINK/ PGLK LINK to DFHWKP after AP domain waits for STATISTICS_UNAVAILABLE).
- Does *not* write PPT entries to the global catalog. (PPT entries are only written to the catalog when they are installed or changed.)
- (Finally) Sets the PG state to quiesced.

**Terminate domain**

In terminate processing, the program manager domain sets the PG state to terminated, and makes the program manager domain unavailable to EXEC CICS commands.

### Modules

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHPGAI</td>
<td>A kernel subroutine called internally from the Program Manager to support the autoinstall for programs function.</td>
</tr>
</tbody>
</table>
| DFHPGAQ | Handles the following requests:  
|         | INQUIRE_AUTOINSTALL  
|         | SET_AUTOINSTALL  
| DFHPGDD | Handles the following requests:  
|         | DEFINE_PROGRAM  
|         | DELETE_PROGRAM  
| DFHPGDM | Handles the following requests:  
|         | INITIALIZE_DOMAIN  
|         | QUIESCE_DOMAIN  
|         | TERMINATE_DOMAIN  
| DFHPGDUF | PG domain offline dump formatting routine |
| DFHPGEX | Handles the following requests:  
|         | INITIALIZE_EXIT  
|         | TERMINATE_EXIT  
| DFHPGHM | Handles the following requests:  
|         | SET_CONDITIONS  
|         | IGNORE_CONDITIONS  
|         | INQ_CONDITION  
|         | SET_AID5  
|         | INQ_AID  
|         | SET_ABEND  
|         | INQ_ABEND  
|         | PUSH_HANDLE  
|         | POP_HANDLE  
|         | FREE_HANDLE_TABLES  
|         | CLEAR_LABELS  
| DFHPGIS | Handles the following requests:  
|         | INQUIRE_PROGRAM  
|         | INQUIRE_CURRENT_PROGRAM  
|         | SET_PROGRAM  
|         | START_BROWSE_PROGRAM  
|         | GET_NEXT_PROGRAM  
|         | END_BROWSE_PROGRAM  
|         | REFRESH_PROGRAM  

Chapter 66. Program manager domain (PG)  915
Program manager domain (PG)

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHPGLD</td>
<td>Handles the following requests:</td>
<td>LOAD_EXEC, LOAD, RELEASE_EXEC, RELEASE</td>
</tr>
<tr>
<td>DFHPGLE</td>
<td>Handles the following requests:</td>
<td>LINK_EXEC</td>
</tr>
<tr>
<td>DFHPGLK</td>
<td>Handles the following requests:</td>
<td>LINK, LINK_PLT</td>
</tr>
<tr>
<td>DFHPGLU</td>
<td>Handles the following requests:</td>
<td>LINK_URM</td>
</tr>
<tr>
<td>DFHPGPG</td>
<td>Handles the following requests:</td>
<td>INITIAL_LINK</td>
</tr>
<tr>
<td>DFHPGRE</td>
<td>Handles the following requests:</td>
<td>PREPARE_RETURN_EXEC</td>
</tr>
<tr>
<td>DFHPGRP</td>
<td>Program manager domain recovery program, responsible for recovering</td>
<td>program definitions from the global catalog.</td>
</tr>
<tr>
<td>DFHPGST</td>
<td>Handles the following requests:</td>
<td>COLLECT_STATISTICS, COLLECT_RESOURCE_STATS</td>
</tr>
<tr>
<td>DFHPGUE</td>
<td>Handles program manager domain service requests.</td>
<td></td>
</tr>
<tr>
<td>DFHPGTRI</td>
<td>Interprets PG domain trace entries</td>
<td></td>
</tr>
<tr>
<td>DFHPGX</td>
<td>Handles the following requests:</td>
<td>PREPARE_XCTL_EXEC</td>
</tr>
<tr>
<td>DFHPGXM</td>
<td>Handles the following requests:</td>
<td>INITIALIZE_TRANSACTION, TERMINATE_TRANSACTION</td>
</tr>
</tbody>
</table>

Exits

No global user exit points are provided in this domain.

Trace

The point IDs for the program manager domain are of the form PG xxxx; the corresponding trace levels are PG 1, PG 2, and Exc.

For more information about the trace points, see the CICS Trace Entries. For more information about using traces in problem determination, see the CICS Problem Determination Guide.

916 CICS Transaction Server: CICS Diagnosis Reference
Chapter 67. Program preparation utilities

The program preparation utilities consist of the command-language translators, which are utility programs that run offline to translate CICS application programs using command-level CICS requests. They convert the EXEC commands into call statements in the language in which the EXEC commands are embedded. Versions of the translator program are available for:

- COBOL (DFHECP1$)
- PL/I (DFHEPP1$)
- C/370 (DFHEDP1$)
- Assembler language (DFHEAP1$).

Design overview

The command-language translators manage storage by creating a stack from a single area allocated at the start of the program.

Because the input is free-format, the translators move it into a buffer area that can hold data spanning two or more source records. The analysis of the source is mainly table driven.

The translators build the replacement source code for each EXEC command in a form appropriate to the language:

- For COBOL, the replacement code contains a series of MOVE statements, followed by a CALL statement.
- For PL/I, the replacement code contains a declaration of an entry variable followed by a CALL statement. These statements are contained within a DO group.
- For C/370, the replacement code contains a function call (dfhexec) and may also contain assignment statements.
- For assembler language, the replacement code is an invocation of the DFHECALL macro.

Errors in the source can be detected. Spelling corrections are made to the source, and any unrecognizable or duplicate keywords and options are ignored. For COBOL, PL/I, and C/370, the translator produces error diagnostics that are collected together on the output listing. The assembler language translator, however, produces error diagnostics in the translated output following the EXEC command in which the error occurred.

Modules

DFHECP1$, DFHEPP1$, DFHEDP1$, DFHEAP1$

Exits

Global user exit points are not applicable to offline utilities.

Trace

Trace points are not applicable to offline utilities.
Chapter 68. Recovery Manager Domain (RM)

Recovery Manager (RM) is a domain which is responsible for ensuring that the resource updates for a unit of work are all committed or all backed out, including updates across multiple systems.

Resource Owners, such as File Control, are responsible for processing update requests from applications and for backing out updates. Recovery Manager provides interfaces which Resource Owners use to participate in a unit of work. So Recovery Manager coordinates the Resource Owners ensuring that they all either commit or back out the updates for a particular unit of work. Each Resource Owner protects Recovery Manager from the details of how its resources are managed.

Updates on multiple systems are also coordinated by Recovery Manager. However, since systems are connected in a variety of ways, Recovery Manager uses Recovery Manager Connectors (RMCs) to communicate with remote systems. RMCs, such as the LU 6.2 RMC, are responsible for adapting the Recovery Manager protocols to the inter-system protocols. RMCs protect Recovery Manager from the details of the various inter-system protocols.

Additionally, Recovery Manager supports failures such as a system crash, a remote connection failure, or a local resource failure (e.g. an I/O error). It also supports the forward recovery of local resources allowing them to be reconstructed to a consistent state.

Recovery Manager Domain’s specific gates

Table 82 summarizes the Recovery Manager domain’s specific gates. It shows the level-1 trace point IDs of the modules providing the functions for the gate, the functions provided by the gate, and whether or not the functions are available through the exit programming interface (XPI).

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>XPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>RMUW</td>
<td>RM 0201</td>
<td>CREATE_UOW</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>RM 0202</td>
<td>INQUIRE_UOW_ID</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INQUIRE_UOW_TOKEN</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INQUIRE_UOW</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SET_UOW</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COMMIT_UOW</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FORCE_UOW</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>START_UOW_BROWSE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GET_NEXT_UOW</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>END_UOW_BROWSE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BACKOUT_UOW</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BIND_UOW_TO_TXN</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>REATTACH_REPLY</td>
<td>NO</td>
</tr>
</tbody>
</table>
## Recovery Manager Domain (RM)

**Table 82. Recovery Manager domain’s specific gate (continued)**

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>XPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>RMLN</td>
<td>RM 0301</td>
<td>ADD_LINK</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>RM 0302</td>
<td>DELETE_LINK</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INQUIRE_LINK</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SET_LINK</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ISSUE_PREPARE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INBOUND_FLOW</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INITIATE_RECOVERY</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SET_RECOVERY_STATUS</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>REPORT_RECOVERY_STATUS</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TERMINATE_RECOVERY</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SET_MARK</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>START_LINK_BROWSE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GET_NEXT_LINK</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>END_LINK_BROWSE</td>
<td>NO</td>
</tr>
<tr>
<td>RMNM</td>
<td>RM 0161</td>
<td>INQUIRE_LOGNAME</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>RM 0162</td>
<td>SET_LOGNAME</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CLEAR_PENDING</td>
<td>NO</td>
</tr>
<tr>
<td>RMCD</td>
<td>RM 0121</td>
<td>REGISTER</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>RM 0122</td>
<td>SET_GATE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INQUIRE_CLIENT_DATA</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SET_CLIENT_DATA</td>
<td>NO</td>
</tr>
<tr>
<td>RMDM</td>
<td>RM 0101</td>
<td>INQUIRE_STARTUP</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>RM 0102</td>
<td>SET_STARTUP</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SET_LOCAL_LU_NAME</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SET_PARAMETERS</td>
<td>NO</td>
</tr>
<tr>
<td>RMKD</td>
<td>RM 0231</td>
<td>KEYPOINT_DATA</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>RM 0232</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RMRE</td>
<td>RM 0231</td>
<td>APPEND</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>RM 0232</td>
<td>FORCE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>REMOVE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AVRIL</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>REQUEST_FORGET</td>
<td>NO</td>
</tr>
<tr>
<td>RMSL</td>
<td>RM 06E1</td>
<td>TAKE_ACTIVITY_KEYPOINT</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>RM 06E2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RMWT</td>
<td>RM 0201</td>
<td>INQUIRE_WORK_TOKEN</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>RM 0202</td>
<td>SET_WORK_TOKEN</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>START_WORK_TOKEN_BROWSE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GET_NEXT_WORK_TOKEN</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>END_WORK_TOKEN_BROWSE</td>
<td>NO</td>
</tr>
</tbody>
</table>

### RMUW gate, CREATE_UOW function

Create a unit of work object under the currently executing transaction.

**Input parameters**

**UOW_ID**

An optional parameter specifying the network UOWID to be given to the unit of work object. This parameter will be present if the unit of work being created is part of a distributed unit of work that originated on another system.
HEURISM
An optional parameter specifying whether the unit of work should take a unilateral decision if a failure occurs in the in doubt window? It can have any one of these values:
YES|NO

CHOICE
An optional parameter specifying whether the unit of work should commit or backout if requested to take a unilateral decision. It can have any one of these values:
FORWARD|BACKWARD

INDOUBT_TIMEOUT_INTERVAL
An optional parameter specifying the period of time that the unit of work should be prepared to wait in doubt.

Output parameters
RESPONSE is the Recovery Manager domain’s response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

RMUW gate, INQUIRE_UOW_ID function
Return the network and local UOWIDs of the unit of work of the currently executing transaction.

Input parameters
UOW_ID
An optional parameter specifying a buffer in which the network UOWID will be returned.

Output parameters
LOCAL_UOW_ID
An optional parameter to receive the local UOWID.
RESPONSE is the Recovery Manager domain’s response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

RMUW gate, INQUIRE_UOW_TOKEN function
Return the token identifying the unit of work object with the specified local UOWID.

Input parameters
LOCAL_UOW_ID
The local UOWID of the required unit of work.

Output parameters
UOW_TOKEN
A token identifying the unit of work object.
RESPONSE is the Recovery Manager domain’s response to the call. It can have any one of these values:
Recovery Manager Domain (RM)

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NOT_FOUND</td>
</tr>
</tbody>
</table>

**RMUW INQUIRE_UOW function**

This function is used to query information about a particular unit of work.

**Input parameters**

**UOW_TOKEN**
An optional parameter specifying a token used to identify the unit of work object being queried.

**TRANSACTION_TOKEN**
An optional parameter specifying a token of a transaction whose unit of work object is to be queried.

**UOW_ID**
An optional parameter specifying a buffer in which the network UOWID will be returned.

**LOGNAME**
An optional parameter specifying a buffer in which the log name of the coordinating system will be returned.

**LOCAL_ACCESS_ID**
An optional parameter specifying a buffer in which the local access id of resource causing the unit of work to shunt will be returned.

**REMOTE_ACCESS_ID**
An optional parameter specifying a buffer in which the netname of coordinating system will be returned.

**LINK_ID**
An optional parameter specifying a buffer in which the termid of the link to the coordinating system will be returned.

**Output parameters**

**OUT_UOW_TOKEN**
The token used to identify the unit of work object.

**LOCAL_UOW_ID**
The local unit of work id.

**TRANID**
The tranid of the task that created the unit of work object.

**TERMID**
The termid associated with the task that created the unit of work object.

**TERMINAL_LU_NAME**
The terminal LU name associated with the task that created the unit of work object.

**USERID**
The userid associated with the task that created the unit of work object.
CHOICE
The choice of whether the unit of work should commit or backout if requested to take a unilateral decision. It can have any one of these values:
FORWARD|BACKWARD

UOW_STATUS
The status of the unit of work. It can have any one of these values:
FORWARD|BACKWARD|IN_DOUBT|IN_FLIGHT|
HEURISTIC_FORWARD|HEURISTIC_BACKWARD

SHUNTED
The unit of work may or may not be shunted. It can have any one of these values:
YES|NO

DURATION
An 8 byte Store Clock representation of the time the unit of work changed state.

CREATION_TIME
An 8 byte Store Clock representation of the time the unit of work was created.

CLIENT_NAME
The name of the Recovery Manager client that owns the resource that has caused the unit of work to shunt.

ACCESS_ID_TYPE
The type of resource that has caused the unit of work to shunt. It can have any one of these values:
LOCAL|REMOTE

TRANNUM
The task number of the task that created the unit of work.

OP_ID
The Operator Id associated with the task that created the unit of work.

FIRST_UOW_FOR_TXN
It can have any one of these values:
YES|NO

HEURISM
Whether the unit of work should take a unilateral decision if a failure occurs in the in doubt window? It can have any one of these values:
YES|NO

AWAITING_FORGET
The unit of work might have completed syncpoint processing, and be merely waiting for confirmation that subordinates have completed theirs. It can have any one of these values:
YES|NO

RESPONSE
is the Recovery Manager domain’s response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:
RMUW gate, SET_UOW function

This function is used to set characteristics of the currently executing unit of work.

**Input parameters**

- **HEURISM**
  Determines whether the unit of work will take a unilateral decision if a failure occurs in the in doubt window, or waits for communication with the coordinating system to be reestablished. It can have any one of these values:
  YES | NO

- **HEURISTIC_CAUSE**
  An indication of the reason a unilateral decision must be taken. It can have any one of these values:
  TD_CLIENT | LU61_CLIENT | MRO_CLIENT | RMI_CLIENT | OTHER_CLIENT

**Output parameters**

- **USERID**
  When requested this parameter causes the userid associated with unit of work to be reset to that of the currently executing transaction.

- **RESPONSE**
  is the Recovery Manager domain’s response to the call. It can have any one of these values:
  OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NOT_FOUND</td>
</tr>
</tbody>
</table>

RMUW gate, COMMIT_UOW function

This function attempts to commit the changes made in a unit of work.

**Input parameters**

- **CONTINUE**
  Is the task continuing into a following, new unit of work. This parameter can have any one of these values:
  YES | NO

**Output parameters**

- **RESPONSE**
  is the Recovery Manager domain’s response to the call. It can have any one of these values:
  OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NOT_FOUND</td>
</tr>
</tbody>
</table>
### RMUW gate, FORCE_UOW function

This function forces an in doubt unit of work to unilaterally commit or backout its changes rather than continue waiting for resynchronization with the coordinating system.

**Input parameters**

**UOW_TOKEN**

The token identifying the unit of work object.

**DIRECTION**

Parameter specifying whether to commit (FORWARD), backout (BACKWARD) or obey the ACTION attribute in the definition of the originating transaction. It can have any one of these values:

FORWARD | BACKWARD | HEURISTIC

**HEURISTIC_CAUSE**

The reason for the force. It can have any one of these values:

OPERATOR | TIMEOUT | OTHER_CAUSE

**Output parameters**

**RESPONSE**

is the Recovery Manager domain’s response to the call. It can have any one of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON]

is returned when RESPONSE is EXCEPTION. Possible values are:

**RESPONSE** | **Possible REASON values**
--- | ---
EXCEPTION | NOT_FOUND, RESYNCH_IN_PROGRESS, UOW_NOT_INDOUBT

### RMUW gate, START_UOW_BROWSE function

This function is used to start a browse of unit of work objects in the system.
Recovery Manager Domain (RM)

Input parameters

SHUNTED
The browse can be of only shunted units of work, only non-shunted units of work or all units of work. This parameter can have any one of these values:
YES|NO|BOTH

Output parameters

BROWSE_TOKEN
A token to be used on subsequent GET_NEXT_UOW calls.

RESPONSE
is the Recovery Manager domain’s response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NOT_FOUND</td>
</tr>
</tbody>
</table>

RMUW gate, GET_NEXT_UOW function

This function returns information about the next unit of work object in the browse.

Input parameters

BROWSE_TOKEN
A token obtained from a previous START_UOW_BROWSE call.

UOW_ID
An optional parameter specifying a buffer in which the network UOWID will be returned.

LOGNAME
An optional parameter specifying a buffer in which the log name of the coordinating system will be returned.

LOCAL_ACCESS_ID
An optional parameter specifying a buffer in which the local access id of resource causing the unit of work to shunt will be returned.

REMOTE_ACCESS_ID
An optional parameter specifying a buffer in which the netname of coordinating system will be returned.

LINK_ID
An optional parameter specifying a buffer in which the termid of the link to the coordinating system will be returned.

Output parameters

OUT_UOW_TOKEN
The token used to identify the unit of work object.

LOCAL_UOW_ID
The local unit of work id.
TRANID
The tranid of the task that created the unit of work object.

TERMID
The termid associated with the task that created the unit of work object.

TERMINAL_LUNAME
The terminal LU name associated with the task that created the unit of work object.

USERID
The userid associated with the task that created the unit of work object.

CHOICE
The choice of whether the unit of work should commit or backout if requested to take a unilateral decision. It can have any one of these values:
FORWARD|BACKWARD

UOW_STATUS
The status of the unit of work. It can have any one of these values:
FORWARD|BACKWARD|IN_DOUBT|IN_FLIGHT|
HEURISTIC_FORWARD|HEURISTIC_BACKWARD

SHUNTED
The unit of work may or may not be shunted. It can have any one of these values:
YES|NO

DURATION
An 8 byte Store Clock representation of the time the unit of work changed state.

CREATION_TIME
An 8 byte Store Clock representation of the time the unit of work was created.

CLIENT_NAME
The name of the Recovery Manager client that owns the resource that has caused the unit of work to shunt.

ACCESS_ID_TYPE
The type of resource that has caused the unit of work to shunt. It can have any one of these values:
LOCAL|REMOTE

TRANNUM
The task number of the task that created the unit of work.

OP_ID
The Operator Id associated with the task that created the unit of work.

FIRST_UOW_FOR_TXN
It can have any one of these values:
YES|NO

HEURISM
Whether the unit of work should take a unilateral decision if a failure occurs in the in doubt window? It can have any one of these values:
YES|NO
Recovery Manager Domain (RM)

**AWAITING_FORGET**
The unit of work might have completed syncpoint processing, and be merely waiting for confirmation that subordinates have completed theirs. It can have any one of these values:
YES|NO

**RESPONSE**
is the Recovery Manager domain’s response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**[REASON]**
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>INVALID_BROWSE_TOKEN, BROWSE_END</td>
</tr>
</tbody>
</table>

**RMUW gate, END_UOW_BROWSE function**
This function is used at the end of a browse of the unit of work objects in the system.

**Input parameters**

**BROWSE_TOKEN**
A token obtained from a previous START_UOW_BROWSE call.

**Output parameters**

**RESPONSE**
is the Recovery Manager domain’s response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**[REASON]**
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>INVALID_BROWSE_TOKEN</td>
</tr>
</tbody>
</table>

**RMUW gate, BACKOUT_UOW function**
This function causes the changes in a unit of work to be backed out.

**Input parameters**

**CONTINUE**
This parameter indicates whether the task is continuing into a following, new unit of work. This parameter can have any one of these values:
YES|NO

**RESTART**
This parameter is only applicable when CONTINUE(NO) is specified and indicates whether or not transaction restart will be performed.
Output parameters

RESPONSE

is the Recovery Manager domain’s response to the call. It can have any one of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON]

is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>BACKOUT_FAILURE, COMMIT_FAILURE, ROLLBACK_NOT_SUPPORTED, REMOTE_COMMIT_ABENDED</td>
</tr>
</tbody>
</table>

RMUW gate, BIND_UOW_TO_TXN function

Make the specified unit of work the current unit of work for the current transaction.

Input parameters

UOW_TOKEN

The token identifying the unit of work object.

Output parameters

RESPONSE

is the Recovery Manager domain’s response to the call. It can have any one of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

RMUW gate, REATTACH_REPLY function

This function gives control to Recovery Manager to do its unshunt processing under a re-attached transaction.

Input parameters

UOW_TOKEN

The token identifying the unit of work object.

Output parameters

RESPONSE

is the Recovery Manager domain’s response to the call. It can have any one of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

RMLN gate, ADD_LINK function

This function adds a link to a remote system to a unit of work. The unit of work is distributed across more than one system and Recovery Manager will manage the syncpoint processing between systems.

Input parameters

CLIENT_NAME

Name of the communications protocol used on the link. It can have any one of these values:
Recovery Manager Domain (RM)

LOGNAME_BUFFER
An optional parameter specifying a buffer containing the logname of the remote system.

REMOTE_ACCESS_ID_BUFFER
A buffer containing the netname of the remote system, or the name of the External Resource Manager.

LINK_ID_BUFFER
A buffer containing the termid of the session to the remote system, or the External Resource Manager qualifier.

LINK_ID_SOURCE
An optional parameter specifying whether the local or remote system allocated the session. It can have any one of these values:
LOCAL | REMOTE

RMC_TOKEN
A token to be passed to the client on all callback functions.

LAST
A parameter specifying whether the remote system supports the last agent optimization. It can have any one of these values:
YES | NO | MAYBE | DESIRABLE

PRESUMPTION
A parameter specifying whether the remote system assumes the presume abort or presume nothing protocols. It can have any one of these values:
ABORT | NOTHING

PRELOGGING
A parameter specifying whether the client requires to be called with the PERFORM_PRELOGGING callback function. It can have any one of these values:
YES | NO

SINGLE_UPDATER
A parameter specifying whether the remote system supports the single updater optimization. It can have any one of these values:
YES | NO

COORDINATOR
A parameter specifying whether the remote system is the coordinator of the distributed unit of work. It can have any one of these values:
YES | NO

INITIATOR
A parameter specifying whether the remote system is the initiator of the syncpoint. It can have any one of these values:
YES | NO

RECOVERY_STATUS
A parameter specifying whether recoverable work has taken place as part of the distributed unit of work on the remote system. It can have any one of these values:
NECESSARY | UNNECESSARY | SYNC_LEVEL_1
Output parameters

LINK_TOKEN
A token identifying the new Recovery Manager Link object.

RESPONSE
is the Recovery Manager domain’s response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>LINK_UNKNOWN, COORDINATOR_ALREADY</td>
</tr>
</tbody>
</table>

RMLN gate, DELETE_LINK function

This function removes a link to a remote system from a unit of work. The remote system will not now be included in syncpoint processing for the current unit of work.

Input parameters

LINK_TOKEN
A token identifying the Recovery Manager Link object.

Output parameters

RESPONSE
is the Recovery Manager domain’s response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>LINK_UNKNOWN</td>
</tr>
</tbody>
</table>

RMLN gate, INQUIRE_LINK function

This function returns information about a given Recovery Manager Link object.

Input parameters

LINK_TOKEN
A token identifying a Recovery Manager Link object.

RESOLVE_TO_CURRENT_LINK
Up to two Recovery Manager Link objects may be associated with a token. This optional parameter specifies whether to return information about the most recent or not. It can have any one of these values:
YES|NO

REMOTE_ACCESS_ID_BUFFER
A buffer in which the netname of the remote system, or External Resource Manager name will be returned.
Recovery Manager Domain (RM)

LOGNAME_BUFFER
A buffer in which the logname of the remote system will be returned.

LINK_ID_BUFFER
A buffer in which the termid of the session to the remote system, or
External Resource Manager qualifier will be returned.

Output parameters

CLIENT_NAME
The name of the protocol that owns the Recovery Manager Link object. It
can have any one of these values:
IRC |IRCO|LU61|LU62|RMI |IND

COORDINATOR
Whether the remote system is the coordinator of the distributed unit of
work. It can have any one of these values:
YES|NO

INITIATOR
Whether the remote system is the initiator of the syncpoint of the
distributed unit of work. It can have any one of these values:
YES|NO

LAST
Whether the remote system supports the last agent optimization. It can
have any one of these values:
YES|NO|MAYBE

SINGLE_UPDATER
Whether the remote system supports the single updater optimization. It
can have any one of these values:
YES|NO

PRESUMPTION
Whether the remote system assumes the presume abort or presume
nothing protocols. It can have any one of these values:
ABORT|NOTHING

RECOVERY_STATUS
Whether recoverable work has taken place as part of the distributed unit of
work on the remote system. It can have any one of these values:
NECESSARY|UNNECESSARY|SYNC_LEVEL_1

FORGET
Whether all obligations to the remote system with respect to recovery have
been discharged. It can have any one of these values:
YES|NO

MARK
Whether the Recovery Manager Link object has been marked during
resynchronization. It can have any one of these values:
YES|NO

UNSHUNTED
Whether the unit of work is not currently shunted. It can have any one of
these values:
YES|NO

RESYNC_SCHEDULED
Whether resynchronization activity has been scheduled. It can have any
one of these values:
ACCESSIBLE
Whether the communications link to the remote system is active or not. It can have any one of these values:
YES | NO | SHUNTED

LINK_ID_SOURCE
Whether the local or remote system allocated the session. It can have any one of these values:
LOCAL | REMOTE

UOW_TOKEN
The token identifying the unit of work object.

LOCAL_UOW_ID
The local unit of work id of the unit of work to which the Recovery Manager Link object belongs.

HEURISM
Whether the unit of work to which the Recovery Manager Link object belongs will take a unilateral decision if a failure occurs in the in doubt window. It can have any one of these values:
YES | NO

RMC_TOKEN
A token to be passed to the client on all callback functions.

RESPONSE is the Recovery Manager domain’s response to the call. It can have any one of these values:
OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>LINK_UNKNOWN</td>
</tr>
</tbody>
</table>

RMLN gate, SET_LINK function
This function is used to set characteristics of a Recovery Manager Link object.

Input parameters

LINK_TOKEN
A token used to identify a Recovery Manager Link object.

RESOLVE_TO_CURRENT_LINK
Up to two Recovery Manager Link objects may be associated with a token. This optional parameter specifies whether to set characteristics of the most recent or not. It can have any one of these values:
YES | NO

LOGNAME_BUFFER
An optional parameter specifying a buffer containing a logname to be associated with the Recovery Manager Link object.

COORDINATOR
A parameter specifying whether the remote system is the coordinator of the distributed unit of work. It can have any one of these values:
Recovery Manager Domain (RM)

YES|NO

INITIATOR
A parameter specifying whether the remote system is the initiator of the syncpoint. It can have any one of these values:
YES|NO

RECOVERY_STATUS
A parameter specifying whether recoverable work has taken place as part of the distributed unit of work on the remote system. It can have any one of these values:
NECESSARY|UNNECESSARY|SYNC_LEVEL_1

SINGLE_UPDATER
A parameter specifying whether the remote system supports the single updater optimization. It can have any one of these values:
YES|NO

PRELOGGING
A parameter specifying whether the client requires to be called with the PERFORM_PRELOGGING callback function. It can have any one of these values:
YES|NO

LINK_ID_BUFFER
A buffer containing the termid of the session to the remote system, or the External Resource Manager qualifier.

LINK_ID_SOURCE
An optional parameter specifying whether the local or remote system allocated the session. It can have any one of these values:
LOCAL|REMOTE

UNSHUNTED
A parameter specifying whether the unit of work is not currently shunted. It can have any one of these values:
YES|NO

RESYNC_SCHEDULED
A parameter specifying whether resynchronization activity has been scheduled. It can have any one of these values:
YES|NO

ACCESSIBLE
A parameter specifying that the communications link to the remote system has failed. It can have any one of these values:
NO|SHUNTED

FORGET
A parameter specifying whether all obligations to the remote system with respect to recovery have been discharged. It can have any one of these values:
YES|NO

Output parameters

RESPONSE
is the Recovery Manager domain’s response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED
RMLN gate, ISSUE_PREPARE function
This function performs phase 1 of syncpoint processing on the specified Recovery Manager Link object.

Input parameters

**LINK_TOKEN**
A token used to identify a Recovery Manager Link object.

**CONTINUE**
Is the task continuing into a following, new unit of work. This parameter can have any one of these values:
YES|NO

Output parameters

**VOTE**
The vote from the client owning the Recovery Manager Link object. This parameter can have any one of these values:
YES|NO|NO_CONTINUE|READ_ONLY

**RESPONSE**
is the Recovery Manager domain’s response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th><strong>RESPONSE</strong></th>
<th><strong>Possible REASON values</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>LINK_UNKNOWN,</td>
</tr>
<tr>
<td></td>
<td>COORDINATOR_ALREADY,</td>
</tr>
<tr>
<td></td>
<td>INITIATOR_ALREADY</td>
</tr>
</tbody>
</table>

RMLN gate, INBOUND_FLOW function
This function is used to notify Recovery Manager of the successful completion of syncpoint processing on the remote system, or a communications failure with the remote system.

Input parameters

**LINK_TOKEN**
A token used to identify a Recovery Manager Link object.

**FLOW**
A parameter specifying successful completion (DATA) or communication failure (UNBIND). It can have any one of these values:
DATA|UNBIND
**Recovery Manager Domain (RM)**

**Output parameters**

**RESPONSE**

is the Recovery Manager domain's response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**[REASON]**

is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>LINK_UNKNOWN, LINK_INACCESSIBLE</td>
</tr>
</tbody>
</table>

**RMLN gate, INITIATE_RECOVERY function**

This function identifies a Recovery Manager Link object in an in doubt failed unit of work and marks it as being resynchronized.

**Input parameters**

**UOW_ID**

An optional parameter specifying a buffer containing the network UOWID of the unit of work to be resynchronized.

**LOCAL_UOW_ID**

An optional parameter specifying the local UOWID.

**CLIENT_NAME**

The name of the Recovery Manager client that owns the Recovery Manager Link object over which resynchronization is to take place.

**REMOTE_ACCESS_ID_BUFFER**

A buffer containing the netname of the remote system, or the name of the External Resource Manager of the Recovery Manager Link object over which resynchronization is to take place.

**LINK_ID_BUFFER**

A buffer containing the termid of the session to the remote system, or the External Resource Manager qualifier of the Recovery Manager Link object over which resynchronization is to take place.

**LINK_ID_SOURCE**

An optional parameter specifying whether the local or remote system allocated the session associated with the Recovery Manager Link object over which resynchronization is to take place. It can have any one of these values:

LOCAL|REMOTE

**DIRECTION**

A parameter specifying whether the resynchronization activity was initiated by the local or remote system. It can have any one of these values:

INBOUND|OUTBOUND

**Output parameters**

**UOW_TOKEN**

The token identifying the unit of work object to which the Recovery Manager Link object being resynchronized belongs.
LINK_TOKEN
A token identifying the Recovery Manager Link object being resynchronized.

COORDINATOR
A parameter specifying whether the remote system is the coordinator of the distributed unit of work. It can have any one of these values:
YES|NO

INITIATOR
A parameter specifying whether the remote system is the initiator of the syncpoint. It can have any one of these values:
YES|NO

PRESUMPTION
Whether the remote system assumes the presume abort or presume nothing protocols. It can have any one of these values:
ABORT|NOTHING

UOW_STATUS
The status of the unit of work object that the Recovery Manager Link object belongs to. It can have any one of these values:
INDOUBT|FORWARD|BACKWARD|
HEURISTIC_FORWARD|HEURISTIC_BACKWARD

FAILURE_TIME
An 8 byte Store Clock representation of the in doubt failure time.

RESPONSE
is the Recovery Manager domain’s response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>LINK_UNKNOWN,</td>
</tr>
<tr>
<td></td>
<td>RECOVERY_ALREADY_IN_PROG,</td>
</tr>
<tr>
<td></td>
<td>LINK_ACTIVE</td>
</tr>
</tbody>
</table>

RMLN gate, SET_RECOVERY_STATUS function
This function is used to notify an Recovery Manager Link object of the outcome of a distributed unit of work which failed in the in doubt window. It results in the shunted unit of work the Recovery Manager Link object belongs to unshunting and committing or backing out its resource updates as appropriate.

Input parameters

LINK_TOKEN
A token identifying the Recovery Manager Link object being resynchronized.

DIRECTION
A parameter specifying whether the resynchronization activity was initiated by the local or remote system. It can have any one of these values:
INBOUND|OUTBOUND
Recovery Manager Domain (RM)

**REMOTE_UOW_STATUS**

The status of the unit of work in the remote system. It can have any one of these values:

```markdown
INDOUBT | HEURISTIC_FORWARD | HEURISTIC_BACKWARD |
FORWARD  | BACKWARD           | HEURISTIC_MIXED   |
COLD     | RESET              | UNKNOWN
```

**TOLERATE_VIOLATIONS**

A parameter specifying the rules to be used to detect resynchronization protocol violations. It can have any one of these values:

YES | NO

**Input parameters**

**UOW_ID**

A parameter specifying a buffer containing the network UOWID of the unit of work to be resynchronized.

**REMOTE_ACCESS_ID_BUFFER**

A buffer containing the netname of the remote system, or the name of the External Resource Manager of the Recovery Manager Link object over which resynchronization is to take place.

**REMOTE_UOW_STATUS**

The status of the unit of work in the remote system. It can have any one of these values:

```markdown
INDOUBT | HEURISTIC_FORWARD | HEURISTIC_BACKWARD |
FORWARD  | BACKWARD           | HEURISTIC_MIXED   |
```

**Output parameters**

**UOW_STATUS**

The status (as a result of the resynchronization) of the unit of work object to which the Recovery Manager Link object belongs. It can have any one of these values:

```markdown
INDOUBT | HEURISTIC_FORWARD | HEURISTIC_BACKWARD |
FORWARD  | BACKWARD           |
```

**RESPONSE**

is the Recovery Manager domain’s response to the call. It can have any one of these values:

```markdown
OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED
```

**[REASON]**

is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>LINK_UNKNOWN, RECOVERY_NOT_IN_PROGRESS, ALREADY_SET</td>
</tr>
</tbody>
</table>

**RMLN gate, REPORT_RECOVERY_STATUS function**

This function is similar to SET_RECOVERY_STATUS but is applicable in the case of Presumed Abort or Last Agent resynchronization where the coordinator has backed out and has no record of the UOW. The participant may have gone indoubt, and needs to resynchronize.

**Input parameters**

**UOW_ID**

A parameter specifying a buffer containing the network UOWID of the unit of work to be resynchronized.

**REMOTE_ACCESS_ID_BUFFER**

A buffer containing the netname of the remote system, or the name of the External Resource Manager of the Recovery Manager Link object over which resynchronization is to take place.

**REMOTE_UOW_STATUS**

The status of the unit of work in the remote system. It can have any one of these values:

```markdown
INDOUBT | HEURISTIC_FORWARD | HEURISTIC_BACKWARD |
HEURISTIC_MIXED
```
Output parameters

RESPONSE is the Recovery Manager domain’s response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

RMLN gate, TERMINATE_RECOVERY function

Input parameters

LINK_TOKEN
A token identifying the Recovery Manager Link object being resynchronized.

DIRECTION
A parameter specifying whether the resynchronization activity was initiated by the local or remote system. It can have any one of these values:
INBOUND|OUTBOUND

FORGET
A parameter specifying whether all obligations to the remote system with respect to recovery have been discharged. It can have any one of these values:
YES|NO

OPERATOR_INITIATED
A parameter specifying whether the function is the result of an explicit user action. It can have any one of these values:
YES|NO

Output parameters

RESPONSE is the Recovery Manager domain’s response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>LINK_UNKNOWN, RECOVERY_NOT_IN_PROGRESS, SET_NOT_DONE</td>
</tr>
</tbody>
</table>

RMLN gate, SET_MARK function

This function marks a Recovery Manager Link object during recovery.

Input parameters

LINK_TOKEN
A token identifying the Recovery Manager Link object to be marked.

MARK
It can have any one of these values:
YES|NO
Recovery Manager Domain (RM)

Output parameters

**RESPONSE**

is the Recovery Manager domain’s response to the call. It can have any one of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON]

is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>LINK_UNKNOWN, LINK_ACTIVE, RECOVERY_IN_PROGRESS</td>
</tr>
</tbody>
</table>

RMLN gate, START_LINK_BROWSE function

This function starts a browse of Recovery Manager Link objects. The browse can return either

- all the Recovery Manager Link objects in the system owned by a particular Recovery Manager client and associated with a particular remote system or External Resource Manager, or
- all Recovery Manager Link objects belonging to a particular unit of work object.

Input parameters

**CLIENT_NAME**

The name of a Recovery Manager client.

**REMOTE_ACCESS_ID_BUFFER**

A buffer containing the netname of the remote system, or the name of the External Resource Manager.

**UOW_TOKEN**

The token identifying a unit of work object.

Output parameters

**LINK_BROWSE_TOKEN**

A token to be used during a browse of all Recovery Manager Link objects for a particular Recovery Manager client.

**UOW_BROWSE_TOKEN**

A token to be used during a browse of all Recovery Manager Link objects for a particular unit of work object.

**RESPONSE**

is the Recovery Manager domain’s response to the call. It can have any one of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON]

is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>UOW_UNKNOWN, CLIENT_UNKNOWN</td>
</tr>
</tbody>
</table>
**RMLN gate, GET_NEXT_LINK function**

This function returns information about the next Recovery Manager Link object in a browse.

**Input parameters**

**LINK_BROWSE_TOKEN**
A token identifying a browse of all the Recovery Manager Link objects belonging to a particular Recovery Manager client.

**UOW_BROWSE_TOKEN**
A token identifying a browse of all the Recovery Manager Link objects belonging to a particular unit of work object.

**REMOTE_ACCESS_ID_BUFFER**
A buffer in which the netname of the remote system, or External Resource Manager name will be returned.

**LOGNAME_BUFFER**
A buffer in which the logname of the remote system will be returned.

**LINK_ID_BUFFER**
A buffer in which the termid of the session to the remote system, or External Resource Manager qualifier will be returned.

**Output parameters**

**LINK_TOKEN**

**CLIENT_NAME**
The name of the protocol that owns the Recovery Manager Link object. It can have any one of these values:
- IRC
- IRCO
- LU61
- LU62
- RMI
- IND

**COORDINATOR**
Whether the remote system is the coordinator of the distributed unit of work. It can have any one of these values:
- YES
- NO

**INITIATOR**
Whether the remote system is the initiator of the syncpoint of the distributed unit of work. It can have any one of these values:
- YES
- NO

**LAST**
Whether the remote system supports the last agent optimization. It can have any one of these values:
- YES
- NO
- MAYBE

**SINGLE_UPDATER**
Whether the remote system supports the single updater optimization. It can have any one of these values:
- YES
- NO

**PRESCRIPTION**
Whether the remote system assumes the presume abort or presume nothing protocols. It can have any one of these values:
- ABORT
- NOTHING

**RECOVERY_STATUS**
Whether recoverable work has taken place as part of the distributed unit of work on the remote system. It can have any one of these values:
Recovery Manager Domain (RM)

**NECESSARY|UNNECESSARY|SYNC\_LEVEL\_1**

**FORGET**
Whether all obligations to the remote system with respect to recovery have been discharged. It can have any one of these values:
YES\|NO

**MARK**
Whether the Recovery Manager Link object has been marked during resynchronization. It can have any one of these values:
YES\|NO

**UNSHUNTED**
Whether the unit of work is not currently shunted. It can have any one of these values:
YES\|NO

**RESYNC\_SCHEDULED**
Whether resynchronization activity has been scheduled. It can have any one of these values:
YES\|NO

**ACCESSIBLE**
Whether the communications link to the remote system is active or not. It can have any one of these values:
YES\|NO\|SHUNTED

**LINK\_ID\_SOURCE**
Whether the local or remote system allocated the session. It can have any one of these values:
LOCAL\|REMOTE

**UOW\_TOKEN**
The token identifying the unit of work object.

**LOCAL\_UOW\_ID**
The local unit of work id of the unit of work to which the Recovery Manager Link object belongs.

**HEURISM**
Whether the unit of work to which the Recovery Manager Link object belongs will take a unilateral decision if a failure occurs in the in doubt window. It can have any one of these values:
YES\|NO

**RMC\_TOKEN**
A token to be passed to the client on all callback functions.

**RESPONSE**
is the Recovery Manager domain’s response to the call. It can have any one of these values:
OK\|EXCEPTION\|DISASTER\|INVALID\|KERNERROR\|PURGED

[**REASON**]
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>UOW_UNKNOWN,</td>
</tr>
<tr>
<td></td>
<td>END_BROWSE</td>
</tr>
</tbody>
</table>
RMLN gate, **END_LINK_BROWSE function**

This function is used to terminate a browse of Recovery Manager Link objects.

**Input parameters**

**LINK_BROWSE_TOKEN**

A token identifying a browse of all the Recovery Manager Link objects belonging to a particular Recovery Manager client.

**UOW_BROWSE_TOKEN**

A token identifying a browse of all the Recovery Manager Link objects belonging to a particular unit of work object.

**Output parameters**

**RESPONSE**

is the Recovery Manager domain’s response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**[REASON]**

is returned when RESPONSE is INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>INVALID_BROWSE</td>
</tr>
</tbody>
</table>

RMNM gate, **INQUIRE_LOGNAME function**

This function returns the logname and data associated with the specified remote system being communicated with via the specified Recovery Manager client.

**Input parameters**

**CLIENT_NAME**

Name of a Recovery Manager client.

**REMOTE_ACCESS_ID_BUFFER**

A buffer containing the netname of the remote system.

**LOGNAME_BUFFER**

A buffer to be used to return the logname.

**RMC_DATA_BUFFER**

A buffer to be used to return data owned by the Recovery Manager client.

**Output parameters**

**IN_USE**

Whether there are any Recovery Manager Link object in the system associated with the logname. It can have any one of these values:

YES|NO

**RESPONSE**

is the Recovery Manager domain’s response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED
Recovery Manager Domain (RM)

[REASON]

is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NOT_FOUND, UNKNOWN_CLIENT</td>
</tr>
</tbody>
</table>

**RMNM gate, SET_LOGNAME function**

This function is used to associate a logname and some data with the netname of a remote system for a specified Recovery Manager client.

**Input parameters**

**CLIENT_NAME**

A name of a Recovery Manager client.

**REMOTE_ACCESS_ID_BUFFER**

A buffer containing the netname of a remote system.

**LOGNAME_BUFFER**

A buffer containing the logname to be associated with the netname.

**RMC_DATA_BUFFER**

A buffer containing data to be associated with the netname.

**Output parameters**

**RESPONSE**

is the Recovery Manager domain’s response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]

is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>UNKNOWN_CLIENT</td>
</tr>
</tbody>
</table>

**RMNM gate, CLEAR_PENDING function**

This function is used to remove Recovery Manager Link objects associated with a specified remote system. Affected indoubt units of work will take a unilateral decision to commit or backout their resource updates.

**Input parameters**

**CLIENT_NAME**

A name of a Recovery Manager client.

**REMOTE_ACCESS_ID_BUFFER**

A buffer containing the netname of the remote system.

**COLD**

A parameter specifying whether the remote system has a new log and so has lost recovery information with respect to units of work in this system. It can have any one of these values:

YES|NO

**ALL**

A parameter specifying whether only Recovery Manager Link objects with
the same logname as that currently associated with the remote system should be removed or all Recovery Manager Link objects.

Output parameters
RESPONSE
is the Recovery Manager domain’s response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NOT_FOUND</td>
</tr>
</tbody>
</table>

RMCD gate, REGISTER function
This function is used to register a Recovery Manager client.

Input parameters
CLIENT_NAME
A name of a Recovery Manager client.

CLIENT_TYPE
Whether the client owns local (RO) or remote (RMC) resources. It can have any one of these values:
RO|RMC

GATE An optional parameter specifying the kernel gate that services the client’s callback functions.

Output parameters
RESPONSE
is the Recovery Manager domain’s response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>ALREADY_REGISTERED, TOO_LATE</td>
</tr>
</tbody>
</table>

RMCD gate, SET_GATE function
This function is used to inform Recovery Manager of the kernel gate that services a Recovery Manager clients callback functions.

Input parameters
CLIENT_NAME
A name of a Recovery Manager client.

GATE A parameter specifying the kernel gate that services the client’s callback functions.
Recovery Manager Domain (RM)

Output parameters

RESPONSE
is the Recovery Manager domain’s response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>UNKNOWN_CLIENT,</td>
</tr>
<tr>
<td></td>
<td>GET_ALREADY_SET</td>
</tr>
</tbody>
</table>

RMCD gate, INQUIRE_CLIENT_DATA function

This function returns data associated with a Recovery Manager client.

Input parameters

CLIENT_NAME
A name of a Recovery Manager client.

CLIENT_DATA_BUFFER
A buffer to contain the data returned.

Output parameters

RESPONSE
is the Recovery Manager domain’s response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>UNKNOWN_CLIENT,</td>
</tr>
<tr>
<td></td>
<td>CLIENT_DATA_TOO_LONG</td>
</tr>
</tbody>
</table>

RMCD gate, SET_CLIENT_DATA function

This function associates some data with a Recovery Manager client.

Input parameters

CLIENT_NAME
A name of a Recovery Manager client.

CLIENT_DATA_BUFFER
A buffer containing the data to be associated with the Recovery Manager client.

Output parameters

RESPONSE
is the Recovery Manager domain’s response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED
[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>UNKNOWN_CLIENT, CLIENT_DATA_TOO_LONG</td>
</tr>
</tbody>
</table>

**RMDM gate, INQUIRE_STARTUP function**

This function returns information about the type of system start being performed.

**Input parameters**

None

**Output parameters**

**STARTUP**

It can have any one of these values:

- COLD
- WARM
- EMERGENCY

**ALL**

A value specifying whether all components are cold starting. It can have any one of these values:

- YES
- NO

**INITIAL_START**

A value specifying whether the cold start is in fact an initial one. It can have any one of these values:

- YES
- NO

**LAST_COLD_START_TIME**

An 8 byte Store Clock representation of the last cold start time.

**LAST_EMER_START_TIME**

An 8 byte Store Clock representation of the last emergency start time.

**LAST_INIT_START_TIME**

An 8 byte Store Clock representation of the last initial start time.

**RESPONSE**

is the Recovery Manager domain’s response to the call. It can have any one of these values:

- OK
- EXCEPTION
- DISASTER
- INVALID
- KERNERROR
- PURGED

**RMDM gate, SET_STARTUP function**

This function sets the type of start that will be performed when this system is next restarted.

**Input parameters**

**STARTUP**

The type of start. It can have any one of these values:

- COLD
- NORESTART

**Output parameters**

**RESPONSE**

is the Recovery Manager domain’s response to the call. It can have any one of these values:

- OK
- EXCEPTION
- DISASTER
- INVALID
- KERNERROR
- PURGED
Recovery Manager Domain (RM)

**RMDM gate, SET_LOCAL_LU_NAME function**

This function sets the local LU name, that is used in the generation of network UOWIDs by in this system.

**Input parameters**

- **LOCAL_LU_NAME**
  A parameter specifying the local LU name.

- **LOCAL_LU_NAME_LENGTH**
  A parameter specifying the length of the local LU name.

**Output parameters**

- **RESPONSE**
  is the Recovery Manager domain’s response to the call. It can have any one of these values:
  OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**RMDM gate, SET_PARAMETERS function**

This function is used only by Parameter Manager Domain to inform Recovery Manager of initialization parameters.

**Input parameters**

- **DELETE_LOG**
  An optional parameter specifying whether an initial start has been requested in the System Initialization Table, and so the contents of the system log should be deleted. It can have any one of these values:
  YES|NO

- **STARTUP**
  An optional parameter used in the case where OFFSITE=YES has been specified as a SIT override. It can only have the value EMERGENCY.

**Output parameters**

- **RESPONSE**
  is the Recovery Manager domain’s response to the call. It can have any one of these values:
  OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**RMKD gate, KEYPOINT_DATA function**

This function writes Recovery Manager client data to the system log for keypointing purposes.

**Input parameters**

- **CLIENT_NAME**
  A name of a Recovery Manager client.

- **DATA**
  Address of an extended Iliffe vector. An extended Iliffe vector consists of a linked list of at least one element. Each element of the linked list consists of a variable length array of address length pairs. Each address and length field is four bytes long. The top bit of each address is off except for the last which may be on.

  If an address is binary zero, then this terminates the element and the linked list.
If an address has the top bit on, then it terminates the element and points to the next element in the linked list.

An extended Iliffe vector simply represents the block of data formed by concatenating all the blocks which are pointed to by address length pairs in the vector which have the address top bit off. The order is from front to back of the linked list and from low to high index within each array.

REMARK
An optional parameter for the benefit of trace to describe the data being logged.

RAISE_INV_DATA_LENGTH
An optional parameter specifying whether the caller wishes to be informed of there being too much data to be logged. It can have any one of these values:
YES|NO

Output parameters
RESPONSE
is the Recovery Manager domain’s response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>INVALID_DATA_LENGTH, INVALID_CLIENT_NAME,</td>
</tr>
<tr>
<td></td>
<td>NO_DATA</td>
</tr>
</tbody>
</table>

RMRE gate, APPEND function
This function writes data to the system log. The data written is associated with the current unit of work of the currently executing transaction if either FORWARD_DATA(YES) or BACKWARD_DATA(YES) is specified.

Input parameters
CLIENT_NAME
A name of a Recovery Manager client.

RESOURCE_ID
A parameter specifying the name of the resource with which the data to be logged is associated.

DATA
Address of an extended Iliffe vector. An extended Iliffe vector consists of a linked list of at least one element. Each element of the linked list consists of a variable length array of address length pairs. Each address and length field is four bytes long. The top bit of each address is off except for the last which may be on.

If an address is binary zero, then this terminates the element and the linked list.

If an address has the top bit on, then it terminates the element and points to the next element in the linked list.
Recovery Manager Domain (RM)

An extended Iliffe vector simply represents the block of data formed by concatenating all the blocks which are pointed to by address length pairs in the vector which have the address top bit off. The order is from front to back of the linked list and from low to high index within each array.

**FORCE_DATA**
A parameter specifying whether the data is forced out on to the non-volatile log or can merely be written to the volatile log buffer. It can have any one of these values:
- YES
- NO

**FORWARD_DATA**
A parameter specifying whether the data is used for forward recovery purposes. It can have any one of these values:
- YES
- NO

**BACKWARD_DATA**
A parameter specifying whether the data is used for backward recovery purposes. It can have any one of these values:
- YES
- NO

**REMARK**
An optional parameter for the benefit of trace to describe the data being logged.

**LOG_BUFFER_SUSPEND**
An optional parameter specifying whether the caller can tolerate the task suspending to wait for space in a log buffer. It can have any one of these values:
- YES
- NO

**RAISE_INV_DATA_LENGTH**
An optional parameter specifying whether the caller wishes to be informed of there being too much data to be logged. It can have any one of these values:
- YES
- NO

**Output parameters**

**FORCE_TOKEN**
A token that can be used to force the data on to the non-volatile log with the FORCE function of the RMRE gate.

**RESPONSE**
is the Recovery Manager domain’s response to the call. It can have any one of these values:
- OK
- EXCEPTION
- DISASTER
- INVALID
- KERNERROR
- PURGED

**[REASON]**
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>INVALID_DATA_LENGTH,</td>
</tr>
<tr>
<td></td>
<td>INSUFFICIENT_BUFFER_SPACE,</td>
</tr>
<tr>
<td></td>
<td>INVALID_CLIENT_NAME,</td>
</tr>
<tr>
<td></td>
<td>INVALID_Resource_ID,</td>
</tr>
<tr>
<td></td>
<td>NO_DATA</td>
</tr>
</tbody>
</table>

CICS Transaction Server: CICS Diagnosis Reference
RMRE gate, FORCE function
This function forces data written previously to a log buffer to the non-volatile log.

Input parameters
FORCE_TOKEN
A token returned on a previous call to the APPEND function of the RMRE gate.

Output parameters
RESPONSE
is the Recovery Manager domain’s response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

RMRE gate, REMOVE function
This function removes data logged by a Recovery Manager client and associated with a particular local resource from a unit of work.

Input parameters
UOW_ID
The network UOWID under which the data was logged.

LOCAL_UOW_ID
The local UOWID under which the data was logged.

CLIENT_NAME
The name of the Recovery Manager client that logged the data.

LOCAL_ACCESS_ID
The name of the local resource with which the logged data was associated.

Output parameters
RESPONSE
is the Recovery Manager domain’s response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>UOW_NOT_SHUNTED,</td>
</tr>
<tr>
<td></td>
<td>UOW_NOT_BACKWARDS,</td>
</tr>
<tr>
<td></td>
<td>INVALID_CLIENT_NAME,</td>
</tr>
<tr>
<td></td>
<td>INVALID_LOCAL_ACCESS_ID</td>
</tr>
</tbody>
</table>

RMRE gate, AVAIL function
This function informs Recovery Manager that a local resource has become available. It is used when either a backout failure or a commit failure has previously occurred and the resource (or reason for the failure) has now cleared - or there is now reason to believe it may have cleared.
Recovery Manager Domain (RM)

**Input parameters**

- **CLIENT_NAME**
  The name of the Recovery Manager client that owns the local resource.

- **LOCAL_ACCESS_ID**
  The name of the local resource.

**Output parameters**

- **RESPONSE**
  is the Recovery Manager domain’s response to the call. It can have any one of these values:
  - OK
  - EXCEPTION
  - DISASTER
  - INVALID
  - KERNERROR
  - PURGED

- **[REASON]**
  is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>LOCAL_ACCESS_ID_UNKNOWN</td>
</tr>
</tbody>
</table>

**RMRE gate, REQUEST_FORGET function**

This function associates a Recovery Manager client and a named local resource with a requirement to engage in forget processing.

**Input parameters**

- **CLIENT_NAME**
  The name of the Recovery Manager client that owns the local resource.

- **LOCAL_ACCESS_ID**
  The name of the local resource.

**Output parameters**

- **RESPONSE**
  is the Recovery Manager domain’s response to the call. It can have any one of these values:
  - OK
  - EXCEPTION
  - DISASTER
  - INVALID
  - KERNERROR
  - PURGED

- **[REASON]**
  is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>INVALID_CLIENT_NAME, INVALID_LOCAL_ACCESS_ID</td>
</tr>
</tbody>
</table>

**RMSL gate, TAKE_ACTIVITY_KEYPOINT function**

This function performs the activity associated with taking a keypoint.

**Input parameters**

None

**Output parameters**

- **RESPONSE**
  is the Recovery Manager domain’s response to the call. It can have any one of these values:
[REASON]
is returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>None</td>
</tr>
</tbody>
</table>

RMWT gate, INQUIRE_WORK_TOKEN function
This function returns the value of the work token belonging to the named Recovery Manager client in a particular unit of work object.

**Input parameters**

**UOW_TOKEN**
An optional parameter specifying the token identifying a unit of work object. If not specified the work token from the current unit of work of the currently executing transaction is returned.

**CLIENT_NAME**
The name of a Recovery Manager client.

**Output parameters**

**WORK_TOKEN**
The value of the Recovery Manager clients work token in the specified unit of work object.

**RESPONSE**
is the Recovery Manager domain’s response to the call. It can have any one of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NOT_FOUND</td>
</tr>
</tbody>
</table>

RMWT gate, START_WORK_TOKEN_BROWSE function
This function starts a browse of a all the non-zero work tokens in the system for a specific Recovery Manager client.

**Input parameters**

**CLIENT_NAME**
The name of a Recovery Manager client.

**Output parameters**

**BROWSE_TOKEN**
A token to be used during the browse.

**RESPONSE**
is the Recovery Manager domain’s response to the call. It can have any one of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED
Recovery Manager Domain (RM)

[REASON]

is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NOT_FOUND</td>
</tr>
</tbody>
</table>

RMWT gate, GET_NEXT_WORK_TOKEN function

This function returns the next non-zero work token for the Recovery Manager client specified on the START_WORK_TOKEN_BROWSE. The token used to identify the unit of work object and local UOWID associated with the work token are also optionally returned.

Input parameters

BROWSE_TOKEN
A token identifying the browse.

Output parameters

WORK_TOKEN
The value of the Recovery Manager clients work token.

UOW_TOKEN
The token identifying the unit of work object.

LOCAL_UOW_ID
The local UOWID.

RESPONSE
is the Recovery Manager domain’s response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]

is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>INVALID_BROWSE_TOKEN, BROWSE_END</td>
</tr>
</tbody>
</table>

RMWT gate, END_WORK_TOKEN_BROWSE function

This function terminates a browse of work tokens.

Input parameters

BROWSE_TOKEN
A token identifying the browse.

Output parameters

RESPONSE
is the Recovery Manager domain’s response to the call. It can have any one of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]

is returned when RESPONSE is EXCEPTION. Possible values are:
Recovery Manager domain’s generic gates

Table 83 summarizes the Recovery Manager domain’s generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gate, the functions provided by the gate, and the generic format for calls to the gate.

Table 83. Recovery Manager domain’s generic gate

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMDM</td>
<td>RM 0101</td>
<td>INITIALIZE_DOMAIN</td>
<td>DMDM</td>
</tr>
<tr>
<td></td>
<td>RM 0102</td>
<td>QUIESCE_DOMAIN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TERMINATE_DOMAIN</td>
<td></td>
</tr>
</tbody>
</table>

You can find descriptions of these functions and their input and output parameters, in the section dealing with the corresponding generic format, in “Chapter 28. Domain manager domain (DM)” on page 369.

In Initialization processing, the Recovery Manager
• Obtains initialization parameters from Parameter Manager,
• Determines the type of start to be performed,
• Processes it’s data from the Global Catalog,
• Processes recovery information from the System Log.

In Quiesce processing, the Recovery Manager takes the warm keypoint.

Recovery Manager domain’s call back formats

Table 84 describes the call back format owned by the Recovery Manager domain and shows the function performed on the calls.

Table 84. Call back format owned by the Recovery Manager domain

<table>
<thead>
<tr>
<th>Format</th>
<th>Calling module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>RMRO</td>
<td>DFHRMUO</td>
<td>PERFORM_COMMIT</td>
</tr>
<tr>
<td></td>
<td>DFHRMUP</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DFHRMUQ</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DFHRMUW</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DFHRMUO</td>
<td>PERFORM_PREPARE</td>
</tr>
<tr>
<td></td>
<td>DFHRMRO2</td>
<td>START_BACKOUT</td>
</tr>
<tr>
<td></td>
<td>DFHRMRO3</td>
<td>DELIVER_BACKOUT_DATA</td>
</tr>
<tr>
<td></td>
<td>DFHRMRO4</td>
<td>END_BACKOUT</td>
</tr>
<tr>
<td></td>
<td>DFHRMROS</td>
<td>PERFORM_SHUNT</td>
</tr>
<tr>
<td></td>
<td>DFHRMROU</td>
<td>PERFORM_UNSHUNT</td>
</tr>
<tr>
<td>RMDE</td>
<td>DFHRMR1S</td>
<td>START_DELIVERY</td>
</tr>
<tr>
<td></td>
<td>DFHRMR1D</td>
<td>DELIVER_RECOVERY</td>
</tr>
<tr>
<td></td>
<td>DFHRMR1E</td>
<td>END_DELIVERY</td>
</tr>
<tr>
<td></td>
<td>DFHRMR1D</td>
<td>DELIVER_FORGET</td>
</tr>
<tr>
<td>RMKP</td>
<td>DFHRMR1K</td>
<td>TAKE_KEYPOINT</td>
</tr>
</tbody>
</table>
Table 84. Call back format owned by the Recovery Manager domain (continued)

<table>
<thead>
<tr>
<th>Format</th>
<th>Calling module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>RMLK</td>
<td>DFHRMLSP</td>
<td>PERFORM_PRELOGGING</td>
</tr>
<tr>
<td></td>
<td>DFHRMLSP</td>
<td>PERFORM_PREPARE</td>
</tr>
<tr>
<td></td>
<td>DFHRMLSD</td>
<td>REPLY_DO_COMMIT</td>
</tr>
<tr>
<td></td>
<td>DFHRMLSD</td>
<td>SEND_DO_COMMIT</td>
</tr>
<tr>
<td></td>
<td>DFHRMLSO</td>
<td>PERFORM_COMMIT</td>
</tr>
<tr>
<td></td>
<td>DFHRMLSS</td>
<td>PERFORM_SHUNT</td>
</tr>
<tr>
<td></td>
<td>DFHRMLSU</td>
<td>PERFORM_UNSHUNT</td>
</tr>
</tbody>
</table>

In the descriptions of the formats that follow, the “input” parameters are input not to Recovery Manager domain, but to the domain being called by the Recovery Manager. Similarly, the “output” parameters are output by the domain that was called by Recovery Manager domain, in response to the call.

**RMRO gate, PERFORM_COMMIT function**

This function requires the Recovery Manager client to perform phase 2 of syncpoint processing.

**Input parameters**

**WORK_TOKEN**

The Recovery Manager client’s work token for the syncpointing unit of work.

**CONTINUE**

A parameter specifying whether the current transaction will continue into a following unit of work. It can have any one of these values:

YES | NO

**UOW_STATUS**

The status of the current unit of work. It can have any one of these values:

FORWARD | BACKWARD

**RESTART**

An optional parameter specifying whether a backing out transaction will be restarted. It can have any one of these values:

YES | NO

**Output parameters**

**FORGET_RECORD**

A value specifying whether all obligations to this Recovery Manager client have been discharged. It can have any one of these values:

YES | NO

**RESPONSE**

is the Recovery Manager client’s response to the call. It can have any one of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

**RMRO gate, PERFORM_PREPARE function**

This function requires the Recovery Manager client to perform phase 1 of syncpoint processing.
Input parameters

WORK_TOKEN
The Recovery Manager client’s work token for the syncpointing unit of work.

CONTINUE
A parameter specifying whether the current transaction will continue into a following unit of work. It can have any one of these values:
YES|NO

Output parameters

VOTE
A value specifying the Recovery Manager client’s vote on the outcome of the syncpointing unit of work. It can have any one of these values:
YES|NO|NO_CONTINUE|READ_ONLY

RESPONSE
is the Recovery Manager client’s response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

RMRO gate, START_BACKOUT function
This function notifies the Recovery Manager client that backout processing is about to be performed for the unit of work.

Input parameters

WORK_TOKEN
The Recovery Manager client’s work token for the syncpointing unit of work.

CONTINUE
A parameter specifying whether the current transaction will continue into a following unit of work. It can have any one of these values:
YES|NO

REMOVE
A parameter specifying whether or not the backout is due to an invocation of the REMOVE function of the RMRE gate. It can have any one of these values:
YES|NO

Output parameters

RESPONSE
is the Recovery Manager client’s response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

RMRO gate, DELIVER_BACKOUT_DATA function
This function requires the Recovery Manager client process backout data from the system log for the unit of work.

Input parameters

WORK_TOKEN
The Recovery Manager client’s work token for the syncpointing unit of work.
Recovery Manager Domain (RM)

**DATA** A buffer containing the data previously logged with BACKWARD_DATA(YES) via the APPEND function of the RMRE gate.

**RESOURCE_ID** An optional parameter specifying the name of the resource with which the logged data is associated.

**CONTINUE** A parameter specifying whether the current transaction will continue into a following unit of work. It can have any one of these values:

- YES
- NO

**FORWARD_DATA** A parameter specifying whether or not the data was originally logged as FORWARD_DATA. It can have any one of these values:

- YES
- NO

**REMOVE** A parameter specifying whether or not the backout is due to an invocation of the REMOVE function of the RMRE gate. It can have any one of these values:

- YES
- NO

**CLUSTER_ID** A buffer to receive a symbolic name identifying the resource.

**LOCAL_ACCESS_ID** A buffer to receive the specific name of the resource.

**Output parameters**

**KEEP** A value specifying whether the backout action failed, implying the record should be kept and not forgotten. It can have any one of these values:

- YES
- NO

**RESPONSE** is the Recovery Manager client’s response to the call. It can have any one of these values:

- OK
- EXCEPTION
- DISASTER
- INVALID
- KERNERROR
- PURGED

**RMRO gate, END_BACKOUT function**

This function notifies the Recovery Manager client that backout processing has completed for the unit of work.

**Input parameters**

**WORK_TOKEN** The Recovery Manager client’s work token for the syncpointing unit of work.

**CONTINUE** A parameter specifying whether the current transaction will continue into a following unit of work. It can have any one of these values:

- YES
- NO

**REMOVE** A parameter specifying whether or not the backout is due to an invocation of the REMOVE function of the RMRE gate. It can have any one of these values:

- YES
- NO
Output parameters

RESPONSE
is the Recovery Manager client’s response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

RMRO gate, PERFORM_SHUNT function
This function notifies the Recovery Manager client that the unit of work is about to shunt.

Input parameters

WORK_TOKEN
The Recovery Manager client’s work token for the syncpointing unit of work.

CONTINUE
A parameter specifying whether the current transaction will continue into a following unit of work. It can have any one of these values:
YES|NO

Output parameters

NEXT_WORK_TOKEN
A value for the Recovery Manager client’s work token in the following unit of work.

RESPONSE
is the Recovery Manager client’s response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

RMRO gate, PERFORM_UNSHUNT function
This function notifies the Recovery Manager client that the unit of work is unshunting.

Input parameters

WORK_TOKEN
The Recovery Manager client’s work token for the syncpointing unit of work.

Output parameters

RESPONSE
is the Recovery Manager client’s response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

RMDE gate, START_DELIVERY function
This function notifies the Recovery Manager client that system recovery processing is about to be performed.

Input parameters
None
Recovery Manager Domain (RM)

Output parameters

RESPONSE
is the Recovery Manager client’s response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

RMDE gate, DELIVER_RECOVERY function

This function requires the Recovery Manager client to process recovery data from the system log.

Input parameters

RESOURCE_ID
An optional parameter specifying the name of the resource with which the logged data is associated.

DATA A buffer containing the data previously logged with BACKWARD_DATA(YES) via the APPEND function of the RMRE gate.

FORWARD_DATA
A parameter specifying whether or not the data was originally logged as FORWARD_DATA. It can have any one of these values:
YES|NO

BACKWARD_DATA
A parameter specifying whether or not the data was originally logged as BACKWARD_DATA. It can have any one of these values:
YES|NO

KEYPOINT
A parameter specifying whether or not the data was logged as part of a keypoint. It can have any one of these values:
YES|NO

BACKED_OUT
A parameter specifying whether or not the update the data is associated with backed out. It can have any one of these values:
YES|NO

UOW A parameter specifying whether the data is related to a particular unit of work. It can have any one of these values:
YES|NO

UOW_STATUS
An optional parameter specifying the status of unit of work the data belongs to (if any). It can have any one of these values:
FORWARD|BACKWARD|IN_DOUBT|IN_FLIGHT

LOCAL_UOW_ID
An optional parameter specifying the local UOWID of the unit of work the data belongs to (if any).

Output parameters

RESPONSE
is the Recovery Manager client’s response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED
**RMDE gate, END_DELIVERY function**

This function notifies the Recovery Manager client that all recovery information from the system log has been processed.

**Input parameters**
None

**Output parameters**

RESPONSE
is the Recovery Manager client’s response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**RMDE gate, DELIVER_FORGET function**

This function notifies the Recovery Manager client that FORGET processing is required for some resource in a unit of work.

**Input parameters**

LOCAL_ACCESS_ID
A parameter specifying the name of the resource associated with the forget processing.

UOW
It can only have the value YES.

UOW_STATUS
The status of the unit of work. It can have any one of these values:
FORWARD|BACKWARD|IN_Doubt|IN_FLIGHT

LOCAL_UOW_ID
The local UOWID of the unit of work.

**Output parameters**

RESPONSE
is the Recovery Manager client’s response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**RMKP gate, TAKE_KEYPOINT function**

This function requires the Recovery Manager client to perform keypoint processing.

**Input parameters**

SHUTDOWN
A parameter specifying whether the keypoint is the warm keypoint taken during shutdown or an activity keypoint. It can have any one of these values:
YES|NO

**Output parameters**

RESPONSE
is the Recovery Manager client’s response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED
RMLK gate, PERFORM_PRELOGGING function

This function notifies the Recovery Manager client that phase 1 of syncpoint processing is about to occur.

Input parameters

**RMC_TOKEN**
The Recovery Manager client’s token associated with the Recovery Manager Link object.

**INITIATOR**
A parameter specifying whether the remote system is the initiator of the syncpoint. It can have any one of these values:
- YES
- NO

**COORDINATOR** (YES|NO)
A parameter specifying whether the remote system is the coordinator of the distributed unit of work. It can have any one of these values:
- YES
- NO

Output parameters

**RESPONSE**
is the Recovery Manager client’s response to the call. It can have any one of these values:
- OK
- EXCEPTION
- DISASTER
- INVALID
- KERNERROR
- PURGED

RMLK gate, PERFORM_PREPARE function

This function requires the Recovery Manager client perform phase 1 of syncpoint processing.

Input parameters

**RMC_TOKEN**
The Recovery Manager client’s token associated with the Recovery Manager Link object.

**CONTINUE**
A parameter specifying whether the current transaction will continue into a following unit of work. It can have any one of these values:
- YES
- NO

**SYSTEM**
A parameter specifying whether PERFORM_PREPARE call is part of a syncpoint or the result of EXEC CICS ISSUE PREPARE. It can have any one of these values:
- YES
- NO

**RECOVERY_STATUS**
A parameter specifying whether recoverable work has taken place as part of the distributed unit of work on the remote system. It can have any one of these values:
- NECESSARY
- UNNECESSARY
- SYNC_LEVEL_1

Output parameters

**VOTE**
A value specifying the Recovery Manager client’s vote on the outcome of the syncpointing unit of work. It can have any one of these values:
- YES
- NO
- NO_CONTINUE
- READ_ONLY
- HEURISTIC_MIXED
**RESPONSE**

is the Recovery Manager client’s response to the call. It can have any one of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

**RMLK gate, REPLY_DO_COMMIT function**

This function requires the Recovery Manager client communicate the result of this systems phase 1 syncpoint processing to the coordinating system, and obtain the outcome of the distributed unit of work.

**Input parameters**

**RMC_TOKEN**

The Recovery Manager client’s token associated with the Recovery Manager Link object.

**CONTINUE**

A parameter specifying whether the current transaction will continue into a following unit of work. It can have any one of these values:

YES | NO

**SINGLE_UPDATER**

A parameter specifying whether the single updater optimization is being performed. It can have any one of these values:

YES | NO

**Output parameters**

**ACCESSIBLE**

A value specifying whether communication with the remote system failed. It can have any one of these values:

YES | NO | SHUNTED

**VOTE**

A value specifying the outcome of the syncpointing unit of work. It can have any one of these values:

YES | NO | NO_CONTINUE | READ_ONLY | HEURISTIC_MIXED

**RESPONSE**

is the Recovery Manager client’s response to the call. It can have any one of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

**RMLK gate, SEND_DO_COMMIT function**

This function requires the Recovery Manager client communicate the result of this systems phase 1 syncpoint processing to the last agent system, and obtain the outcome of the distributed unit of work.

**Input parameters**

**RMC_TOKEN**

The Recovery Manager client’s token associated with the Recovery Manager Link object.

**CONTINUE**

A parameter specifying whether the current transaction will continue into a following unit of work. It can have any one of these values:

YES | NO
Recovery Manager Domain (RM)

SINGLE_UPDATER
A parameter specifying whether the single updater optimization is being performed. It can have any one of these values:
YES|NO

Output parameters

ACCESSIBLE
A value specifying whether communication with the remote system failed. It can have any one of these values:
YES|NO|SHUNTED

VOTE
A value specifying the outcome of the syncpointing unit of work. It can have any one of these values:
YES|NO|NO_CONTINUE|READ_ONLY|HEURISTIC_MIXED

RESPONSE
is the Recovery Manager client’s response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

RMLK gate, PERFORM_COMMIT function

This function requires the Recovery Manager client perform phase 2 of syncpoint processing.

Input parameters

RMC_TOKEN
The Recovery Manager client’s token associated with the Recovery Manager Link object.

CONTINUE
A parameter specifying whether the current transaction will continue into a following unit of work. It can have any one of these values:
YES|NO

SINGLE_UPDATER
A parameter specifying whether the single updater optimization is being performed. It can have any one of these values:
YES|NO

UOW_STATUS
The status of the syncpointing unit of work. It can have any one of these values:
FORWARD|BACKWARD

RESTART
An optional parameter specifying whether a backing out transaction will be restarted. It can have any one of these values:
YES|NO

COORDINATOR
A parameter specifying whether the remote system is the coordinator of the distributed unit of work. It can have any one of these values:
YES|NO

INITIATOR
A parameter specifying whether the remote system is the initiator of the syncpoint. It can have any one of these values:
PRESUMPTION
A parameter specifying whether the remote system assumes the presume
abort or presume nothing protocols. It can have any one of these values:
ABORT|NOTHING

RECOVERY_STATUS
A parameter specifying whether recoverable work has taken place as part
of the distributed unit of work on the remote system. It can have any one
of these values:
NECESSARY|UNNECESSARY|SYNC_LEVEL_1

Output parameters

ACCESSIBLE
A parameter specifying that the communications link to the remote system
has failed. It can have any one of these values:
YES|NO|SHUNTED

FORGET
A parameter specifying whether all obligations to the remote system with
respect to recovery have been discharged. It can have any one of these
values:
YES|NO

PASS
A parameter specifying whether an equivalent Recovery Manager Link
object should be created in the following unit of work. It can have any one
of these values:
YES|NO

ABEND
A parameter specifying whether an abend occurred during the
PERFORM_COMMIT call-back. It can have any one of these values:
YES|NO

NEXT_RECOVERY_STATUS
A parameter specifying the initial RECOVERY_STATUS of the Recovery
Manager Link object created in the following unit of work as a result of
PASS(YES). It can have any one of these values:
NECESSARY|UNNECESSARY|SYNC_LEVEL_1|DEFAULT

RESPONSE
is the Recovery Manager client’s response to the call. It can have any one
of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

RMLK gate, PERFORM_SHUNT function
This function notifies the Recovery Manager client that the unit of work is
shunting.

Input parameters

RMC_TOKEN
The Recovery Manager client’s token associated with the Recovery
Manager Link object.

CONTINUE
A parameter specifying whether the current transaction will continue into a
following unit of work. It can have any one of these values:
Recovery Manager Domain (RM)

**RECOVERY_STATUS**
A parameter specifying whether recoverable work has taken place as part of the distributed unit of work on the remote system. It can have any one of these values:
NECESSARY|UNNECESSARY|SYNC_LEVEL_1

**Output parameters**

**FORGET**
A parameter specifying whether all obligations to the remote system with respect to recovery have been discharged. It can have any one of these values:
YES|NO

**RESPONSE**
is the Recovery Manager client’s response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**RMLK gate, PERFORM_UNSHUNT function**
This function notifies the Recovery Manager client that the unit of work is unshunting.

**Input parameters**

**LINK_TOKEN**
A token identifying the Recovery Manager Link object to be unshunted.

**LOGNAME_BUFFER**
A parameter specifying a buffer containing the logname of the remote system.

**REMOTE_ACCESS_ID_BUFFER**
A buffer containing the netname of the remote system, or the name of the External Resource Manager.

**LINK_ID_BUFFER**
A buffer containing the termid of the session to the remote system, or the External Resource Manager qualifier.

**LINK_IDSOURCE**
An optional parameter specifying whether the local or remote system allocated the session. It can have any one of these values:
LOCAL|REMOTE

**Output parameters**

**RESPONSE**
is the Recovery Manager client’s response to the call. It can have any one of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**Modules**

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHRMCD</td>
<td>Handles the functions of the RMCD gate.</td>
</tr>
<tr>
<td>DFHRMCD1</td>
<td>Initialises the Client Directory Class.</td>
</tr>
</tbody>
</table>
DFHRMCD2 Quiesces the Client Directory Class.
DFHRMCI2 Sets the callback gate of a Recovery Manager client.
DFHRMCI3 Waits for a registered Recovery Manager client to set its callback gate.
DFHRMCI4 Waits for a registered Recovery Manager client to set its callback gate and calls it with a given parameter list.
DFHRMDM Recovery Manager domain initialization and termination. Handles the DMDM and RMDM gate functions.
DFHRMUTL Recovery Manager batch utility.
DFHRMDU0 Formats the Recovery Manager control blocks.
DFHRMDU2 Starts a browse of all Recovery Manager client work tokens during dump formatting.
DFHRMDU3 Gets the next Recovery Manager client work token during dump formatting.
DFHRMDU4 Ends a browse of all Recovery Manager client work tokens during dump formatting.
DFHRMLK1 Initialises the Recovery Manager Link Class.
DFHRMLK2 Handles the INITIATE_RECOVERY function of the RMLN gate.
DFHRMLK3 Inquires whether a Logname is in-use by any Recovery Manager gate.
DFHRMLK4 Handles the CLEAR_PENDING function for a particular Recovery Manager Link.
DFHRMLK5 Collects statistics from the Recovery Manager Link Class.
DFHRMLKQ Quiesces the Recovery Manager Link Class.
DFHRMLN Handles the functions of the RMLN gate.
DFHRMLSD Asks the coordinator Recovery Manager Link to decide the outcome of the unit of work.
DFHRMLSF Determines the reason for a unit of work being in doubt.
DFHRMLSO Commits the Recovery Manager Links for a unit of work.
DFHRMLSP Prepares the Recovery Manager Links for a unit of work.
DFHRMLSS Shunts the Recovery Manager Links for a unit of work.
DFHRMLSU Unshunts the Recovery Manager Links for a unit of work.
DFHRML1D Reconstructs Recovery Manager Links from log records.
DFHRMNM Handles the functions of the RMNM gate.
DFHRMNM1 Initialises the Recovery Manager Lognames Class.
DFHRMNS1 Initialises the Recovery Manager Logname Set Class.
DFHRMNS2 Quiesces the Recovery Manager Logname Set Class.
DFHRMOFI Initialises a Recovery Manager Object Factory.
DFHRMRO Handles the functions of the RMRO gate.
DFHRMROO Handles FORGET processing for Recovery Manager Resource Owners.
DFHRMROS Shunts a Recovery Manager Resource Owner.
DFHRMROU Unshunts a Recovery Manager Resource Owner.
DFHRMROV Handles AVAIL processing for Recovery Manager Resource Owners.
DFHRMRO1 Initialises the Recovery Manager Resource Owner Class.
DFHRMRO2 Signals start_backout to a Recovery Manager Resource Owner.
## Recovery Manager Domain (RM)

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHRMRO3</td>
<td>Delivers backout data to a Recovery Manager Resource Owner.</td>
</tr>
<tr>
<td>DFHRMRO4</td>
<td>Signals end_backout to a Recovery Manager Resource Owner.</td>
</tr>
<tr>
<td>DFHRMR1D</td>
<td>Delivers recovery data to a Recovery Manager Resource Owner.</td>
</tr>
<tr>
<td>DFHRMR1E</td>
<td>Signals end of recovery to a Recovery Manager Resource Owner.</td>
</tr>
<tr>
<td>DFHRMR1K</td>
<td>Signals a keypoint to a Recovery Manager Resource Owner.</td>
</tr>
<tr>
<td>DFHRMR1S</td>
<td>Signals start of recovery to a Recovery Manager Resource Owner.</td>
</tr>
<tr>
<td>DFHRMSL</td>
<td>Handles the functions of the RMSL gate.</td>
</tr>
<tr>
<td>DFHRMSLF</td>
<td>Forces the System Log.</td>
</tr>
<tr>
<td>DFHRMSLJ</td>
<td>Checks for Chain independence during recovery.</td>
</tr>
<tr>
<td>DFHRMSLL</td>
<td>Closes a Chain on the System Log.</td>
</tr>
<tr>
<td>DFHRMSLO</td>
<td>Opens a Chain on the System Log.</td>
</tr>
<tr>
<td>DFHRMSLV</td>
<td>Moves a Chain on the System Log.</td>
</tr>
<tr>
<td>DFHRMSLW</td>
<td>Writes a record to a Chain on the System Log.</td>
</tr>
<tr>
<td>DFHRMSL1</td>
<td>Initialises the Recovery Manager System Log Class.</td>
</tr>
<tr>
<td>DFHRMSL2</td>
<td>Starts a browse of a Chain on the System Log.</td>
</tr>
<tr>
<td>DFHRMSL3</td>
<td>Reads a Record from a Chain on the System Log.</td>
</tr>
<tr>
<td>DFHRMSL4</td>
<td>Ends a browse of a Chain on the System Log.</td>
</tr>
<tr>
<td>DFHRMSL5</td>
<td>Performs restart processing for Recovery Manager System Log Class.</td>
</tr>
<tr>
<td>DFHRMSL6</td>
<td>Schedules keypoint activity.</td>
</tr>
<tr>
<td>DFHRMSL7</td>
<td>Performs keypoint processing.</td>
</tr>
<tr>
<td>DFHRMST</td>
<td>Handles STST functions for Recovery Manager.</td>
</tr>
<tr>
<td>DFHRMSST1</td>
<td>Initializes the Recovery Manager Statistics Class.</td>
</tr>
<tr>
<td>DFHRMTRI</td>
<td>Formats Recovery Manager trace entries.</td>
</tr>
<tr>
<td>DFHRMUC</td>
<td>Creates a RMUW (unit of work) object.</td>
</tr>
<tr>
<td>DFHRMUO</td>
<td>Commits a unit of work.</td>
</tr>
<tr>
<td>DFHRMUW</td>
<td>Handles the functions of the RMUW gate.</td>
</tr>
<tr>
<td>DFHRMUWB</td>
<td>Handles data during backout of a unit of work.</td>
</tr>
<tr>
<td>DFHRMUWE</td>
<td>Handles activities when a unit of work is unshunted.</td>
</tr>
<tr>
<td>DFHRMUWF</td>
<td>Forces log records for a unit of work.</td>
</tr>
<tr>
<td>DFHRMUWH</td>
<td>Holds an RMUW object.</td>
</tr>
<tr>
<td>DFHRMUWJ</td>
<td>Forces a unit of work to take a unilateral decision.</td>
</tr>
<tr>
<td>DFHRMUWL</td>
<td>Handles notification that all remote remotes have finished processing.</td>
</tr>
<tr>
<td>DFHRMUWN</td>
<td>Schedules a unit of work to be unshunted.</td>
</tr>
<tr>
<td>DFHRMUWP</td>
<td>Handles notification that a local resource has become available.</td>
</tr>
<tr>
<td>DFHRMUWQ</td>
<td>Handles commit or backout of an unshunted, in doubt unit of work.</td>
</tr>
<tr>
<td>DFHRMUWS</td>
<td>Records the outcome of a unit of work during resynchronization.</td>
</tr>
<tr>
<td>DFHRMUWU</td>
<td>Records the local LU name.</td>
</tr>
<tr>
<td>DFHRMUWV</td>
<td>Handles notification that a local resource has become available.</td>
</tr>
<tr>
<td>DFHRMUWW</td>
<td>Writes a record belonging to a unit of work to the System Log.</td>
</tr>
<tr>
<td>DFHRMUW0</td>
<td>Releases an RMUW object.</td>
</tr>
<tr>
<td>DFHRMUW1</td>
<td>Initializes the Recovery Manager Unit of Work Class.</td>
</tr>
</tbody>
</table>
Recovery Manager Domain (RM)

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHRMUW2</td>
<td>Collects the Recovery Manager Unit of Work Class Statistics.</td>
</tr>
<tr>
<td>DFHRMUW3</td>
<td>Handles the INQUIRE_UOW_TOKEN function.</td>
</tr>
<tr>
<td>DFHRMU1C</td>
<td>Sets the Chain token for a unit of work.</td>
</tr>
<tr>
<td>DFHRMU1D</td>
<td>Handles log records of units of work during recovery.</td>
</tr>
<tr>
<td>DFHRMU1E</td>
<td>Signals that all records have been recovered from the System Log during recovery.</td>
</tr>
<tr>
<td>DFHRMU1F</td>
<td>Handles an in doubt wait timeout.</td>
</tr>
<tr>
<td>DFHRMU1J</td>
<td>Inquires whether all unit of work chains are disjoint.</td>
</tr>
<tr>
<td>DFHRMU1K</td>
<td>Keypoints a unit of work.</td>
</tr>
<tr>
<td>DFHRMU1L</td>
<td>Handle XMPP_FORCE_PURGE_INHIBIT_QUERY.</td>
</tr>
<tr>
<td>DFHRMU1N</td>
<td>Handle XMPP_FORCE_PURGE_INHIBIT_QUERY.</td>
</tr>
<tr>
<td>DFHRMU1Q</td>
<td>Handle the NOTIFY function of the TISR gate.</td>
</tr>
<tr>
<td>DFHRMU1R</td>
<td>Performs restart processing for Recovery Manager Unit of Work Class.</td>
</tr>
<tr>
<td>DFHRMU1S</td>
<td>Signals that recovery of log records is about to be performed.</td>
</tr>
<tr>
<td>DFHRMU1U</td>
<td>Process a unit of work after recovery.</td>
</tr>
<tr>
<td>DFHRMU1V</td>
<td>Requests time out interval notification for a unit of work.</td>
</tr>
<tr>
<td>DFHRMU1W</td>
<td>Cancels wait time out notification for a unit of work.</td>
</tr>
<tr>
<td>DFHRMVP1</td>
<td>Initializes the Recovery Manager Variable Length Subpool Class.</td>
</tr>
<tr>
<td>DFHRMXNE</td>
<td>Reattaches a transaction to process an unshunted unit of work.</td>
</tr>
<tr>
<td>DFHRMXN2</td>
<td>Schedules a keypoint.</td>
</tr>
<tr>
<td>DFHRMXN3</td>
<td>The keypoint program.</td>
</tr>
<tr>
<td>DFHRMXN4</td>
<td>Restarts the Recovery Manager Transaction Class.</td>
</tr>
<tr>
<td>DFHRMXN5</td>
<td>Increments Recovery Manager statistics for a Transaction.</td>
</tr>
</tbody>
</table>

Exits

None

Trace

The point IDs for the Recovery Manager domain are of the form RM xxxx the corresponding trace levels are RM 1, RM 2, and Exc.

For more information about the trace points, see the CICS Trace Entries. For more information about using traces in problem determination, see the CICS Problem Determination Guide.
Chapter 69. RRMS domain (RX)

The RRMS domain is responsible for managing interaction with OS/390 Recoverable Resource Management Services (RRMS) and in particular, Resource Recovery Services (RRS) which is a component of RRMS.

RRMS domain’s specific gates

Table 85 summarizes the RX domain’s specific gates. It shows the level-1 trace point IDs of the modules providing the functions for the gate, the functions provided by the gate, and whether or not the functions are available through the exit programming interface (XPI).

Table 85. RX domain’s specific gate

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>XPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>RXDM</td>
<td>RX 0101</td>
<td>INQUIRE_RRS</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>RX 0102</td>
<td>SET_PARAMETERS</td>
<td>NO</td>
</tr>
<tr>
<td>RXUW</td>
<td>RX 0401</td>
<td>PUT_CLIENT_REQUEST</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>RX 0402</td>
<td>GET_CLIENT_REQUEST</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INQUIRE</td>
<td>NO</td>
</tr>
</tbody>
</table>

RXDM gate, INQUIRE_RRS function

The INQUIRE_RRS function of the RXDM gate is used to determine the status of CICS’s interface with OS/390 Recoverable Resource Management Services (RRMS).

Output Parameters

OPEN  Returns YES or NO to indicate if the interface with RRMS is open.

[RESTART_STATE]

Returns a value to indicate the state of restart processing with Resource Recovery Services (RRS). One of these values is returned:

- NOT_STARTED  Restart processing has not started
- STARTING    Restart is in progress
- COLD        Restart processing is complete, and RRS was cold started.
- WARM        Restart processing is complete, and RRS was warm started.

RESPONSE is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

RXDM gate, SET_PARAMETERS function

The SET_PARAMETERS function of the RXDM gate is used to pass the values of relevant System Initialization parameters to the domain.

Input Parameters

RRMS

Specifies the value of the RRMS System Initialization Parameter. It can have one of these values:

YES|NO
**Output Parameters**

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID |
KERNERROR | PURGED

**RXUW gate, PUT_CLIENT_REQUEST function**

The PUT_CLIENT_REQUEST function of the RXDM gate is used to associate a request from a client with an RRS Unit of Recovery (UR).

**Input parameters**

**TRANSACTION_ID**

The transaction id associated with the request. This parameter is used to correlate successive requests for the same transaction instance.

**USERID**

The userid associated with the request. This parameter is used to correlate successive requests for the same transaction instance.

**CONNECTION**

The connection on which the client request was received. This parameter is used to identify the source of the request in any messages that are issued.

**CONTEXT_TOKEN**

The token representing the RRMS context for which the request is issued.

**URID**

The identifier of the RRS Unit of Recovery associated with the context.

**PASS_TOKEN**

A token used to protect against unauthorised use of the context token and URID.

**CLIENT_TOKEN**

A token representing the client of the UR.

**CLIENT_TYPE**

Indicates the type of client of the transaction. The only permissible value is TERMINAL.

**Output parameters**

**NEW_UR**

Indicates whether a new UR has been created for this request. It can have one of these values:

YES indicates that a new UR has been created
NO indicates that the request was associated with an existing UR

**UR_TOKEN**

is the token by which the UR associated with the request is known by the RX domain.

**TRANSACTION_NUMBER**

The transaction number of the transaction associated with the request.

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID |
KERNERROR | PURGED
RXUW gate, GET_CLIENT_REQUEST function

The GET_CLIENT_REQUEST function of the RXDM gate is used to suspend a transaction until the PUT_CLIENT_REQUEST is issued for the same Unit of Recovery.

Input parameters

UR_TOKEN
is the token by which the UR associated with the request is known by the RX domain.

[TIMEOUT]
The time (in seconds) for which the transaction should be suspended. If this parameter is omitted, the transaction will be suspended indefinitely.

Output parameters

CLIENT_TOKEN
A token representing the client of the UR.

CLIENT_TYPE
Indicates the type of client of the transaction. The only possible value is TERMINAL.

RESPONSE
is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are

SYNCPOINT RRS has requested a syncpoint
BACKOUT RRS has requested rollback
RACE RRS has requested syncpoint or rollback and a client request has been received at the same time

[REASON] is also returned when RESPONSE is PURGED. Possible values are

TASKCANCELLED The task has been purged
TIMEDOUT The request has timed out

RXUW gate, INQUIRE function

The INQUIRE function requests attributes of a Unit of Recovery.

Input parameters

UR_TOKEN
is the token which identifies the Unit of Recovery.

Output parameters

[URID]
The identifier of the Unit of Recovery used by RRMS.

RESPONSE
is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED
## Modules

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHRXDM</td>
<td>RX domain management and global functions.</td>
</tr>
<tr>
<td>DFHRXUW</td>
<td>RX domain unit-of-work related functions.</td>
</tr>
<tr>
<td>DFHRXSVC</td>
<td>RX domain SVC code for RRMS authorized interface.</td>
</tr>
<tr>
<td>DFHRXXRG</td>
<td>RX domain Registration Services exits.</td>
</tr>
<tr>
<td>DFHRXXRM</td>
<td>RX domain Resource Manager exits.</td>
</tr>
<tr>
<td>DFHRXDUF</td>
<td>RX domain dump formatting.</td>
</tr>
<tr>
<td>DFHRXTRI</td>
<td>RX domain trace interpretation.</td>
</tr>
</tbody>
</table>

## Exits

None

## Trace

The point IDs for the RRMS domain are of the form RX xxxx the corresponding trace levels are RX 1, RX 2, and Exc.

An overall description of DL/I database support is given in "Chapter 26. DL/I database support" on page 343. This section gives information that is specific to remote DL/I.

Design overview

This section outlines what you must do to define remote DL/I support, and describes the functions of remote DL/I.

System definition

For a CICS system that supports only remote databases you must, in addition to providing the usual definitions that are required for function shipping, code a PSB directory (PDIR) using the DFHDLPSB macro. Every PDIR entry must have SYSIDNT specified. The PDIR system initialization parameter must be coded specifying the suffix of the PDIR.

DL/I PSB scheduling

When a CICS task requests the scheduling of a DL/I PSB by means of an EXEC DLI SCHEDULE request or DL/I PCB call, and the request is for a remote PSB, control is passed to DFHDLIRP. DFHDLIRP allocates a remote scheduling block (RSB) and issues a DFHIS TYPE=CONVERSE macro to ship the scheduling request to the owning system.

Database calls

For a remote DL/I database call, a DFHIS TYPE=CONVERSE macro is issued to ship the request to the owning system. The return codes are passed back to the user in the user interface block (UIB).

DL/I PSB termination

If a remote PSB is terminated, the actions performed are:
1. Free the RSB and local program communication block (PCB) storage.
2. If the DL/I PSB termination was not caused by a CICS syncpoint, request one now.

Control blocks

Figure 82 on page 974 illustrates some of the control blocks used to support remote DL/I.
The DL/I interface parameter list (DLP) is described in "DL/I interface parameter list (DLP)" on page 345.

The remote PSB directory (PDIR) contains an entry for each remote PSB that can be used from an application program.

The remote scheduling block (RSB) (see Figure 83 on page 977) is acquired when a CICS task issues a PSB schedule request for a remote PSB. The RSB is freed when the task issues a SYNCPOINT or a DLI TERM request.
See the CICS Data Areas manual for a detailed description of these control blocks.
Chapter 71. RequestStream domain (RZ)

The RequestStream domain provides connectivity between elements of the Corbaserver and EJB components in a sysplex to allow transfer of GIOP requests from a requester to a request processor, and to permit appropriate workload balancing of the deployment of those requests.

It provides basic management functions for RequestStreams:
- Create RequestStream
- Destroy RequestStream
- Publicise RequestStream (see PublicId)
- Join requeststream
- Leave RequestStream
- Send/receive requests on RequestStreams

The RequestStream domain also manages Transports, which are single communication mechanisms that transfer bytes from one task to another in the sysplex. They occur in different types, depending upon which technology is appropriate for the route that needs to be taken. For example, there are “InStore” transports, “MRO” transports and “Socket” transports.

RequestStream domain’s specific gates

Table 86 summarizes the RequestStream domain’s specific gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, and the functions provided by the gates. None of the functions are available through the exit programming interface (XPI).

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>XPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>RZSO</td>
<td>RZ 0110</td>
<td>CREATE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>RZ 0111</td>
<td>SEND_REQUEST</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RECEIVE_REPLY</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LEAVE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JOIN</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IS_ID_LOCAL</td>
<td>NO</td>
</tr>
<tr>
<td>RZTA</td>
<td>RZ 0120</td>
<td>RECEIVE_REQUEST</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>RZ 0121</td>
<td>SEND_REPLY</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GET_PUBLIC_ID</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GET_SERVER_DATA</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GET_CURRENT</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GET_JOIN_DATA</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TERMINATE</td>
<td>NO</td>
</tr>
<tr>
<td>RZRT</td>
<td>RZ 0170</td>
<td>SET_EXIT_PROGRAM</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>RZ 0171</td>
<td></td>
<td>NO</td>
</tr>
<tr>
<td>RZRJ</td>
<td>RZ 0180</td>
<td>PERFORM_JOIN</td>
<td>NO</td>
</tr>
</tbody>
</table>

RZSO gate, CREATE function

Create a RequestStream and return a (local region) source RequestStream token for it.
RequestStream domain (RZ)

The target process(or) is identified either by USERID and TRANID or by HOST_IP_ADDRESS and PORT_NUMBER. Precisely one of these groups must be provided.

(The HOST_IP_ADDRESS is a character string as expected by the internal sockets domain interfaces.)

The SERVER_DATA may be retrieved at the target (RZTA) interface and is copied (and fixed) on this call.

The response is (exception, service_not_available) if it is not possible to resolve the target, or to set up a connection to the target. (Success does not guarantee that this exception will not occur on the SEND function.)

The response is (exception, target_unknown) if the HOST_IP_ADDRESS character string is malformed (as detected by the sockets domain interfaces).

The response is invalid when the parameters are badly formed, in particular if there is not the right combination of target identification parameters.

Input parameters

[USERID]
Userid under which the requests are to be processed.

[TRANID]
TranId of the transaction which runs the target processor.

[HOST_IP_ADDRESS]
Identification of the target which is to process the requests.

[PORT_NUMBER]
Further identification of the target.

[SERVER_BLOCK]
Data associated with the RequestStream available at the target end by the server using the RZTA interface.

[SSL_REQUIRED]
Values: YES, NO
Whether to use SSL on a socket transport. Otherwise ignored.

[CERTIFICATE_NAME]
Further information for an SSL socket transport. Otherwise ignored.

Output parameters

RS_TOKEN
Token by which RequestStream is identified on all subsequent requests from this task on this region.

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

REASON
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>SERVICE_NOT_AVAILABLE TARGET_UNKNOWN</td>
</tr>
</tbody>
</table>
**RZSO gate, SEND_REQUEST function**

The source RequestStream token and the request (coded as a RUEI or as a contiguous data block) is passed as input. Either a ruei or a block must be used, not both. If this is not so then an invalid response is returned.

The request is deemed to be entire and may be presented to the target. Data may be transported across the transport mechanism during this call. The request may be of zero length, this does not imply that nothing is transported.

If the source RequestStream token does not exist (in the local region) the response (exception, rs_token_unknown) is returned.

If a transport mechanism fails to respond, or is not functional, then the response (exception, service_not_available) is returned. If it fails during transmission then (exception, transport_failure) is returned. The distinction is that in the former case there is no transport mechanism and in the latter there is still one (albeit inoperational).

**Input parameters**

**RS_TOKEN**
Token returned on CREATE by which RequestStream is identified.

**[REQUEST_RUEI]**
Reusable-extended-Illiffe Vector which describes contiguous bytes to send as a request, supplied in possibly discontiguous blocks. Exclusive with REQUEST_BLOCK.

**[REQUEST_BLOCK]**
Request data to send described as a single block. Exclusive with REQUEST_RUEI.

**Output parameters**

**RESPONSE**
is the domain’s response to the call. It can have any of these values: OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**REASON**
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>RS_TOKEN_UNKNOWN TRANSPORT_FAILURE SERVICE_NOT_AVAILABLE</td>
</tr>
</tbody>
</table>

**RZSO gate, RECEIVE_REPLY function**

A reply is returned (blocks until one is available).

The "reply_buffer" is set with as much data as will fit. The total number of bytes possible to be transferred is returned in the buffer as well as as many bytes of the reply that will fit. The "reply_buffer" size is not set although the number of bytes to be set in the buffer is set as usual. The call can be re-issued before any other calls to the RequestStreams source interface (using this token) to receive the remaining reply bytes. Reply data is not redelivered.

The final call to "receive_reply" indicates that all of the data to be transferred fit within the buffer passed. The final call to "receive_reply" will allow the transport to change direction, which means that further calls to "receive_reply" will be an
RequestStream domain (RZ)

error and will give a response "(exception, transport_failure)". If a "notify" callback has been called before this function is issued then "receive_reply" will be satisfied. The status of the "notify" callback is not affected by this call ("notify" is disabled automatically when it is issued) except that if there is data to deliver, and a "notify" callback is enabled with "listen", and has not been called, then the "notify" callback will be called as part of the processing of this function. This can be avoided, if necessary, by issuing a "cancel" beforehand.

If the RequestStream token is not known then the response is "(exception, rs_token_unknown)". If the transport service is not open then the response "(exception, service_not_available)" is returned. If the transport service is still open but gives some sort of error (for example, is not in the correct state to receive a reply) then the response is "(exception, transport_failure)". If the request processor cannot, for some reason, process the request, and returned an exception, or failed during execution, then the response is "(exception, request_processor_failure)". Processor failure, when it can be ascertained, takes precedence over transport failure.

Input parameters

RS_TOKEN
Token returned on CREATE by which RequestStream is identified.

[REPLY_BUFFER]
Buffer in which reply bytes are assembled.

[MINIMUM_DATA_LENGTH]
Minimum amount of data to accept (multiple transfers may occur until this amount is received).

Output parameters

[REPLY_DATA_LENGTH]
Total length of reply (even if not all received in one call).

RESPONSE
is the domain’s response to the call. It can have any of these values: OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

REASON
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>RS_TOKEN_UNKNOWN SERVICE_NOT_AVAILABLE TRANSPORT_FAILURE</td>
</tr>
<tr>
<td></td>
<td>REQUEST_PROCESSOR_FAILURE</td>
</tr>
</tbody>
</table>

RZSO gate, LEAVE function

Remove this source from its RequestStream. The RequestStream is modified so that the "rs_token" (which must denote a source end of the RequestStream) is no longer valid. (A token value may or may not be reissued by "RZ" on another "create" or "join" request - however the caller must not rely on its value after "leave".)

If the RequestStream "rs_token" is valid but does not denote a source end of a RequestStream known in this region the response "(exception, rs_token_not_source)" is returned.

If the transport mechanism fails then the response "(exception, transport_failure)" is returned, however the RequestStream source token is still invalidated.
RequestStream domain (RZ)

If a "notify" is enabled for this RequestStream token (with a "listen" call) then the "notify" callback is called with the "CLOSE" parameter prior to removing the RequestStream source.

**Input parameters**

**RS_TOKEN**
The token returned on CREATE identifying the RequestStream for this region.

**Output parameters**

**RESPONSE**

is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**REASON**

is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>RS_TOKEN_UNKNOWN TRANSPORT_FAILURE RS_TOKEN_NOT_SOURCE</td>
</tr>
</tbody>
</table>

**RZSO gate, JOIN function**

"Join" the RequestStream identified by the "public_id".

If the required transport mechanism is not available, or fails in use, the appropriate exception is returned as for "create".

If the RequestStream, identified by the "public_id", does not exist (because the target end does not exist) then this call does not detect this. Instead a new request processor will be created implicitly just as for "create".

The "userid" (if supplied) must match that used on the "create", otherwise an error may occur later in (Request Processor) processing. This is not detected at this call. The "tranid" and the "server_data" is supplied in case the RequestStream is recreated on this call, otherwise they are ignored. They may be omitted as in "create".

If the "public_id" is not valid, or cannot be interpreted then the response "(exception, public_id_invalid)" will be returned.

The "rs_token" for the local source RequestStream is returned as result.

**Input parameters**

**PUBLIC_ID**

Public RequestStream Identifier -- valid for all participating regions in the logical server -- of the target RequestStream, which may be in a separate region.

**[USERID]**

The security userid under which the target RequestStream should be executing.

**[TRANID]**

The transaction identifier for the target RequestStream task.

**[SERVER_BLOCK]**

The server data that was specified. This parameter and Tranid are used...
only if a new target RequestStream needs to be created owing to the omission of the target RequestStream processor.

**Output parameters**

**RS_TOKEN**
- The local region token by which the source RequestStream (that connects to the target) is known on all subsequent calls.

**RESPONSE**
- is the domain’s response to the call. It can have any of these values:
  - OK
  - EXCEPTION
  - DISASTER
  - INVALID
  - KERNERROR
  - PURGED

**REASON**
- is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>SERVICE_NOT_AVAILABLE TRANSPORT_FAILURE PUBLIC_ID_INVALID</td>
</tr>
</tbody>
</table>

**RZSO gate, IS_ID_LOCAL function**

Return "yes" if the "public_id" refers to the local region as the target region for the RequestStream. Otherwise "no".

This is a non-blocking call that gives the response "(exception, public_id_invalid)" if the "public_id" is detectably invalid (e.g. has the wrong format) and otherwise has no exceptions.

**Input parameters**

**PUBLIC_ID**
- Public RequestStream Identifier -- valid for all participating regions in the logical server -- of a target RequestStream, which may be in a separate region.

**Output parameters**

**LOCAL**
- Values:
  - YES | NO indicating whether the identified PUBLIC_ID is in the local region or not. There is no guarantee that it exists in either case.

**RESPONSE**
- is the domain’s response to the call. It can have any of these values:
  - OK
  - EXCEPTION
  - DISASTER
  - INVALID
  - KERNERROR
  - PURGED

**REASON**
- is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>PUBLIC_ID_INVALID</td>
</tr>
</tbody>
</table>

**RZTA gate, RECEIVE_REQUEST function**

Get the next request. This call blocks if there is no request ready, and returns when a request becomes available or if the RequestStream is destroyed while waiting ("terminate"d). This call will be satisfied without undue waiting if a "notify" callback has been invoked.
A request is delivered by means of a buffer. The total length of the buffer data available is placed in the buffer content value, and as many bytes as will fit are copied to the buffer data area. If there are more bytes than will fit in the buffer area then another receive request may be issued to receive more, and more will be copied until all are delivered.

Should a request not be fully delivered on this interface then the underlying transport will not be reversed (as in half-duplex communications) until another receive is issued. Only when all of the request is delivered will the request be deemed to be delivered and the underlying transport be able to flip.

Should other interface calls be issued on this RequestStream (target) then data bytes of this request may be lost. The request data are bound to the task which is issuing the receive request commands, and the task cannot change in the middle of receiving a request.

The "correlation_id" (optionally received and identical on all the calls for one request) identifies the source from which the request comes and is required when sending a reply on this RequestStream (with "send_reply"). It is guaranteed distinct for each distinct source of this RequestStream. The reply to this request (if there is one) should be accompanied by the same "correlation_id" (on "send_reply") otherwise no guarantee can be made that the reply will return to the correct source.

**Input parameters**

REQUEST_BUFFER  
Buffer into which the request is received.

[MINIMUM_DATA_LENGTH]  
The minimum data length that should secure a response if not all the request is received. Multiple transfers may occur if not enough data is available when the request is issued.

**Output parameters**

[CORRELATION_ID]  
The identifier of the requester using this RequestStream. It is used when replying to this request (using SEND_REPLY on this RequestStream) so as to identify the source from which the request was issued. It is valid only while this RequestStream is available to this transaction.

RESPONSE  
is the domain's response to the call. It can have any of these values:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>OK</td>
<td></td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>REQUESTSTREAM_NOT_CURRENT SERVICE_NOT_AVAILABLE TRANSPORT_FAILURE</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_BUFFER</td>
</tr>
</tbody>
</table>

REASON  
Returned when RESPONSE is EXCEPTION or INVALID. Possible values are:

**RZTA gate, SEND_REPLY function**

Send a reply to a source identified by "correlation_id".

Chapter 71. RequestStream domain (RZ)  985
RequestStream domain (RZ)

The "correlation_id" must be one returned by the "receive_request" function for the current RequestStream, or else the exception "correlation_id_unknown" may be returned.

A reply may consist of the empty sequence of bytes in which case an empty reply is sent.

The usual exceptions are returned for transportation failures.

Input parameters

REPLY_BLOCK
A block containing the complete contiguous reply.

CORRELATION_ID
The correlation id received on RECEIVE_REQUEST for the request to which this is the reply.

Output parameters

RESPONSE
is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

REASON
Returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>REQUESTSTREAM_NOT_CURRENT CORRELATION_ID_UNKNOWN SERVICE_NOT_AVAILABLE TRANSPORT_FAILURE</td>
</tr>
</tbody>
</table>

RZTA gate, GET_SERVER_DATA function

Return the server data for the current RequestStream.

If there is a current RequestStream then server data are available. If there is no current RequestStream the response "(exception, requeststream_not_current)" is returned. If the server data are known but empty (zero-length sequence of bytes), then the outputs are set to indicate zero-length content.

The "server_block" is input so that it may be filled on return. If the data area is too small then the response "(exception, server_block_too_small)" is returned +and no data is transferred+. In this case the output "server_data_length" is set to indicate the total number of bytes in the server data.

Server data do not change while the RequestStream remains current. (A RequestStream may be recreated as a result of a "join" if it has previously been destroyed or terminated. At this point the server data may be different from the original values.)

Input parameters

SERVER_BLOCK
Block in which the server data is placed on output.

Output parameters

SERVER_DATA_LENGTH
The number of bytes of the server data, even if not all were returned.
RESPONSE

is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

REASON

Returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>REQUESTSTREAM_NOT_CURRENT SERVER_BLOCK_TOO_SMALL</td>
</tr>
</tbody>
</table>

RZTA gate, GET_PUBLIC_ID function

The public identifier of the RequestStream for the current transaction is returned.

(If the target of the RequestStream is not internal to the plex there may not be a

public identifier, for example in the case of outbound RequestStreams. In this case

the response is "(exception, public_id_unknown)". However, this should never

happen on this interface, since such a RequestStream will never be set in the "RZ"

transaction manager token for a transaction instance.)

The response "(exception, RequestStream_not_current)" is returned if the XM token

is not valid in the local region, or if no XM token for the RequestStream is set.

Output parameters

PUBLIC_ID

Public RequestStream Identifier -- valid for all participating regions in the

logical server -- of the current target RequestStream which must be

attached to this task/transaction.

RESPONSE

is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

REASON

Returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>REQUESTSTREAM_NOT_CURRENT PUBLIC_ID_UNKNOWN</td>
</tr>
</tbody>
</table>

RZTA gate, GET_CURRENT function

The token for the RequestStream for the current transaction is returned. If the

"XM" token is not set, or is set to an invalid value, then the response "(exception,

RequestStream_not_current)" is returned.

Output parameters

RS_TOKEN

Token for current target RequestStream which must be attached to this

task.

RESPONSE

is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

REASON

Returned when RESPONSE is EXCEPTION. Possible values are:
RZTA gate, GET_JOIN_DATA function

This is a utility function used by the join task which can thereby reduce the number of domain calls to RZ when acting as intermediary to another task on remote join.

If there is no current RequestStream the response "(exception, requeststream_not_current)" is returned.

Output parameters

USERID
The userid of the request processor to be joined.

TRANID
The transid of the request processor to be joined.

PUBLIC_ID
The public id of the target RequestStream on the processor to be joined.

REQUEST_DATA_LENGTH
The data length of the request to be passed to the processor to be joined.

RESPONSE
Values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

REASON
Returned when RESPONSE is EXCEPTION. Possible values are:

RZTA gate, TERMINATE function

Terminate the current (target) RequestStream either normally or abnormally. After this call the "XM" token in the transaction instance is cleared and no longer denotes a RequestStream.

If no "rs_token" is specified and there is no current RequestStream ("XM" token) then the response "(exception, RequestStream_not_current)" is returned.

If "rs_token" is specified it is taken as the token of the RequestStream to terminate.

If the RequestStream token is not in the +RegionRSTable+ (or else is not a +target+ RequestStream in that table) then the response "(exception, rs_token_unknown)" is returned.

If the termination is "normal" then transports are tested for being in a state that accepts normal termination and if they are they are closed and storage associated with the RequestStream is returned and the local region token is invalidated. The +ResionRSTable+ has the "rs_token" removed, and the current RequestStream is unset (the "XM" token is cleared but only if the "rs_token" was +not+ specified on the call). If any transports are +not+ in a correct state then no action is taken and the response "(exception, cannot_terminate_normally)" is returned.
If the termination is "abnormal" then the transports are closed in whatever state they are found, and the RequestStream is terminated as for normal termination.

The usual exceptions concerning the transport mechanism (if used) are possible however +the RequestStream was still terminated even if a transport exception is returned+.

**Output parameters**

**RESPONSE**

* is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**REASON**

Returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>REQUESTSTREAM_NOT_CURRENT RS_TOKEN_UNKNOWN SERVICE_NOT_AVAILABLE TRANSPORT_FAILURE CANNOT_TERMINATE_NORMALLY</td>
</tr>
</tbody>
</table>

**RZRT gate, SET_EXIT_PROGRAM function**

The following defines the syntax of the SET_EXIT_PROGRAM function.

This is used to identify the name of the distributed routing URM at domain initialization time and when the program name is dynamically changed. During CICS initialization the local sysid is also passed to “RZ” from Parameter Manager via this interface.

**Input parameters**

**PROGRAM_NAME**

The name of the program which is the URM for the Distributed Dynamic Routing program.

**[LOCAL_SYSID]**

The SYSID for the local CICS region to recognize it in routing URM responses.

**Output parameters**

**RESPONSE**

* is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**REASON**

Returned when RESPONSE is DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
</tbody>
</table>

**RZRJ gate, PERFORM_JOIN function**

This function reduces the calls necessary from the join task (in remote join capability) to the RZ domain. It initiates the procedures necessary to pass an attached RequestStream to a local processor.
RequestStream domain (RZ)

Output parameters

RESPONSE is the domain’s response to the call. It can have any of these values:
OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

REASON Returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>TRANSPORT_FAILURE JOIN_NOT_POSSIBLE</td>
</tr>
</tbody>
</table>

RequestStream domain’s generic gates

Table 87 summarizes the generic gates which the RequestStream domain implements. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 87. RequestStream domain’s generic gate calls

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>RZDM</td>
<td>RZ 0101</td>
<td>INITIALISE_DOMAIN</td>
<td>DMDM</td>
</tr>
<tr>
<td></td>
<td>RZ 0102</td>
<td>QUIESCE_DOMAIN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TERMINATE_DOMAIN</td>
<td></td>
</tr>
</tbody>
</table>

For descriptions of these functions and their input and output parameters, refer to the §s dealing with the corresponding generic format:

Functions and parameters

Format DMDM—“Domain manager domain’s generic formats” on page 377

RequestStream domain formats

Table 88 shows the formats owned by the requeststream domain, and shows the functions performed. The modules that issue requests in this format make them to RZ.

Table 88. Generic formats owned by the RequestStream domain.

<table>
<thead>
<tr>
<th>Format</th>
<th>Module called</th>
<th>Calling modules</th>
<th>Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSTN</td>
<td>DFHRZLN</td>
<td>DFHIIDM</td>
<td>REGISTER</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DFHIIRP</td>
<td>LISTEN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DFHIIRR</td>
<td>CANCEL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DFHRZLN</td>
<td>DEREGISTER</td>
</tr>
<tr>
<td>NOTI</td>
<td>DFHIIRP</td>
<td>DFHIIRR</td>
<td>NOTIFY</td>
</tr>
</tbody>
</table>

LSTN gate, REGISTER function

Register the notify interface with the server domain. The callback_gate is the gate number +in the caller’s domain+. The caller’s domain is inferred by kernel linkage.

Returns the “notify_token” which identifies this registration.
RequestStream domain (RZ)

Returns ("exception, registration_rejected") if the domain is not ready to allow registrations (initializing or quiescing, for example), or if it cannot issue any more tokens (due to resource limitations, for example). This response is given if the other functions of the domain are not compromised by this failure. Higher severity responses may be given otherwise.

The caller must supply the parameters "call_domain()" and "call_gate()" on this function. See design file for details.

**Input parameters**

**CALLBACK_GATE**

The gate number of the Domain Gate to be called for notification.

**Output parameters**

**NOTIFY_TOKEN**

The token identifying this notification registration. Used on Listen, Cancel, and Notify functions.

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK| EXCEPTION| DISASTER| INVALID| KERNERROR| PURGED

**REASON**

Returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>REGISTRATION_REJECTED</td>
</tr>
</tbody>
</table>

**LSTN gate, LISTEN function**

Listen to the server object and notify the registered notify callback gate, identified by "notify_token", if the object requires service. Pass the supplied "client_token" on the notify call.

Exceptions:

"notify_token_unknown"

the notify token cannot be found by this server domain;

"notify_token_in_use"

the notify token is being used and this server does not allow multiple uses;

"notify_token_misused"

the notify token belongs to another domain and this server does not allow multiple client access to it;

"server_token_unknown"

the server does not recognize the server token;

"server_token_in_use"

the object denoted by the server token already has a listen outstanding, and this server does not allow multiple listens.

The caller must supply the parameters "call_domain()" and "call_gate()" on this function. See design file for details.
RequestStream domain (RZ)

**Input parameters**

NOTIFY_TOKEN
Token that identifies the registration for this listen request.

SERVER_TOKEN
Token that identifies the server-owned object being listened to.

CLIENT_TOKEN
Token for the client, returned on Notify to identify this listen request.

**Output parameters**

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

REASON
Returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NOTIFY_TOKEN_UNKNOWN</td>
</tr>
<tr>
<td></td>
<td>NOTIFY_TOKEN_IN_USE</td>
</tr>
<tr>
<td></td>
<td>NOTIFY_TOKEN_MISUSED</td>
</tr>
<tr>
<td></td>
<td>SERVER_TOKEN_UNKNOWN</td>
</tr>
<tr>
<td></td>
<td>SERVER_TOKEN_IN_USE</td>
</tr>
</tbody>
</table>

**LSTN gate, CANCEL function**

Cancel an outstanding listen request. Ensure that it is not driven upon return from this call. The "notify_token" and the "server_token" are to be specified, and optionally the "client_token" that was specified on the "listen" request can be retrieved.

Exceptions:

"notify_token_unknown"
the notify token cannot be found by this server domain;

"notify_token_misused"
the notify token belongs to another domain and this server does not allow multiple client access to it;

"server_token_unknown"
the server does not recognize the server token;

"listen_not_outstanding"
the object denoted by the server token does not have a listen outstanding.

The caller must supply the parameters "call_domain()" and "call_gate()" on this function. See design file for details.

**Input parameters**

NOTIFY_TOKEN
Token that identifies the registration for this cancel request.

SERVER_TOKEN
Token that identifies the server-owned object being listened to.

**Output parameters**

[CLIENT_TOKEN]
Client Token that was passed on LISTEN and would have been passed to NOTIFY.
**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**REASON**

Returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NOTIFY_TOKEN_UNKNOWN, NOTIFY_TOKEN_MISUSED, SERVER_TOKEN_UNKNOWN, LISTEN_NOT_OUTSTANDING</td>
</tr>
</tbody>
</table>

**LSTN gate, DEREREGISTER function**

Call outstanding notifies on this registration (identified by "notify_token") with a normal termination notification, and then remove the registration of the client (caller) domain from this server domain. The "notify_token" is no longer valid.

Exceptions:

- **"notify_token_unknown"**
  - the notify token cannot be found by this server domain;

- **"notify_token_misused"**
  - the notify token belongs to another domain and this server does not allow multiple client domain access to it;

The caller must supply the parameters "call_domain()" and "call_gate()" on this function. See design file for details.

**Input parameters**

**NOTIFY_TOKEN**

Token that identifies the registration token being deregistered.

**Output parameters**

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**REASON**

Returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NOTIFY_TOKEN_UNKNOWN, NOTIFY_TOKEN_MISUSED</td>
</tr>
</tbody>
</table>

**NOTI gate, NOTIFY function**

Call the client domain notify gate registered as "notify_token" passing the "server_token" and the "client_token" given on the "listen" call, and with a status indicating the notification reason. The listen is considered discharged after this call, and this notify will not be called again unless another "listen" request is made.

If the callback returns an exception response then the registration should be deleted. This is equivalent to a "deregister" call, including the call of this same notify callback gate for any other outstanding "listen" requests.

If the callback returns a response more serious than exception ("disaster", "purged", etc.) then the registration should be deleted but no other callbacks are to be made.
RequestStream domain (RZ)

to outstanding "listen"s for this registration. In particular a kernel error, or an "invalid" response, should not attempt to recall the same gate again. This is deemed to be a severe internal error.

The caller (the server domain) will use "call_gate()" with the value of the callback gate supplied on registration, and "call_domain()" with the client domain who registered. This must refer to a valid gate or a kernel error will result.

Input parameters

NOTIFY_TOKEN
Token that identifies the registration for this notify request.

SERVER_TOKEN
Token that identifies the server-owned object being notified.

CLIENT_TOKEN
Token supplied by the client on the listen request.

NOTIFY_STATUS
Values: NOTIFY, CLOSE, ABEND, TIMEOUT

Output parameters

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

REASON
Returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NOTIFY_CALLBACK_FAILED</td>
</tr>
</tbody>
</table>

Modules

The following modules implement the RZ domain:

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHHRZDUF</td>
<td>Dump Formatting program</td>
</tr>
<tr>
<td>DFHHRZIX</td>
<td>XM Attach Client for InStore transports</td>
</tr>
<tr>
<td>DFHHRZN</td>
<td>Join task program</td>
</tr>
<tr>
<td>DFHHRZLN</td>
<td>Listen and Notify calls</td>
</tr>
<tr>
<td>DFHHRZNR2</td>
<td>Init rsnr class (notification object)</td>
</tr>
<tr>
<td>DFHHRZOFI</td>
<td>Init object factory class</td>
</tr>
<tr>
<td>DFHHRZRG2</td>
<td>Init rsrg registration class</td>
</tr>
<tr>
<td>DFHHRZJR</td>
<td>Perform join</td>
</tr>
<tr>
<td>DFHHRZRM</td>
<td>RM Resource Owner for RZ</td>
</tr>
<tr>
<td>DFHHRZRS1</td>
<td>Init rz_reqstream class</td>
</tr>
<tr>
<td>DFHHRZRT</td>
<td>Set Routing Exit program name</td>
</tr>
<tr>
<td>DFHHRZRT1</td>
<td>Init routing URM class (rzrt)</td>
</tr>
<tr>
<td>DFHHRZRT2</td>
<td>Invoke Routing URM</td>
</tr>
<tr>
<td>DFHHRZSO</td>
<td>Source commands on RequestStreams (not Create/Join)</td>
</tr>
<tr>
<td>DFHHRZSO1</td>
<td>Create and Join commands on Source RequestStreams</td>
</tr>
<tr>
<td>DFHHRZTA</td>
<td>Target commands on RequestStreams</td>
</tr>
</tbody>
</table>
### Exits

There are no exits specifically associated with the RZ domain, except for the Distributed Dynamic Routing Exit (URM) for which see...??... called from DFHRZRT2.

### Trace

The point IDs for the RequestStream domain are of the form RZ xxxx; the corresponding trace levels are RZ 1, RZ 2 and Exc.

For more information about the trace points, see the [CICS Trace Entries](#). For more information about using traces in problem determination, see the [CICS Problem Determination Guide](#).
Chapter 72. Resource definition online (RDO)

The CEDA transaction creates and alters the definitions of system resources in the CICS system definition (CSD) data set.

RDO provides:

- Online transactions that can be used to inspect, change, and install resource definitions:
  - CEDA (inspect, change, and install)
  - CEDB (inspect and change)
  - CEDC (inspect only).
- A programmable interface to the CEDA transaction, using an EXEC CICS LINK command in the application program to invoke DFHEDAP directly. (For further information, see the CICS Customization Guide.)
- A set of system programmer API command (the EXEC CICS CREATE commands) for creating CICS resources dynamically.
- An offline utility, DFHCSDUP, to inspect or change resource definitions. (For a description of this utility, see "Chapter 19. CSD utility program (DFHCSDUP)" on page 277.)

Design overview

Resource definitions are maintained on the CICS system definition (CSD) data set. The resource definitions in the CSD data set can be viewed and changed using either the online CEDx transactions, or the offline utility DFHCSDUP.

Installation of resource definitions makes the definitions available to the running CICS system. Resource definitions can be installed at these times:

- When CICS is cold started, using the GRPLIST system initialization parameter.
- During a run of CICS, using the CEDA transaction.

When resource definitions are installed, they are made available through the appropriate resource managers.

Modules

The relationships between the components of RDO, and the components of some of the services it uses, are shown in Figure 84 on page 998.
**DFHEDAP** and **DFHEDAD** control the CEDA, CEDB, and CEDC transactions. They provide screen management for the transactions, and invoke **DFHAMP** to implement any actions that are required.

**DFHSII1** invokes **DFHAMP** when CICS is cold started, to install resource definitions for the current run. These resource definitions are specified by the GRPLIST system initialization parameter. **DFHSII1** passes the GRPLIST system initialization parameter to **DFHAMP**.

**DFHAMP**, the allocation management program, manages all requests to view, change, and install resources. It uses the services provided by other parts of RDO, and by the resource managers:

- **DFHAMP** invokes **DFHPUP** and **DFHDMP** to read, write, and update resource definitions on the CSD data set:
  - **DFHPUP**, the parameter utility program, converts resource definition data between the parameter list format provided by **DFHAMP** and the record format needed by the CSD.
  - **DFHDMP**, the CSD management program, manages I/O of resource definition data to and from the CSD data set.
- **DFHAMP** invokes **DFHTOR**, the terminal object resolution program, to merge TERMINAL, TYPETERM, CONNECTION, and SESSION definitions.
When requests are made to install TERMINALs, TYPETERMs, CONNECTIONs, and SESSIONs, DFHTOR merges TYPETERM and TERMINAL information, and also CONNECTION and SESSION information, and passes this merged information back to DFHAMP.

DFHAMP passes the merged definitions to DFHZCQ to install in the running CICS system. Any merged TERMINAL definitions that are to be used as autoinstall terminal models are passed to the autoinstall terminal model (AITM) manager.

When TYPETERM definitions are installed, DFHTOR records the information about the CICS global catalog for subsequent use.

When the CHECK command is issued, DFHTOR checks the appropriate TERMINAL, TYPETERM, CONNECTION, and SESSION definitions for consistency.

• DFHAMP calls the appropriate resource managers to install resources in the running CICS system:
  - DFHZCQ is invoked to install CONNECTION, SESSION, and TERMINAL definitions.
  - DFHAMXM is invoked to install TRANSACTION and PROFILE definitions.
  - DFHPGDD is invoked to install PROGRAM, MAPSET, and PARTITIONSET definitions.
  - These subroutine “gates” are called to install resources related to file control: FCMT, for FCT entries
    FCRL, for LSR pools
    FCDN, for DSN blocks
    FCFS, to open and close files
    AFMT, for AFCT entries for files.
  - The AITM manager is invoked, using an AITM ADD_REPL_TERM_MODEL subroutine call (see “Chapter 9. Autoinstall terminal model manager” on page 127), to install autoinstall terminal models.
  - The partner resource manager is invoked, using a PRPT ADD_REPLACE_PARTNER subroutine call (see “Chapter 63. Partner resource manager” on page 869), to install partner resources for the SAA communications interface.

DFHEICRE processes all the EXEC CICS CREATE commands. It builds an internal DEFINE command for the resource to be created, and passes it to DFHCAP for interpretation. The encoded command is then passed directly to DFHAMP to install the resource in the running system. The CSD file is not accessed at all during this processing.

DFHCSDUP, the offline CICS system definition utility program, uses batch versions of routines from DFHPUP and DFHDMP to read, write, and update resource definitions on the CSD data set (see “Chapter 19. CSD utility program (DFHCSDUP)” on page 277).

For a detailed description of how the CEDA transaction handles terminal resources, see “Chapter 93. Terminal control” on page 1193.

Exits

The XRSINDI global user exit is invoked at each install or EXEC CICS CREATE.
Trace

The following point IDs are provided, with a trace level of AP 1:
- AP 00EB (DFHAMP)
- AP 00EC (DFHDMP)
- AP 00EF (DFHTOR)
- AP 00E2 (DFHPUP).

For more information about the trace points, see the CICS Trace Entries. For more information about using traces in problem determination, see the CICS Problem Determination Guide.
Chapter 73. SAA Communications and Resource Recovery interfaces

This section describes the CICS implementation of the Communications and Resource Recovery elements of the Systems Application Architecture Common Programming Interface (also known as the SAA Communications and SAA Resource Recovery interfaces respectively).

The SAA Communications and Resource Recovery interfaces are both call-based application programming interfaces that are common across all programming languages and across hardware systems.

The common programming interface (CPI) component of CICS, also sometimes known as the CP component, provides application programming interfaces that conform to SAA specifications for Communications and Resource Recovery interfaces.

Note: This CICS component does not currently handle any other SAA interface elements.

The CPI component is part of the AP domain, and is shipped as object code only (OCO).

The SAA Communications interface allows CICS applications to communicate via APPC (LU6.2) links to partner applications on any system that conforms to SAA standards. This interface consists of a set of defined verbs as program calls that are adapted for the language being used. For further information about the general call-based API, see the SAA CPI Communications Reference manual, SC26-4399.

The SAA Resource Recovery interface provides an SAA application programming interface for commit and backout of recoverable resources. This interface consists of two defined verbs as program calls that are adapted for the language being used:

**SRRCMIT**
Commit

**SRRBACK**
Backout

For further information, see the SAA CPI Resource Recovery Reference manual, SC31-6821.

The SAA Resource Recovery interface in CICS provides an alternative to the use of EXEC CICS SYNCPOINT and EXEC CICS SYNCPOINT ROLLBACK commands.
SAA Communications and Resource Recovery interfaces

The SRRCMIT call is equivalent to the EXEC CICS SYNCPOINT command, and the SRRBACK call is equivalent to the EXEC CICS SYNCPOINT ROLLBACK command. A single application can use SAA Resource Recovery calls, EXEC CICS commands, or a mixture of both.

Design overview

This section describes the SAA Communications and Resource Recovery interfaces.

The SAA Communications interface

When an application issues an SAA Communications call, control passes via the DFHCPLC application link-edit stub to the common programming interface program (DFHCPI), which in turn passes the request to the DFHCPIC program load module. DFHCPIC verifies the parameters, checks the conversation state, and (if required) issues a DFHLUC macro call to invoke the LU6.2 application request logic module (DFHZARL). For details of DFHZARL, see “Chapter 25. Distributed transaction processing” on page 331.

Figure 85 shows how the SAA Communications interface support relates to CICS intersystem communication (ISC) using VTAM LU6.2. The numbers in Figure 85 refer to the notes that follow it. CMxxxx represents a program call defined in the SAA Communications interface.

Figure 85. SAA Communications application request processing

Notes:

1. Distributed transaction processing (DTP) allows a transaction using EXEC CICS commands to communicate with a transaction running in another system. This is carried out by DFHEIP and related EXEC interface processor modules. For a VTAM LU6.2 intersystem link, each request is converted into DFHLUC macro requests that call DFHZARL.
SAA Communications and Resource Recovery interfaces

2. The SAA Communications interface is implemented by the DFHCPIC load module within the CP (or CPI) component. DFHCPIC maps the CMxxxx application requests into DFHLUC macro calls.

3. To begin a conversation, the SAA Communications interface requires specific information (side information) about the partner program, including its name and system details. This is implemented within CICS as an RDO object called the PARTNER, which is encapsulated by the partner resource manager (PR) component. Further details of the PR component are given under "Chapter 63 Partner resource manager" on page 869.

Using the SAA Communications interface on recoverable conversations

When using the SAA Communications interface on recoverable conversations (that is, conversations with the syncllevel set to CM_SYNC_POINT), DFHLUC syncpoint requests are routed to DFHZARL via the SAA Communications interface syncpoint request handler (DFHCPSRH) in the DFHCPIC load module. This allows the conversation state to be tracked.

For the equivalent EXEC CICS syncllevel 2 conversations, DFHLUC syncpoint requests pass directly to DFHZARL.

The SAA Resource Recovery Interface

When an application issues an SAA Resource Recovery call, control passes via the DFHCPLRR application link-edit stub to the common programming interface program (DFHCPI), which in turn passes the request to the DFHCPIRR program load module. DFHCPIRR verifies the parameters, and (if required) issues an appropriate DFHSP macro call: DFHSP TYPE=USER for SRRCMIT, or DFHSP TYPE=ROLLBACK for SRRBACK.

Figure 86 on page 1004 shows how the SAA Resource Recovery interface support relates to the processing of EXEC CICS SYNCPOINT commands. The number in the figure refers to the accompanying note. SRRxxxx represents a program call defined in the SAA Resource Recovery interface, namely, SRRBACK or SRRCMIT.
SAA Communications and Resource Recovery interfaces

![Diagram showing SAA Resource Recovery interfaces]

**Note:** The SAA Resource Recovery interface is implemented by the DFHCPIRR load module within the CP (or CPI) component. DFHCPIRR maps SRRxxxx application requests into DFHSP macro calls.

*Figure 86. SAA Resource Recovery application request processing*

---

**SAA Communications and Resource Recovery interfaces**

---

**Functions provided by the CPI component**

Table 89 summarizes the external subroutine interfaces provided by the CPI component. It shows the subroutine call formats, the level-1 trace point IDs of the modules providing the functions for these formats, and the functions provided.

<table>
<thead>
<tr>
<th>Format</th>
<th>Trace</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPIN</td>
<td>AP 0C01</td>
<td>START_INIT</td>
</tr>
<tr>
<td></td>
<td>AP 0C02</td>
<td>COMPLETE_INIT</td>
</tr>
<tr>
<td>CPSP</td>
<td>AP 0CD0</td>
<td>SYNCPOINT_REQUEST</td>
</tr>
<tr>
<td></td>
<td>AP 0CD1</td>
<td></td>
</tr>
</tbody>
</table>

**CPIN format, START_INIT function**

The START_INIT function of the CPIN format is used to attach a CICS task to perform initialization of the CPI component.

**Input parameters**

None.

**Output parameters**

**RESPONSE**

is the subroutine’s response to the call. It can have any of these values:
OK|DISASTER|KERNERROR

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>GETMAIN_FAILED, ADD_SUSPEND_FAILED</td>
</tr>
</tbody>
</table>

**CPIN format, COMPLETE_INIT function**

The COMPLETE_INIT function of the CPIN format is used to wait for the initialization task attached by the START_INIT function to complete processing.

**Input parameters**

None.

**Output parameters**

**RESPONSE**

is the subroutine’s response to the call. It can have any of these values:

OK|DISASTER|KERNERROR

[REASON]
is returned when RESPONSE is DISASTER. It has this value:

INIT_TASK_FAILED

**CPSP format, SYNCPOINT_REQUEST function**

The SYNCPOINT_REQUEST function of the CPSP format is used to send LU6.2 syncpoint flows on recoverable conversations using the SAA Communications interface, and to update the conversation state as required.

**Input parameters**

**CPC_ADDRESS**

is the address of the SAA Communications conversation control block (CPC).

**LUC_ADDRESS**

is the address of the DFHLUC parameter list.

**Output parameters**

**RESPONSE**

is the subroutine’s response to the call. It can have either of these values:

OK|KERNERROR

**Modules**

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHAPTRF</td>
<td>Trace interpreter for the SAA Communications and Resource Recovery interfaces</td>
</tr>
<tr>
<td>DFHCPARH</td>
<td>SAA Communications application request handler (entry processor for all application calls to the DFHCPIC load module, routing them to the appropriate DFHPCCx module)</td>
</tr>
<tr>
<td>DFHPCCx</td>
<td>Components of the DFHCPIC load module, each object module typically handling a different CMxxxx application request</td>
</tr>
<tr>
<td>DFHCPDUF</td>
<td>Offline system dump formatter for CP keyword</td>
</tr>
</tbody>
</table>
### SAA Communications and Resource Recovery interfaces

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHCPI</td>
<td>Common programming interface program (link-edited with DFHEIP and DFHAICBP to form the DFHAIP load module)</td>
</tr>
<tr>
<td>DFHCPIN1</td>
<td>Initialization management program for the SAA Communications and Resource Recovery interfaces</td>
</tr>
<tr>
<td>DFHCPIN2</td>
<td>Runs as a CICS task to perform initialization for the SAA Communications and Resource Recovery interfaces</td>
</tr>
<tr>
<td>DFHCPIR</td>
<td>SAA Resource Recovery entry processor, handling all calls to the DFHCPIRR load module</td>
</tr>
<tr>
<td>DFHCPLC</td>
<td>Link-edit stub for applications using the SAA Communications interface</td>
</tr>
<tr>
<td>DFHCPLRR</td>
<td>Link-edit stub for applications using the SAA Resource Recovery interface</td>
</tr>
<tr>
<td>DFHCPSRH</td>
<td>SAA Communications syncpoint request handler (part of the DFHCPI load module)</td>
</tr>
</tbody>
</table>

### Exits

No global user exit points are provided for this component.

### Trace

The following point ID is provided for this component:

- AP 0Cxx, for which the trace levels are CP 1, CP 2, and Exc.

For more information about the trace points, see the [CICS Trace Entries](#). For more information about using traces in problem determination, see the [CICS Problem Determination Guide](#).
Chapter 74. Scheduler Services domain(SH)

The scheduler services domain (also sometimes known simply as “scheduler services”) is used to harden schedule requests between UOWs and to route schedule requests to a target region identified by the distributed routing exit program. A schedule request may be viewed as a request to undertake a piece of work, execute a named transaction. The domain is part of CICS business transaction services.

Scheduler services domain’s specific gate

Table 90 summarizes the scheduler services domain’s specific gate. It shows the level-1 trace point IDs of the modules providing the functions for the gate, the functions provided by the gate, and whether or not the functions are available through the exit programming interface (XPI).

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>XPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHPR</td>
<td>SH 0151</td>
<td>ADD_PENDING_REQUEST</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>SH 0152</td>
<td>DELETE_PENDING_REQUEST</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SET_BOUND_REQUEST</td>
<td>NO</td>
</tr>
<tr>
<td>SHRT</td>
<td>SH 0141</td>
<td>SET_EXIT_PROGRAM</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>SH 0142</td>
<td>INQUIRE_EXIT_PROGRAM</td>
<td>NO</td>
</tr>
<tr>
<td>SHRQ</td>
<td>SH 0111</td>
<td>PERFORM_RESTART_DREDGE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>SH 0112</td>
<td>PERFORM_REGULAR_DREDGE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PERFORM_SHUTDOWN</td>
<td>NO</td>
</tr>
<tr>
<td>SHRR</td>
<td>SH 0161</td>
<td>ROUTE_REQUEST</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>SH 0162</td>
<td>RECEIVE_REQUEST</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RETRY_REQUEST</td>
<td>NO</td>
</tr>
</tbody>
</table>

SHPR gate, ADD_PENDING_REQUEST function

The ADD_PENDING_REQUEST function of the SHPR gate is used to add a pending schedule request to the scheduler services queue associated with this UOW. The pending schedule requests are hardened to the scheduler services local request queue (LRQ) as part of syncpoint processing.

Input parameters

TRANID

is an 4-character transaction id.

USERID

is an 8-character userid.

TIME

is a string of length 8, used when a request is delayed for a period time.

TOKEN

is a string of length 4, used to identify the pending queue.

BALANCE

indicates whether this schedule request is eligible for workload balancing. It can have either of these values:

YES | NO
PTYPE is the 8-character process type.

PNAME is the 36-character process name.

ACTIVITY_ID is a block containing the activity id.

ACTIVITY_REQUEST_BLOCK is a block containing the BAM domain activity request block.

Output parameters
RESPONSE is the domain’s response to the call. It can have any of these values: OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

**SHPR gate, DELETE_PENDING_REQUEST function**
The DELETE_PENDING_REQUEST of the SHPR gate is used to delete a pending request queue.

**Input parameters**
TOKEN is a string of length 4, which identifies the queue to be deleted.

**Output parameters**
RESPONSE is the domain’s response to the call. It can have any of these values: OK|EXCEPTION|PURGED|INVALID|DISASTER|KERNERROR

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>REQUEST_NOT_FOUND</td>
</tr>
</tbody>
</table>

**SHPR gate, SET_BOUND_REQUEST function**
The SET_BOUND_REQUEST function of the SHPR gate is used to update the schedule request to indicate that a process and/or activity has completed.

**Input parameters**

ACTIVITY_COMPLETE indicates whether the activity associated with this UOW has completed. It can have either of these values: YES|NO

PROCESS_COMPLETE indicates whether the process associated with this UOW has completed. It can have either of these values: YES|NO

**Output parameters**
RESPONSE is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|INVALID|DISASTER|KERNERROR

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>REQUEST_NOT_FOUND</td>
</tr>
</tbody>
</table>

**SHRT gate, SET_EXIT_PROGRAM function**

The SET_EXIT_PROGRAM function of the SHRT gate is used to alter the distributed routing exit program, initially named on the DSRTPGM system initialisation parameter. The sysid of the local system is passed during CICS initialisation.

**Input parameters**

PROGRAM_NAME
is the 8-character exit program name.

LOCAL_SYSID
is the 4-character local sysid.

**Output parameters**

RESPONSE
is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
</tbody>
</table>

**SHRT gate, INQUIRE_EXIT_PROGRAM function**

The INQUIRE_EXIT_PROGRAM function of the SHRT gate is used to return the name of the distributed routing exit program, initially named on the DSRTPGM system initialisation parameter.

**Input parameters**

PROGRAM_NAME
is the 8-character exit program name.

**Output parameters**

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND</td>
</tr>
</tbody>
</table>
Scheduler Services domain (SH)

**SHRQ gate, PERFORM_RESTART_DREDGE function**

The PERFORM_RESTART_DREDGE of the SHRQ gate is used to initiate the dredging of expired schedule requests on the local request queue (LRQ) after a CICS system restart.

**Output parameters**

RESPONSE is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | INVALID | DISASTER | KERNERROR | PURGED

**SHRQ gate, PERFORM_REGULAR_DREDGE function**

The PERFORM_REGULAR_DREDGE function of the SHRQ gate initiates the periodic dredging of expired schedule requests on the local request queue (LRQ).

**Output parameters**

RESPONSE is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | INVALID | DISASTER | KERNERROR | PURGED

**SHRQ gate, PERFORM_SHUTDOWN function**

The PERFORM_SHUTDOWN function of the SHRQ gate is used to stop dredging of schedule requests on the local request queue (LRQ), preventing any further CICS BTS work from being initiated.

**Output parameters**

RESPONSE is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | INVALID | DISASTER | KERNERROR | PURGED

**SHRR gate, ROUTE_REQUEST function**

The ROUTE_REQUEST function of the SHRR gate is used to identify a target region to which a schedule request should be routed.

**Input parameters**

REQUEST_BUFFER is a buffer used to hold the schedule request which is to be routed.

**Output parameters**

SYSID is the 4-character sysid of the region to which the schedule request should be routed.

RESPONSE is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | INVALID | DISASTER | KERNERROR | PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NO_REQUEST_FOUND, REQUEST_BUFFER_TOO_SMALL, NO_SYSTEM</td>
</tr>
</tbody>
</table>
SHRR gate, RECEIVE_REQUEST function

The RECEIVE_REQUEST function of the SHRR gate is used to receive a schedule request once it has been routed to the target region.

**Input parameters**

REQUEST_BUFFER

is a buffer used to hold the received schedule request.

**Output parameters**

RESPONSE

is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON]

is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>INVALID_REQUEST_RECEIVED</td>
</tr>
</tbody>
</table>

SHRR gate, RETRY_REQUEST function

The RETRY_REQUEST function of the SHRR gate is used to obtain another target region if the initial attempt at routing the schedule request fails.

**Input parameters**

REQUEST_BUFFER

is a buffer used to hold the schedule request which is to be routed.

ROUTE_ERROR

indicates the reason why the routing of the schedule request failed. It can have a value of:

SYSID_NOT_FOUND|SYSID_OUT_OF_SERVICE|NO_SESSIONS|ALLOCATE_REJECTED|QUEUE_PURGED|FUNC_NOT_SUPPORTED|LEGERR|PGMIDERR|INVREQ|NOTAUTH|TERMERR

**Output parameters**

SYSID

is the 4-character sysid of the region to which the schedule request should be routed.

LOCAL

indicates whether we should retry the schedule request on the local region. It can take the values:

YES|NO

ABEND_CODE

is the 4-character abend code.

RESPONSE

is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON]

is returned when RESPONSE is EXCEPTION. Possible values are:
Scheduler Services domain (SH)

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NO_REQUEST_FOUND, REQUEST_BUFFER_TOO_SMALL, NO_SYSTEM</td>
</tr>
</tbody>
</table>

Scheduler service domain’s generic gates

Table 91 summarizes the scheduler services domain’s generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 91. Scheduler services domain’s generic gates

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMDM</td>
<td>SH 0101</td>
<td>PRE_INITIALIZE</td>
<td>DMDM</td>
</tr>
<tr>
<td></td>
<td>SH 0102</td>
<td>INITIALIZE_DOMAIN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>QUIESCE_DOMAIN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TERMINATE_DOMAIN</td>
<td></td>
</tr>
<tr>
<td>XMAC</td>
<td>SH 0121</td>
<td>INIT_XM_CLIENT</td>
<td>XMAC</td>
</tr>
<tr>
<td></td>
<td>SH 0122</td>
<td>BIND_XM_CLIENT</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>RELEASE_XM_CLIENT</td>
<td></td>
</tr>
<tr>
<td>RMDE</td>
<td>SH 0131</td>
<td>START_DELIVERY</td>
<td>RMDE</td>
</tr>
<tr>
<td></td>
<td>SH 0132</td>
<td>DELIVER_RECOVERY</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>END_DELIVERY</td>
<td></td>
</tr>
<tr>
<td>RMKP</td>
<td>SH 0131</td>
<td>TAKE_KEYPOINT</td>
<td>RMKP</td>
</tr>
<tr>
<td></td>
<td>SH 0132</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RMRO</td>
<td>SH 0131</td>
<td>PERFORM_PERPARE</td>
<td>RMDE</td>
</tr>
<tr>
<td></td>
<td>SH 0132</td>
<td>PERFORM_COMMIT</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PERFORM_SHUNT</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PERFORM_UNSHUNT</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>START_BACKOUT</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>DELIVER_BACKOUT</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>END_BACKOUT</td>
<td></td>
</tr>
<tr>
<td>TISR</td>
<td>SH 0701</td>
<td>NOTIFY</td>
<td>TISR</td>
</tr>
<tr>
<td></td>
<td>SH 0702</td>
<td></td>
<td></td>
</tr>
<tr>
<td>KETI</td>
<td>SH 0701</td>
<td>NOTIFY_RESET</td>
<td>KETI</td>
</tr>
<tr>
<td></td>
<td>SH 0702</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For descriptions of these functions and their input and output parameters, refer to the §§ dealing with the corresponding generic formats:
Functions and parameters

Format DMDM—“Domain manager domain’s generic formats” on page 377
Format XMAC—“Chapter 100. Transaction manager domain (XM)” on page 1261
Format RMDE—“Recovery Manager domain’s call back formats” on page 953
Format RMRO—“Recovery Manager domain’s call back formats” on page 953
Format RMKP—“Recovery Manager domain’s call back formats” on page 953
Format TISR—“Timer domain’s specific gate” on page 1225
Format KETI—“Kernel domain’s specific gates” on page 731

When invoked for the DMDM INITIALIZE_DOMAIN function scheduler services obtains its anchor block and initializes its various classes. This would include starting the scheduler services system task, CSHY and obtaining the name of the distributed routing exit program named on the DSRTPGM system initialization parameter.

When invoked by transaction manager via the XMAC generic gate, for INIT_XM_CLIENT SH domain obtains a user token in order to set up the correct transaction environment. For BIND_XM_CLIENT SH domain initializes recoverable resources, which includes setting the RM work token and logging a backout request for this UOW. SH domain also determines the name of the program to be invoked on the initial program link.

When invoked for the RMRO PERFORM_PREPARE function SH domain prepares to commit the pending request for the UOW by adding them to the local request queue (LRQ). On receipt of the RMRO PERFORM_COMMIT the schedule requests for this UOW are committed or destroyed, depending upon whether we are committing forwards or backwards.

When invoked for the RMDE DELIVER_RECOVERY function SH domain recreates the pending request queues and in the case of inflight UOWs attempts to retry the associated BTS activation.

Scheduler services makes use of the TISR functions, REQUEST_NOTIFY_INTERVAL and NOTIFY to deal with delayed schedule requests i.e. EXEC CICS DEFINE TIMER calls.

The KETI interface is used when the time is adjusted, causing the time at which delayed schedule requests are to expire to be recalculated.
## Modules

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHSHDM</td>
<td>Handles the following requests:  &lt;br&gt;PRE_INITIALIZE  &lt;br&gt;INITIALIZE_DOMAIN  &lt;br&gt;QUIESCE_DOMAIN  &lt;br&gt;TERMINATE_DOMAIN</td>
</tr>
<tr>
<td>DFHSHRM</td>
<td>Handles the following requests:  &lt;br&gt;PERFORM_PREPARE  &lt;br&gt;PERFORM_COMMIT  &lt;br&gt;START_BACKOUT  &lt;br&gt;DELIVER_BACKOUT  &lt;br&gt;END_BACKOUT  &lt;br&gt;PERFORM_SHUNT  &lt;br&gt;PERFORM_UNSHUNT  &lt;br&gt;TAKE_KEYPOINT  &lt;br&gt;START_DELIVERY  &lt;br&gt;DELIVER_RECOVERY  &lt;br&gt;END_DELIVERY</td>
</tr>
<tr>
<td>DFHSHXM</td>
<td>Handles the following requests:  &lt;br&gt;INIT_XM_CLIENT  &lt;br&gt;BIND_XM_CLIENT  &lt;br&gt;RELEASE_XM_CLIENT</td>
</tr>
<tr>
<td>DFHSHTI</td>
<td>Handles the following requests:  &lt;br&gt;NOTIFY  &lt;br&gt;NOTIFY_RESET</td>
</tr>
<tr>
<td>DFHSHRQ</td>
<td>Handles the following requests:  &lt;br&gt;PERFORM_RESTART_DREDGE  &lt;br&gt;PERFORM_REGULAR_DREDGE  &lt;br&gt;PERFORM_SHUTDOWN</td>
</tr>
<tr>
<td>DFHSHPR</td>
<td>Handles the following requests:  &lt;br&gt;ADD_PENDING_REQUEST  &lt;br&gt;DELETE_PENDING_REQUEST  &lt;br&gt;SET_BOUND_REQUEST</td>
</tr>
<tr>
<td>DFHSHRT</td>
<td>Handles the following requests:  &lt;br&gt;SET_EXIT_PROGRAM  &lt;br&gt;INQUIRE_EXIT_PROGRAM</td>
</tr>
<tr>
<td>DFHSHRR</td>
<td>Handles the following requests:  &lt;br&gt;ROUTE_REQUEST  &lt;br&gt;RECEIVE_REQUEST  &lt;br&gt;RETRY_REQUEST</td>
</tr>
<tr>
<td>DFHSHSY</td>
<td>Implements the SH domain system task, CSHY.</td>
</tr>
<tr>
<td>DFHSHRRP</td>
<td>The SH domain request receiving program, the back-end to SH domain DPL requests.</td>
</tr>
<tr>
<td>DFHSHRSP</td>
<td>The SH domain request sending program, the front-end to SH domain DPL requests.</td>
</tr>
<tr>
<td>DFHSHDUF</td>
<td>Formats the SH domain control blocks</td>
</tr>
<tr>
<td>DFHSHTRI</td>
<td>Interprets SH domain trace entries</td>
</tr>
<tr>
<td>DFHSHRE1</td>
<td>Initializes the SH domain request class.</td>
</tr>
<tr>
<td>DFHSHOFI</td>
<td>Initializes the SH domain object factory class.</td>
</tr>
<tr>
<td>DFHSHVP1</td>
<td>Initializes the SH domain variable length storage class.</td>
</tr>
</tbody>
</table>
Scheduler Services domain (SH)

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHSHRT1</td>
<td>Initializes the SH domain request routing class.</td>
</tr>
<tr>
<td>DFHSHRT2</td>
<td>Invokes the distributed routing exit program, named on the DSRTPGM system initialization parameter.</td>
</tr>
<tr>
<td>DFHSHRQ1</td>
<td>Initializes the SH domain request queue class.</td>
</tr>
</tbody>
</table>

**Exits**

No global user exit points are provided in this domain.

**Trace**

The point IDs for the scheduler services domain are of the form SH xxxx; the corresponding trace levels are SH 1, SH 2, and Exc.

For more information about the trace points, see the [CICS User's Handbook](#). For more information about using traces in problem determination, see the [CICS Problem Determination Guide](#).
Chapter 75. Security manager domain

The security manager domain provides an optional facility for checking user authority to run transactions and access resources.

Security manager domain’s specific gates

Table 92 summarizes the security manager domain’s specific gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and whether or not the functions are available through the exit programming interface (XPI).

Table 92. Security manager domain’s specific gates

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>XPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>XSAD</td>
<td>XS 0201</td>
<td>ADD_USER_WITH_PASSWORD,</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>XS 0202</td>
<td>ADD_USER_WITHOUT_PASSWORD</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>XS 0203</td>
<td>DELETE_USER_SECURITY</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>XS 0204</td>
<td>INQUIRE_USER_ATTRIBUTES</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>XS 0205</td>
<td>VALIDATE_USERID</td>
<td>NO</td>
</tr>
<tr>
<td>XSFL</td>
<td>XS 0501</td>
<td>FLATTEN_USER_SECURITY</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>XS 0502</td>
<td>UNFLATTEN_USER_SECURITY</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>XS 0503</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>XS 0504</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>XS 0505</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>XS 0509</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>XS 050A</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>XS 050B</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>XS 050C</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>XS 050D</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XSIS</td>
<td>XS 0301</td>
<td>INQUIRE_REGION_USERID</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>XS 0302</td>
<td>INQ_SECURITY_DOMAIN_PARMS</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>XS 0303</td>
<td>SET_SECURITY_DOMAIN_PARMS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>XS 0304</td>
<td>SET_NETWORK_IDENTIFIER</td>
<td></td>
</tr>
<tr>
<td></td>
<td>XS 0305</td>
<td>SET_SPECIAL_TOKENS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>XS 0306</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>XS 0307</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>XS 0308</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>XS 0309</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>XS 030A</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>XS 030B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XSLU</td>
<td>XS 0801</td>
<td>GENERATE_APPC_BIND</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>XS 0802</td>
<td>GENERATE_APPC_RESPONSE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>XS 0803</td>
<td>VALIDATE_APPC_RESPONSE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>XS 0804</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>XS 0805</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>XS 0806</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>XS 0807</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>XS 0808</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>XS 0809</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>XS 080A</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Security manager domain

#### Table 92. Security manager domain’s specific gates (continued)

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>XPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>XSPW</td>
<td>XS 0601</td>
<td>CREATE_PASSTICKET</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>XS 0602</td>
<td>INQUIRE_PASSWORD_DATA</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>XS 0603</td>
<td>UPDATE_PASSWORD</td>
<td></td>
</tr>
<tr>
<td></td>
<td>XS 0604</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>XS 0605</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>XS 0606</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XSRC</td>
<td>XS 0701</td>
<td>CHECK_CICS_RESOURCE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>XS 0702</td>
<td>CHECK_CICS_COMMAND</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>XS 0703</td>
<td>CHECK_NON_CICS_RESOURCE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>XS 0704</td>
<td>CHECK_SURROGATE_USER</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>XS 0705</td>
<td>REBUILD_RESOURCE_CLASSES</td>
<td></td>
</tr>
<tr>
<td></td>
<td>XS 0706</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>XS 0707</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>XS 0708</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>XS 0709</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>XS 070A</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>XS 070B</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>XS 070C</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>XS 070D</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>XS 070E</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XSXM</td>
<td>XS 0401</td>
<td>ADD_TRANSACTION_SECURITY</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>XS 0402</td>
<td>DEL_TRANSACTION_SECURITY</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>XS 0403</td>
<td>END_TRANSACTION</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>XS 0404</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>XS 0405</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>XS 0409</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### XSAD gate, ADD_USER_WITH_PASSWORD function

The ADD_USER_WITH_PASSWORD function of the XSAD gate is used to add a user to the security domain and verify the associated password or oidcard.

**Input parameters**

**USERID**

is the identifier of the user (a userid of 1 through 10 alphanumeric characters) to be added to the security domain.

**USERID_LENGTH**

is the length of the USERID value.

**[PASSWORD]**

is the current password, 1 through 10 alphanumeric characters, for the userid specified by the USERID value.

**[PASSWORD_LENGTH]**

is the 8-bit length of the PASSWORD value. This parameter is only valid if PASSWORD is also specified.

**[NEW_PASSWORD]**

is a new password, 1 through 10 alphanumeric characters, to be assigned to the userid (specified by the USERID value). This parameter is only valid if PASSWORD is also specified.

**[NEW_PASSWORD_LENGTH]**

is the 8-bit length of the NEW_PASSWORD value. This parameter is only valid if NEW_PASSWORD is also specified.
Security manager domain

[OIDCARD] is an optional oidcard (operator identification card); a 65-byte field containing further security data from a magnetic strip reader (MSR) on 32xx devices.

[GROUPlD] is an optional identifier, 1 through 10 alphanumeric characters, of a RACF user group to which the userid (specified by the USERID value) is to be assigned.

[GROUPlD_LENGTH] is the 8-bit length of the GROUPID value. This parameter is only valid if GROUPID is also specified.

[ENTRY_PORT_NAME] is an optional name of an entry port, 1 through 8 alphanumeric characters, to be assigned to the userid (specified by the USERID value).

[ENTRY_PORT_TYPE] is the type of the optional entry port to be assigned to the userid (specified by the USERID value). It can have either of these values:

TERMINAL|CONSOLE

This parameter is only valid if ENTRY_PORT_NAME is also specified.

SIGNON_TYPE is the type of signon for the userid (specified by the USERID value). It can have any of these values:

ATTACH_SIGN_ON|DEFAULT_SIGN_ON|IRC_SIGN_ON|
LU61_SIGN_ON|LU62_SIGN_ON|NON_TERMINAL_SIGN_ON|
PRESET_SIGN_ON|USER_SIGN_ON|XRF_SIGN_ON

Output parameters

SECURITY_TOKEN is the token identifying the userid.

[SAF_RESPONSE] is the optional 32-bit SAF response code to the call.

[SAF_REASON] is the optional 32-bit SAF reason returned with SAF_RESPONSE.

[ESM_RESPONSE] is the optional 32-bit ESM response code to the call.

[ESM_REASON] is the optional 32-bit ESM reason returned with ESM_RESPONSE.

RESPONSE is the domains response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:
Security manager domain

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>APPLICATION_NOTAUTH, ENTRY_PORT_NOTAUTH, ESM_INACTIVE, ESM_TRANQUIL, GETMAIN_FAILURE, GROUP_ACCESS_REVOKED, INVALID_GROUPID, INVALID_NEW_PASSWORD, OIDCARD_NOTAUTH, OIDCARD_REQUIRED, PASSWORD_REQUIRED, PASSWORD_EXPIRED, PASSWORD_NOTAUTH, SECLABEL_FAILURE, SECURITY_INACTIVE, UNKNOWN_ESM_ERROR, USERID_NOT_IN_GROUP, USERID_REVOKED, USERID_UNDEFINED</td>
</tr>
</tbody>
</table>

**XSAD gate, ADD_USER WITHOUT PASSWORD function**

The ADD_USER WITHOUT PASSWORD function of the XSAD gate is used to add a user to the security domain *without* verification of a associated password or oidcard.

**Input parameters**

**USERID**

is the identifier of the user (a userid of 1 through 10 alphanumeric characters) to be added to the security domain.

**USERID LENGTH**

is the 8-bit length of the USERID value.

**GROUPID**

is an optional identifier, 1 through 10 alphanumeric characters, of a RACF user group to which the userid (specified by the USERID value) is to be assigned.

**GROUPID LENGTH**

is the 8-bit length of the GROUPID value. This parameter is only valid if GROUPID is also specified.

**GROUPID**

is the RACF user group to which the userid (specified by the USERID value) is to be assigned.

**ENTRY_PORT_NAME**

is an optional name of an entry port, 1 through 8 alphanumeric characters, to be assigned to the userid (specified by the USERID value).

**ENTRY_PORT_TYPE**

is the type of the optional entry port to be assigned to the userid (specified by the USERID value). It can have either of these values:

- TERMINAL
- CONSOLE

This parameter is only valid if ENTRY_PORT_NAME is also specified.

**SIGNON_TYPE**

is the type of signon for the userid (specified by the USERID value). It can have any of these values:

- ATTACH_SIGN_ON
- DEFAULT_SIGN_ON
- IRC_SIGN_ON
- LU61_SIGN_ON
- LU62_SIGN_ON
- NON_TERMINAL_SIGN_ON
- PRESET_SIGN_ON
- USER_SIGN_ON
- XRF_SIGN_ON

**Output parameters**

**SECURITY_TOKEN**

is the token identifying the userid.

**SAF_RESPONSE**

is the optional 32-bit SAF response code to the call.
Security manager domain

[SAF_REASON]
is the optional 32-bit SAF reason returned with SAF_RESPONSE.

[ESM_RESPONSE]
is the optional 32-bit ESM response code to the call.

[ESM_REASON]
is the optional 32-bit ESM reason returned with ESM_RESPONSE.

RESPONSE
is the domains response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>APPLICATION_NOTAUTH, ENTRY_PORT_NOTAUTH, ESM_INACTIVE, ESM_TRANQUIL, GETMAIN_FAILURE, GROUP_ACCESS_REVOKED, INVALID_GROUPID, SECLABEL_FAILURE, SECURITY_INACTIVE, UNKNOWN_ESM_ERROR, USERID_NOT_IN_GROUP, USERID_REVOKED, USERID_UNDEFINED</td>
</tr>
</tbody>
</table>

XSAD gate, DELETE_USER_SECURITY function
The DELETE_USER_SECURITY function of the XSAD gate is used to delete the storage held to store the ACEE and ACEE pointer for the user represented by the security token.

Input parameters
SECURITY_TOKEN
is the token identifying the userid.

SIGNOFF_TYPE
is the type of signoff for the userid identified by the SECURITY_TOKEN value. It can have any of these values:
ABNORMAL_SIGN_OFF|ATTACH_SIGN_OFF|DEFERRED_SIGN_OFF|DELETE_SIGN_OFF|LINK_SIGN_OFF|NON_TERMINAL_SIGN_OFF|PRESET_SIGN_OFF|UNFLATTEN_USER_SIGN_OFF|USER_SIGN_OFF|XRF_SIGN_OFF

Output parameters
[SAF_RESPONSE]
is the optional 32-bit SAF response code to the call.

[SAF_REASON]
is the optional 32-bit SAF reason returned with SAF_RESPONSE.

[ESM_RESPONSE]
is the optional 32-bit ESM response code to the call.

[ESM_REASON]
is the optional 32-bit ESM reason returned with ESM_RESPONSE.

RESPONSE
is the domains response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:
**XSAD gate, INQUIRE_USER_ATTRIBUTES function**

The INQUIRE_USER_ATTRIBUTES function of the XSAD gate is used to inquire about the attributes of the user represented by the security token.

**Input parameters**

**SECURITY_TOKEN**
- is the token identifying the userid.

**[USERNAME]**
- is an optional buffer into which the attributes of the user are placed.

**Output parameters**

**[USERID]**
- is the identifier of the user (a userid of 1 through 10 alphanumeric characters). The userid (specified by the SECURITY_TOKEN value) is assigned.

**USERID_LENGTH**
- is the length of the USERID value.

**[CURRENT_GROUPID]**
- is the identifier, 1 through 10 alphanumeric characters, of the current RACF user group to which the userid (specified by the SECURITY_TOKEN value) is assigned.

**[CURRENT_GROUPID_LENGTH]**
- is the 8-bit length of the GROUPID value.

**[NATIONAL_LANGUAGE]**
- is a three-character code identifying the national language for the userid. It can have any of the values in Table 93 on page 1023.

**[OPCLASS]**
- is the operator class, in the range 1 through 24, for the userid.

**[OPIDENT]**
- is the operator identification code, 1 through 3 alphanumeric characters, for the userid.

**[OPPRTYT]**
- is the operator priority value, in the range 0 through 255 (where 255 is the highest priority), for the userid.

**[TIMEOUT]**
- is the number of minutes, in the range 0 through 60, that must elapse since the user last used the terminal before CICS "times-out" the terminal.

**Notes:**
1. CICS rounds values up to the nearest multiple of 5.
2. A TIMEOUT value of 0 means that the terminal is not timed out.

**[XRFSOFF]**
- indicates whether or not you want CICS to sign off the userid following an XRF takeover. It can have either of these values:
FORCE | NOFORCE

[ACEE_PTR]
is a pointer to the access control environment element, the control block that is generated by an external security manager (ESM) when the user signs on. If the user is not signed on, the address of the CICS DFLTUSER’s ACEE is returned. If an ACEE does not exist, CICS sets the pointer reference to the null value, X’FF000000’.

[SAF_RESPONSE]
is the optional 32-bit SAF response code to the call.

[SAF_REASON]
is the optional 32-bit SAF reason returned with SAF_RESPONSE.

[ESM_RESPONSE]
is the optional 32-bit ESM response code to the call.

[ESM_REASON]
is the optional 32-bit ESM reason returned with ESM_RESPONSE.

RESPONSE
is the domains response to the call. It can have any of these values:
OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>ESTAE_FAILURE, EXTRACT_FAILURE, INVALID_ACEE, INVALID_ESM_PARAMETER, INVALID_SECURITY_TOKEN, NOTAUTH, PROFILE_UNKNOWN, SECURITY_INACTIVE</td>
</tr>
</tbody>
</table>

Table 93. National language codes (three-characters)

<table>
<thead>
<tr>
<th>Code</th>
<th>Language Name</th>
<th>Original Name</th>
<th>Principal Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFR</td>
<td>Afrikaans</td>
<td>Afrikaans</td>
<td>South Africa</td>
</tr>
<tr>
<td>ARA</td>
<td>Arabic</td>
<td>Arabi</td>
<td>Arab Countries</td>
</tr>
<tr>
<td>BEL</td>
<td>Byelorussian</td>
<td>Belaruskaja (mova)</td>
<td>Belarus</td>
</tr>
<tr>
<td>BGR</td>
<td>Bulgarian</td>
<td>Bulgarski</td>
<td>Bulgaria</td>
</tr>
<tr>
<td>CAT</td>
<td>Catalan</td>
<td>Catala</td>
<td>Spain</td>
</tr>
<tr>
<td>CHT</td>
<td>Traditional Chinese</td>
<td>Zhongwen</td>
<td>R.O.C.</td>
</tr>
<tr>
<td>CHS</td>
<td>Simplified Chinese</td>
<td>Chinese</td>
<td>P.R.C.</td>
</tr>
<tr>
<td>CSY</td>
<td>Czech</td>
<td>Cesky</td>
<td>Czechoslovakia</td>
</tr>
<tr>
<td>DAN</td>
<td>Danish</td>
<td>Dansk</td>
<td>Denmark</td>
</tr>
<tr>
<td>DEU</td>
<td>German</td>
<td>Deutsch</td>
<td>Germany</td>
</tr>
<tr>
<td>DES</td>
<td>Swiss German</td>
<td>Schweizer-Deutsch</td>
<td>Switzerland</td>
</tr>
<tr>
<td>ELL</td>
<td>Greek</td>
<td>Ellinika</td>
<td>Greece</td>
</tr>
<tr>
<td>ENA</td>
<td>Australian English</td>
<td>English</td>
<td>Australia</td>
</tr>
<tr>
<td>ENG</td>
<td>UK English</td>
<td>English</td>
<td>United Kingdom</td>
</tr>
<tr>
<td>ENU</td>
<td>US English</td>
<td>English</td>
<td>United States</td>
</tr>
<tr>
<td>ENP</td>
<td>English Upper Case</td>
<td>English</td>
<td>United States</td>
</tr>
<tr>
<td>ESP</td>
<td>Spanish</td>
<td>Espanol</td>
<td>Spain</td>
</tr>
<tr>
<td>FAR</td>
<td>Farsi</td>
<td>Persian</td>
<td>Iran</td>
</tr>
<tr>
<td>FIN</td>
<td>Finnish</td>
<td>Suomi</td>
<td>Finland</td>
</tr>
<tr>
<td>FRA</td>
<td>French</td>
<td>Francais</td>
<td>France</td>
</tr>
<tr>
<td>FRB</td>
<td>Belgian French</td>
<td>Belgian</td>
<td>Belgium</td>
</tr>
<tr>
<td>FRC</td>
<td>Canadian French</td>
<td>Canadian</td>
<td>Canada</td>
</tr>
<tr>
<td>FRS</td>
<td>Swiss French</td>
<td>Suisse-francais</td>
<td>Switzerland</td>
</tr>
</tbody>
</table>
Table 93. National language codes (three-characters) (continued)

<table>
<thead>
<tr>
<th>Code</th>
<th>Language Name</th>
<th>Original Name</th>
<th>Principal Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>GAE</td>
<td>Irish Gaelic (Irish)</td>
<td>Gaeilge</td>
<td>Ireland</td>
</tr>
<tr>
<td>HEB</td>
<td>Hebrew</td>
<td>Irvith</td>
<td>Israel</td>
</tr>
<tr>
<td>HRV</td>
<td>Croatian</td>
<td>Hrvatski</td>
<td>Croatia</td>
</tr>
<tr>
<td>HUN</td>
<td>Hungarian</td>
<td>Magyar</td>
<td>Hungary</td>
</tr>
<tr>
<td>ISL</td>
<td>Icelandic</td>
<td>Islenska</td>
<td>Iceland</td>
</tr>
<tr>
<td>ITA</td>
<td>Italian</td>
<td>Italiano</td>
<td>Italy</td>
</tr>
<tr>
<td>ITS</td>
<td>Swiss Italian</td>
<td>Italiano svizzero</td>
<td>Switzerland</td>
</tr>
<tr>
<td>JPN</td>
<td>Japanese</td>
<td>Nihongo</td>
<td>Japan</td>
</tr>
<tr>
<td>KOR</td>
<td>Korean</td>
<td>Choson-o; Hanguk-o</td>
<td>Korea</td>
</tr>
<tr>
<td>MKD</td>
<td>Macedonian</td>
<td>Makedonski</td>
<td>FYR Macedonia</td>
</tr>
<tr>
<td>NLD</td>
<td>Dutch</td>
<td>Nederlands</td>
<td>Netherlands</td>
</tr>
<tr>
<td>NLB</td>
<td>Belgian Dutch</td>
<td></td>
<td>Belgium</td>
</tr>
<tr>
<td>NOR</td>
<td>Norwegian - Bokmal</td>
<td>Norsk - Bokmal</td>
<td>Norway</td>
</tr>
<tr>
<td>NON</td>
<td>Norwegian - Nynorsk</td>
<td>Norsk - Nynorsk</td>
<td>Norway</td>
</tr>
<tr>
<td>PLK</td>
<td>Polish</td>
<td>Polski</td>
<td>Poland</td>
</tr>
<tr>
<td>PTG</td>
<td>Portuguese</td>
<td>Portugues</td>
<td>Portugal</td>
</tr>
<tr>
<td>PTB</td>
<td>Brazilian Portuguese</td>
<td></td>
<td>Brazil</td>
</tr>
<tr>
<td>RMS</td>
<td>Rhaeto-Romanic</td>
<td>Romontsch</td>
<td>Switzerland</td>
</tr>
<tr>
<td>ROM</td>
<td>Romanian</td>
<td>Romana</td>
<td>Romania</td>
</tr>
<tr>
<td>RUS</td>
<td>Russian</td>
<td>Russkij</td>
<td>Russia</td>
</tr>
<tr>
<td>SHC</td>
<td>Serbo-Croatian (Cyr)</td>
<td>Srpsko-hrvatski</td>
<td>Yugoslavia</td>
</tr>
<tr>
<td>SHL</td>
<td>Serbo-Croatian (Lat)</td>
<td></td>
<td>Yugoslavia</td>
</tr>
<tr>
<td>SKY</td>
<td>Slovakian</td>
<td>Slovensky</td>
<td>Czechoslovakia</td>
</tr>
<tr>
<td>SLO</td>
<td>Slovenian</td>
<td>Slovenski</td>
<td>Slovenia</td>
</tr>
<tr>
<td>SRL</td>
<td>Serbian (Latin)</td>
<td>Srpski (Latin)</td>
<td>Serbia-Montenegro</td>
</tr>
<tr>
<td>SRB</td>
<td>Serbian</td>
<td>Srpski</td>
<td>Serbia-Montenegro</td>
</tr>
<tr>
<td>SQI</td>
<td>Albanian</td>
<td>Shqip</td>
<td>Albania</td>
</tr>
<tr>
<td>SVE</td>
<td>Swedish</td>
<td>Svenska</td>
<td>Sweden</td>
</tr>
<tr>
<td>THA</td>
<td>Thai</td>
<td>Thai</td>
<td>Thailand</td>
</tr>
<tr>
<td>TRK</td>
<td>Turkish</td>
<td>Turkce</td>
<td>Turkey</td>
</tr>
<tr>
<td>UKR</td>
<td>Ukrainian</td>
<td>Ukrainska (mova)</td>
<td>Ukraine</td>
</tr>
<tr>
<td>URD</td>
<td>Urdu</td>
<td>Urdu</td>
<td>Pakistan</td>
</tr>
</tbody>
</table>

**XSAD gate, VALIDATE_USERID function**

The VALIDATE_USERID function of the XSAD gate is used to check whether the specified userid is valid. It is used especially when the userid has to be validated without the user being added to the system; usually because the userid was specified in a deferred START command, and the user does not need to be added to the system until the started task actually begins to execute.

**Input parameters**

USERID

is the identifier of the user (a userid of 1 through 10 alphanumeric characters) to be added to the security domain.

USERID_LENGTH

is the length of the USERID value.

**Output parameters**

RESPONSE

is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]

is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>SECURITY_INACTIVE, USERID_NOT_DEFINED, USERID_NOT_DETERMINED</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

XSFL gate, FLATTEN_USER_SECURITY function

The FLATTEN_USER_SECURITY function of the XSFL gate is used to flatten the user’s security state and place it into the FLATTENED_SECURITY buffer provided.

Input parameters

SECURITY_TOKEN

is the token identifying the userid.

FLATTENED_SECURITY

is the buffer into which the flattened security state is placed.

Output parameters

[SAF_RESPONSE]

is the optional 32-bit SAF response code to the call.

[SAF_REASON]

is the optional 32-bit SAF reason returned with SAF_RESPONSE.

[ESM_RESPONSE]

is the optional 32-bit ESM response code to the call.

[ESM_REASON]

is the optional 32-bit ESM reason returned with ESM_RESPONSE.

RESPONSE

is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]

is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ESM_ABENDED ABEND, LOOP EXCEPTION</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>INVALID_SECURITY_TOKEN, SECURITY_INACTIVE, UNKNOWN_ESM_RESPONSE</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION, INVALID_FLATTENED_BUFFER</td>
</tr>
</tbody>
</table>

XSFL gate, UNFLATTEN_USER_SECURITY function

The UNFLATTEN_USER_SECURITY function of the XSFL gate is used to unflatten the user security state data in the FLATTENED_SECURITY buffer, and add the userid to the security domain.
Security manager domain

**Input parameters**

**FLATTENED_SECURITY**

is a buffer containing flattened security state data for a userid.

**Output parameters**

**SECURITY_TOKEN**

is the token identifying the userid.

*[ACEE_PTR]*

is a pointer to the access control environment element, the control block that is generated by an external security manager (ESM) when the user signs on.

**USERID**

is the identifier of the user (a userid of 1 through 10 alphanumeric characters). The userid (specified by the SECURITY_TOKEN value) is assigned.

**USERID_LENGTH**

is the length of the USERID value.

**CURRENT_GROUPID**

is the identifier, 1 through 10 alphanumeric characters, of the current RACF user group to which the userid is assigned.

**CURRENT_GROUPID_LENGTH**

is the 8-bit length of the GROUPID value.

**ENTRY_PORT_NAME**

is the name of an entry port, 1 through 8 alphanumeric characters, for the userid.

*[ENTRY_PORT_TYPE]*

is the type of the entry port for the userid. It can have either of these values:

TERMINAL|CONSOLE|NULL

*[SAF_RESPONSE]*

is the optional 32-bit SAF response code to the call.

*[SAF_REASON]*

is the optional 32-bit SAF reason returned with SAF_RESPONSE.

*[ESM_RESPONSE]*

is the optional 32-bit ESM response code to the call.

*[ESM_REASON]*

is the optional 32-bit ESM reason returned with ESM_RESPONSE.

**RESPONSE**

is the domains response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

*[REASON]*

is returned when RESPONSE is DISASTER, EXCEPTION or INVALID.

Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ESM_ABENDED, ABEND, LOOP</td>
</tr>
</tbody>
</table>
### XSIS gate, INQUIRE_REGION_USER function

The INQUIRE_REGION_USER function of the XSIS gate is used to return the userid and groupid associated with the jobstep that is currently executing this CICS region.

**Input parameters**
None.

**Output parameters**

**REGION_USERID**
is the user identifier of the CICS jobstep (a userid of 1 through 8 alphanumeric characters).

**REGION_USERID_LENGTH**
is the length of the REGION_USERID value.

**[REGION_GROUPID]**
is the identifier, 1 through 8 alphanumeric characters, of the current RACF user group to which the region userid is assigned.

**[REGION_GROUPID_LENGTH]**
is the 8-bit length of the REGION_GROUPID value.

**RESPONSE**
is the domains response to the call. It can have any of these values: OK|EXCEPTION|DISASTER|INVALID

**[REASON]**
is returned when RESPONSE is DISASTER, EXCEPTION or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>ESM_INACTIVE</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

### XSIS gate, INQ_SECURITY_DOMAIN_PARMS function

The INQ_SECURITY_DOMAIN_PARMS function of the XSIS gate is used to return the current values of parameters from the security state data.

**Input parameters**
None.

**Output parameters**

**APPLID**
is the generic applid of the CICS region
Security manager domain

**CMDSEC**
indicates whether or the CICS region should obey the CMDSEC option specified on a transaction’s resource definition. It can have either of these values:

YES | NO

**ESMEXITS**
indicates whether or not installation data is to be passed via the RACROUTE interface to the ESM for use in user exits written for the ESM. It can have either of these values:

YES | NO

**PREFIX**
returns the value of the prefix that is being applied to all resource names in authorization requests sent to the external security manager. It can contain 0 through 8 alphanumeric characters.

**PSBCHK**
indicates whether or not DL/I security checking is to be performed for a remote terminal initiating a transaction with transaction routing. It can have either of these values:

YES | NO

**RESSEC**
indicates whether the CICS region should obey the RESSEC option specified on a transaction’s resource definition.

**SECURITY**
indicates whether or not security is active for this CICS region. It can have either of these values:

YES | NO

**XAPPC**
indicates whether or not session security checking is used when establishing APPC sessions. It can have either of these values:

YES | NO

**[XCMD]**
indicates whether or not EXEC CICS commands are checked by the ESM. It can have any of these values:

YES | name | NO

where *name* is your own resource class name for EXEC CICS commands.

**[XDCT]**
indicates whether or not destination control entries are checked by the ESM. It can have any of these values:

YES | name | NO

where *name* is your own resource class name for destination control entries.
indicates whether or not file control entries are checked by the ESM. It can have any of these values:

YES|name|NO

where name is your own resource class name for file control entries.

indicates whether or not journal entries are checked by the ESM. It can have any of these values:

YES|name|NO

where name is your own resource class name for journal entries.

indicates whether or not EXEC-started transactions entries are checked by the ESM. It can have any of these values:

YES|name|NO

where name is your own resource class name for EXEC-started transactions entries.

indicates whether or not program entries are checked by the ESM. It can have any of these values:

YES|name|NO

where name is your own resource class name for program entries.

indicates whether or not PSB entries are checked by the ESM. It can have any of these values:

YES|name|NO

where name is your own resource class name for PSB entries.

indicates whether or not attached transaction entries are checked by the ESM. It can have any of these values:

YES|name|NO

where name is your own resource class name for attached transaction entries.

indicates whether or not temporary storage entries are checked by the ESM. It can have any of these values:

YES|name|NO

where name is your own resource class name for temporary storage entries.
Security manager domain

XUSER
indicates whether or not user entries are checked by the ESM. It can have any of these values:

YES | name | NO

where name is your own resource class name for user entries.

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK | DISASTER | INVALID

[REASON]
is returned when RESPONSE is DISASTER or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

Xsis gate, SET_SECURITY_DOMAIN_PARMS function

At CICS startup, loads information for the security domain from the system initialization table (SIT) into the security state data.

Input parameters

APPLID
is the generic applid of the CICS region

CMDSEC
indicates whether or not the CICS region should obey the CMDSEC option specified on a transaction’s resource definition. It can have either of these values:

YES | NO

ESMEXITS
indicates whether or not installation data is to be passed via the RACROUTE interface to the ESM for use in user exits written for the ESM. It can have either of these values:

YES | NO

PREFIX
specifies the prefix to be applied to resource name in any authorization requests send to the external security manager. It can be 1 through 8 alphanumeric characters, or the single character ‘*’, which indicates that the CICS region userid is to be used as the prefix.

PSBCHK
indicates whether or not DL/I security checking is to be performed for a remote terminal initiating a transaction with transaction routing. It can have either of these values:

YES | NO
RESSEC
indicates whether the CICS region should obey the RESSEC option specified on a transaction’s resource definition.

SECURITY
indicates whether or not security is active for this CICS region. It can have either of these values:

YES|NO

XAPPC
indicates whether or not session security checking is used when establishing APPC sessions. It can have either of these values:

YES|NO

[XCMD]
indicates whether or not EXEC CICS commands are checked by the ESM. It can have any of these values:

YES|name|NO

where name is your own resource class name for EXEC CICS commands.

[XDCT]
indicates whether or not destination control entries are checked by the ESM. It can have any of these values:

YES|name|NO

where name is your own resource class name for destination control entries.

[XFCT]
indicates whether or not file control entries are checked by the ESM. It can have any of these values:

YES|name|NO

where name is your own resource class name for file control entries.

[XJCT]
indicates whether or not journal entries are checked by the ESM. It can have any of these values:

YES|name|NO

where name is your own resource class name for journal entries.

[XPCT]
indicates whether or not EXEC-started transactions entries are checked by the ESM. It can have any of these values:

YES|name|NO

where name is your own resource class name for EXEC-started transactions entries.

[XPPT]
indicates whether or not program entries are checked by the ESM. It can have any of these values:
Security manager domain

YES | name | NO

where name is your own resource class name for program entries.

[XPSB]
indicates whether or not PSB entries are checked by the ESM. It can have any of these values:

YES | name | NO

where name is your own resource class name for PSB entries.

[XTRAN]
indicates whether or not attached transaction entries are checked by the ESM. It can have any of these values:

YES | name | NO

where name is your own resource class name for attached transaction entries.

[XTST]
indicates whether or not temporary storage entries are checked by the ESM. It can have any of these values:

YES | name | NO

where name is your own resource class name for temporary storage entries.

XUSER
indicates whether or not user entries are checked by the ESM. It can have any of these values:

YES | name | NO

where name is your own resource class name for user entries.

Output parameters

RESPONSE
is the domains response to the call. It can have any of these values:
OK | DISASTER | INVALID

[REASON]
is returned when RESPONSE is DISASTER or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>CWA_WAIT_PHASE_FAILURE, INQUIRE_CWA_FAILURE, ABEND, LOOP</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

XSIS gate, SET_NETWORK_IDENTIFIER function

When CICS issues an OPEN ACB for VTAM, the CICS SVC is invoked to store the name (netid) of the local network combined with the local luname, and to RAclist the profiles in the External Security Manager (ESM) APPCLU Class. If you have specified either of the SEC=NO or XAPPC=NO system initialization parameters, no action is performed, and the return code is set to OK.
If the RAclist fails, and the CONDITIONAL parameter is NO, then CICS is terminated.

**Input parameters**

**LOCAL_LUNAME**

is the VTAM LU name of the local CICS region.

**LOCAL_LUNAME_LENGTH**

is the length of the VTAM LU name specified by LOCAL_LUNAME.

**CONDITIONAL**

indicates whether or not CICS can tolerate errors in XSIS calls due to the APPCLU profiles not being in storage (LU6.2 connections cannot be validated). It can have either of these values:

YES | NO

**Output parameters**

**RESPONSE**

is the domains response to the call. It can have any of these values:

OK | DISASTER | INVALID | PURGED

**[REASON]**

is returned when RESPONSE is DISASTER or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

**XSLU gate, GENERATE_APPC_BIND function**

The GENERATE_APPC_BIND function of the XSLU gate generates a random number which is sent to the partner LU for partner verification.

**Input parameters**

None

**Output parameters**

**RANDOM_STRING**

A random eight-character string.

**RESPONSE**

is the domains response to the call. It can have any of these values:

OK | INVALID

**[REASON]**

is returned when RESPONSE is INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT INVALID_FUNCTION</td>
</tr>
</tbody>
</table>
XSLU gate, GENERATE_APPC_RESPONSE function

The GENERATE_APPC_RESPONSE function of the XSLU gate encrypts the string received from the LU partner, and generates a new random string for the partner to validate.

Input parameters

LOCAL_LUNAME
is the VTAM LU name of the local CICS region (sending the response).

REMOTE_LUNAME
is the VTAM LU name of the remote CICS region (that sent the bind).

TEST_STRING
is a random eight-character string receive with a bind request (RANDOM_STRING of the GENERATE_APPC_BIND function).

Output parameters

ENCRYPTED_TEST_STRING
is an eight-character string formed by encrypting the test string using shared DES (Data Encryption Standard/System) encryption keys.

RANDOM_STRING
is a random eight-character string.

[SAF_RESPONSE]
is the optional 32-bit SAF response code to the call.

[SAF_REASON]
is the optional 32-bit SAF reason returned with SAF_RESPONSE.

[ESM_RESPONSE]
is the optional 32-bit ESM response code to the call.

[ESM_REASON]
is the optional 32-bit ESM reason returned with ESM_RESPONSE.

RESPONSE
is the domains response to the call. It can have any of these values: OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP, ESM_ABENDED, ESTAE_FAILURE, EXTRACT_FAILURE</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>NOTAUTH, PROFILE_UNKNOWN, PROFILE_LOCKED, PROFILE_EXPIRED,</td>
</tr>
<tr>
<td></td>
<td>SESSION_KEY_NULL, SECURITY_INACTIVE, UNKNOWN_ESM_RESPONSE</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

XSLU gate, VALIDATE_APPC_RESPONSE function

The VALIDATE_APPC_RESPONSE function of the XSLU gate encrypts the string that was previously sent to the partner, and compares it with the encrypted string received from the partner.
Security manager domain

**Input parameters**

**LOCAL_LUNAME**

is the VTAM LU name of the local CICS region (validating the response).

**REMOTE_LUNAME**

is the VTAM LU name of the remote CICS region (that sent the response).

**TEST_STRING**

is a random eight-character string receive with a validate request (RANDOM_STRING of the GENERATE_APPC_RESPONSE function).

**ENCRYPTED_TEST_STRING**

is an eight-character string formed by encrypting the test string using shared DES (Data Encryption Standard/System) encryption keys.

**Output parameters**

**[SAF_RESPONSE]**

is the optional 32-bit SAF response code to the call.

**[SAF_REASON]**

is the optional 32-bit SAF reason returned with SAF_RESPONSE.

**[ESM_RESPONSE]**

is the optional 32-bit ESM response code to the call.

**[ESM_REASON]**

is the optional 32-bit ESM reason returned with ESM_RESPONSE.

**RESPONSE**

is the domains response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

**[REASON]**

is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID.

Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP, ESM_ABENDED, ESTAE_FAILURE, EXTRACT_FAILURE</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>NOTAUTH, VALIDATION_ERROR, PROFILE_UNKNOWN, PROFILE_LOCKED, PROFILE_EXPIRED, SESSION_KEY_NULL, SECURITY_INACTIVE, UNKNOWN_ESM_RESPONSE</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

**XSPW gate, CREATE_PASSTICKET function**

The CREATE_PASSTICKET function of the XSPW gate is used to create a RACF PassTicket (an alternative to a password). When created, the RACF PassTicket can be presented for userid verification *once only*.

**Input parameters**

**APPLID**

is the application identifier for the CICS region.

**[TRANSACTION_NUMBER]**

is an optional number that identifies a transaction from which the caller’s security token is located. If not specified, the caller’s security token is located from the principal security token associated with the current CICS task.
Security manager domain

Output parameters

PASSTICKET
   is the 10-character passticket to be used for the CICS region specified by
   the APPLID value.

PASSTICKET_LENGTH
   is the 8-bit length of the PASSTICKET value.

[ESM_RESPONSE]
   is the optional 32-bit ESM response code to the call.

[ESM_REASON]
   is the optional 32-bit ESM reason returned with ESM_RESPONSE.

RESPONSE
   is the domains response to the call. It can have any of these values:
   OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
   is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID.
   Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>FUNCTION_UNAVAILABLE, PASSTICKET_NOT_CREATED, SECURITY_INACTIVE,</td>
</tr>
<tr>
<td></td>
<td>TRANSACTION_NOT_FOUND, UNKNOWN_ESM_ERROR</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_APPLID, INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

XSPW gate, INQUIRE_PASSWORD_DATA function

The INQUIRE_PASSWORD_DATA function of the XSPW gate provides information from the ESM.

Input parameters

USERID
   is the identifier of the user (a userid of 1 through 10 alphanumeric
   characters) requesting the ESM information.

USERID_LENGTH
   is the length of the USERID value.

PASSWORD
   is the password, 1 through 10 alphanumeric characters, for the userid
   specified by the USERID value.

PASSWORD_LENGTH
   is the 8-bit length of the PASSWORD value.

Output parameters

[DAYS_LEFT]
   is the number of days left before the password must be changed.

[PASSWORD_FAILURES]
   is the number of times that the user has unsuccessfully entered tried to
   enter the password.

[EXPIRY_ABSTIME]
   is the date and time of when the password will expire.
is the date and time of when the password was last used.

[CHANGE_ABSTIME]
is the date and time of when the password was last changed.

[SAF_RESPONSE]
is the optional 32-bit SAF response code to the call.

[SAF_REASON]
is the optional 32-bit SAF reason returned with SAF_RESPONSE.

[ESM_RESPONSE]
is the optional 32-bit ESM response code to the call.

[ESM_REASON]
is the optional 32-bit ESM reason returned with ESM_RESPONSE.

RESPONSE is the domains response to the call. It can have any of these values:
OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP, ESM_ABENDED, ESTAE_FAILURE, EXTRACT_FAILURE</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>ESM_INACTIVE, PASSWORD_NOTAUTH, SECURITY_INACTIVE, UNKNOWN_ESM_ERROR, NOTAUTH, USERID_UNDEFINED, PASSWORD_EXPIRED, USERID_REVOKED</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

XSPW gate, UPDATE_PASSWORD_DATA function
The UPDATE_PASSWORD_DATA function of the XSPW gate assigns a new password to the userid, if the current password is input correctly and the new password meets ESM and installation defined password quality rules.

Input parameters

USERID
is the identifier of the user (a userid of 1 through 10 alphanumeric characters) requesting the ESM information.

USERID_LENGTH
is the length of the USERID value.

PASSWORD
is the current password, 1 through 10 alphanumeric characters, for the userid specified by the USERID value.

PASSWORD_LENGTH
is the 8-bit length of the PASSWORD value.

NEW_PASSWORD
is the new password, 1 through 10 alphanumeric characters, for the userid specified by the USERID value.

NEW_PASSWORD_LENGTH
is the 8-bit length of the NEW_PASSWORD value.
Output parameters

[SAF_RESPONSE]
  is the optional 32-bit SAF response code to the call.

[SAF_REASON]
  is the optional 32-bit SAF reason returned with SAF_RESPONSE.

[ESM_RESPONSE]
  is the optional 32-bit ESM response code to the call.

[ESM_REASON]
  is the optional 32-bit ESM reason returned with ESM_RESPONSE.

RESPONSE
  is the domains response to the call. It can have any of these values:
  OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
  is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID.
  Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP, ESM_ABENDED, ESTAE_FAILURE, EXTRACT_FAILURE</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>USERID_REVOKED, USERID_UNDEFINED, SECLABEL_FAILURE, PASSWORD_NOTAUTH, INVALID_NEW_PASSWORD, ESM_INACTIVE, SECURITY_INACTIVE, UNKNOWN_ESM_ERROR</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

XSRC gate, CHECK_CICSRESOURCE function

The CHECK_CICSRESOURCE function of the XSRC gate performs CICS resource access checks.

Input parameters

RESOURCE
  is the name of the resource, padded with blanks to eight-characters.

RESOURCE_TYPE
  is the type of the resource. It can have any of these values:
  FILE|JOURNAL|PROGRAM|PSB|TDQUEUE|TRANSACTION|TRANSATTACH|TSQUEUE

ACCESS
  is the type of access to be made on the resource. It can have any of these values:
  EXECUTE|READ|UPDATE

[LOGMESSAGE]
  indicates (optionally) whether access failures are logged to the CSCS transient data queue and the MVS System Management Facility (SMF). It can have either of these values:
  YES|NO

[FORCE]
  indicates (optionally) whether or not security checking is forced regardless of the setting of RESSEC in the Security Domain’s transaction token. It can have either of these values:
  YES|NO
Output parameters

[FAILING_USERID]  
is the userid that failed to access the resource.

[FAILING_USERID_LENGTH]  
is the length of the userid (specified by the FAILING_USERID value).

[SAF_RESPONSE]  
is the optional 32-bit SAF response code to the call.

[SAF_REASON]  
is the optional 32-bit SAF reason returned with SAF_RESPONSE.

[ESM_RESPONSE]  
is the optional 32-bit ESM response code to the call.

[ESM_REASON]  
is the optional 32-bit ESM reason returned with ESM_RESPONSE.

RESPONSE  
is the domains response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]  
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NOTAUTH</td>
</tr>
</tbody>
</table>

XSRC gate, CHECK_CICS_COMMAND function

The CHECK_CICS_COMMAND function of the XSRC gate performs CICS command access checks.

Input parameters

RESOURCE_TYPE  
is the type of the resource. It can have any of these values:
AUTINSTMODEL|AUTOINSTALL|CONNECTION|DELETESHIPPED|DSNAME|DUMP|DUMPPDS|EXITPROGRAM|FEPIRESOURCE|FILE|IRBATCH|IRC|JOURNAL|JOURNALMODEL|LINE|LSPOOL|MAPSET|MODENAME|MONITOR|NONVTAM|PARTITIONSET|PARTNER|PROFILE|PROGRAM|PSB|REQID|RESETTIME|SECURITY|SESSIONS|SHUTDOWN|STATISTICS|STORAGE|STREAMNAME|SYSDUMPCODE|SYSTEM|TASK|TCLASS|TDQUEUE|TERMINAL|TIME|TRACE|TRACEDEST|TRACEFLAG|TRACETYPE|TRANSDUMPCODE|TRANSACT|TRANSATTACH|TSQUEUE|TYPETERM|UOW|UOWDSNFAIL|UOWNQ|UOWLINK|VOLUME|VTAM

ACCESS  
is the type of access to be made on the resource. It can have any of these values:
COLLECT|DEFINE|DISCARD|INQUIRE|PERFORM|SET

[LOGMESSAGE]  
indicates (optionally) whether access failures are logged to the CSCS transient data queue and the MVS System Management Facility (SMF). It can have either of these values:
YES|NO
Security manager domain

[FORCE]
indicates (optionally) whether or not security checking is forced regardless of the setting of RESSEC in the Security Domain's transaction token. It can have either of these values:
YES|NO

Output parameters

[FAILING_USERID]
is the userid that failed to access the resource.

[FAILING_USERID_LENGTH]
is the length of the userid (specified by the FAILING_USERID value).

[SAF_RESPONSE]
is the optional 32-bit SAF response code to the call.

[SAF_REASON]
is the optional 32-bit SAF reason returned with SAF_RESPONSE.

[ESM_RESPONSE]
is the optional 32-bit ESM response code to the call.

[ESM_REASON]
is the optional 32-bit ESM reason returned with ESM_RESPONSE.

RESPONSE
is the domains response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NOTAUTH</td>
</tr>
</tbody>
</table>

XSRC gate, CHECK_SURROGATE_USER function

The CHECK_SURROGATE_USER function of the XSRC gate performs surrogate user checking.

Input parameters

USERID
is the identifier of the surrogate user (a userid of 1 through 10 alphanumeric characters).

USERID_LENGTH
is the length of the USERID value.

ACCESS
is the type of access requested. It can have any of these values:
INSTALL|START|CHANGE

[LOGMESSAGE]
indicates (optionally) whether access failures are logged to the CSCS transient data queue and the MVS System Management Facility (SMF). It can have either of these values:
YES|NO
indicates (optionally) whether or not security checking is forced regardless of the setting of RESSEC in the Security Domain’s transaction token. It can have either of these values:

| YES | NO |

Output parameters

[FAILING_USERID] is the userid that failed to access the resource.

[FAILING_USERID_LENGTH] is the length of the userid (specified by the FAILING_USERID value).

[SAF_RESPONSE] is the optional 32-bit SAF response code to the call.

[SAF_REASON] is the optional 32-bit SAF reason returned with SAF_RESPONSE.

[ESM_RESPONSE] is the optional 32-bit ESM response code to the call.

[ESM_REASON] is the optional 32-bit ESM reason returned with ESM_RESPONSE.

RESPONSE is the domains response to the call. It can have any of these values:

| OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED |

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NOTAUTH</td>
</tr>
</tbody>
</table>

XSRC gate, CHECK_NON_CICS_RESOURCE function

The CHECK_NON_CICSRESOURCE function of the XSRC gate performs non-CICS resource access checks.

Input parameters

RESOURCE_NAME is the address and length of the resource name, in the form RESOURCE_NAME(addr,length).

CLASSNAME is the ESM class name in which the resource is defined.

ACCESS is the type of access to be made on the resource. It can have any of these values:

| ALTER | CONTROL | READ | UPDATE |

[LOGMESSAGE] indicates (optionally) whether access failures are logged to the CSCS transient data queue and the MVS System Management Facility (SMF). It can have either of these values:

| YES | NO |
Security manager domain

Output parameters

[Failing_userid] is the userid that failed to access the resource.

[Failing_userid_length] is the length of the userid (specified by the Failing_userid value).

[saferesponse] is the optional 32-bit SAF response code to the call.

[safrational] is the optional 32-bit SAF reason returned with SAF_response.

[esmresponse] is the optional 32-bit ESM response code to the call.

[esmration] is the optional 32-bit ESM reason returned with ESM_response.

Response is the domains response to the call. It can have any of these values:

OK, EXCEPTION, DISASTER, INVALID, KERNERROR, PURGED

[reason] is returned when Response is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NOTAUTH, ESM_NOT_PRESENT, ESM_INACTIVE, RESOURCE_NOT_FOUND, CLASS_NOT_FOUND, INVALID_RESOURCE_NAME</td>
</tr>
</tbody>
</table>

XSRC gate, REBUILD_RESOURCE_CLASSES function

The REBUILD_RESOURCE_CLASSES function of the XSRC gate rebuilds the resource-class profiles.

Input parameters
None.

Output parameters

[Failing_userid] is the userid that failed to access the resource.

[Failing_userid_length] is the length of the userid (specified by the Failing_userid value).

[saferesponse] is the optional 32-bit SAF response code to the call.

[safrational] is the optional 32-bit SAF reason returned with SAF_response.

[esmresponse] is the optional 32-bit ESM response code to the call.

[esmration] is the optional 32-bit ESM reason returned with ESM_response.

Response is the domains response to the call. It can have any of these values:

OK, EXCEPTION, DISASTER, INVALID, KERNERROR, PURGED
**XSXM gate, ADD_TRANSACTION_SECURITY function**

The `ADD_TRANSACTION_SECURITY` function of the XSXM gate sets the transaction options input to be stored as extended security tokens maintained by the transaction manager.

**Input parameters**

- **[PRINCIPAL_SECURITY_TOKEN]** is the optional principal security token.
- **[SESSION_SECURITY_TOKEN]** is the optional session security token.
- **[EDF_SECURITY_TOKEN]** is the optional EDF security token.

**Output parameters**

**RESPONSE**

is the domains response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

**[REASON]**

is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>GETMAIN_FAILED</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>NO_SECURITY_TOKEN</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

**XSXM gate, DEL_TRANSACTION_SECURITY function**

The `DEL_TRANSACTION_SECURITY` function of the XSXM gate deletes the security token of the specified token type for the transaction.

**Input parameters**

**TOKEN_TYPE**

is the type of security token for the transaction. It can have any of these values:

PRINCIPAL | SESSION | EDF

**Output parameters**

**RESPONSE**

is the domains response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

**[REASON]**

is returned when RESPONSE is INVALID. Possible values are:
Security manager domain

RESPONSE

Possible REASON values

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>REASON</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

Security manager domain’s generic gates

Table 94 summarizes the security manager domain’s generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 94. Security manager domain’s generic gates

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>XSDM</td>
<td>XS 0101</td>
<td>INITIALIZE_DOMAIN</td>
<td>DMDM</td>
</tr>
<tr>
<td></td>
<td>XS 0102</td>
<td>QUIESCE_DOMAIN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TERMINATE_DOMAIN</td>
<td></td>
</tr>
</tbody>
</table>

For descriptions of these functions and their input and output parameters, refer to the sections dealing with the corresponding generic formats:

Functions and parameters

Format DMDM—"Domain manager domain’s generic formats” on page 377

In initialization processing, the security manager domain performs internal routines, and sets the initial security options, as for XSIS gate, SET_SECURITY_DOMAIN_PARMS function” on page 1030.

For all starts the information comes from the system initialization parameters.

Security manager domain also issues console messages during initialization to report whether or not security is active.

In quiesce and termination processing, the security manager domain performs internal routines only.

Modules

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHXSAD</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>ADD_USER_WITH_PASSWORD,</td>
</tr>
<tr>
<td></td>
<td>ADD_USER_WITHOUT_PASSWORD</td>
</tr>
<tr>
<td></td>
<td>DELETE_USER_SECURITY</td>
</tr>
<tr>
<td></td>
<td>INQUIRE_USER_ATTRIBUTES</td>
</tr>
<tr>
<td>DFHXSDM</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>INITIALIZE_DOMAIN</td>
</tr>
<tr>
<td></td>
<td>QUIESCE_DOMAIN</td>
</tr>
<tr>
<td></td>
<td>TERMINATE_DOMAIN</td>
</tr>
<tr>
<td>DFHXSDUF</td>
<td>XS domain offline dump formatting routine</td>
</tr>
</tbody>
</table>
Module Function

DFHXSFL Handles the following requests:
   FLATTEN_USER_SECURITY
   UNFLATTEN_USER_SECURITY

DFHXSI S Handles the following requests:
   INQUIRE_SECURITY_DOMAIN_PARMS
   INQUIRE_REGION_USERID
   SET_SECURITY_DOMAIN_PARMS
   SET_NETWORK_IDENTIFIER

DFHXSLU Handles the following requests:
   GENERATE_APPC_BIND
   GENERATE_APPC_RESPONSE
   VALIDATE_APPC_RESPONSE

DFHXSPW Handles the following requests:
   INQUIRE_PASSWORD_DATA
   UPDATE_PASSWORD

DFHXSRC Handles the following requests:
   CHECK_CICS_Resource
   CHECK_CICS_COMMAND
   CHECK_NON_CICS_Resource
   CHECK_SURROGATE_USER
   REBUILD_RESOURCE_CLASSES

DFHXSSA Manages the routing of all security domain supervisor requests, and handles those requests that are concerned with adding and deleting users.

DFHXSSB Handles all the supervisor state interfaces with the ESM that are concerned with extracting data from the ESM’s database.

DFHXSSC Handles all the supervisor state interfaces with the ESM that are concerned with resource checking, including the building and deleting of in-storage profiles for the use of the resource check functions.

DFHXSSD Handles supervisor state interfaces with RACF that are concerned with PassTicket generation.

DFHXSSI Handles the following requests:
   DEACTIVATE_SECURITY
   INITIALIZE_SECURITY_SVC
   TERMINATE_SECURITY_SVC

DFHXSTRI Interprets XS domain trace entries

Exits

No global user exit points are provided in this domain.

Trace

The point IDs for the security manager domain are of the form XS xxxx; the corresponding trace levels are XS 1 and Exc.

For more information about the trace points, see the [CICS Trace Entries](#). For more information about using traces in problem determination, see the [CICS Problem Determination Guide](#).
The following external call is used by the security manager:

- MVS RACROUTE macro to request ESM services
Chapter 76. Sign-on component

The sign-on routine is a component of terminal control that associates users with terminals, connections, and sessions. Because it is a part of terminal control, it is logically part of the application (AP) domain.

Sign-on component’s subroutines

The sign-on component is entered as a single kernel-managed subroutine, DFHSNUS, which handles some function itself and also acts as a router to further kernel-managed subroutines. Table 95 summarizes the sign-on component’s subroutines. It shows the level-1 trace point IDs of the modules providing the functions for the subroutines, the functions provided by the subroutines, and whether or not the functions are available through the exit programming interface (XPI).

Table 95. Sign-on component’s subroutines

<table>
<thead>
<tr>
<th>Subroutine</th>
<th>Trace</th>
<th>Function</th>
<th>XPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHSNAS</td>
<td>AP 2050</td>
<td>SIGNON_ATI_SESSION</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>AP 2051</td>
<td>SIGNOFF_ATI_SESSION</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>AP 2052</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AP 2053</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AP 2054</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AP 2055</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AP 2056</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DFHSNPU</td>
<td>AP 2070</td>
<td>SIGNON_PRESET_USERID</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>AP 2071</td>
<td>SIGNOFF_PRESET_USERID</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>AP 2072</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AP 2073</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AP 2074</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AP 2075</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AP 2076</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AP 2077</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AP 2078</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AP 2079</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DFHSNSG</td>
<td>AP 20C0</td>
<td>SIGNOFF_SURROGATE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>AP 20C1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AP 20C2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AP 20C3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AP 20C4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AP 20C5</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AP 20C6</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Sign-on Component

Table 95. Sign-on component's subroutines (continued)

<table>
<thead>
<tr>
<th>Subroutine</th>
<th>Trace</th>
<th>Function</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHSNSU</td>
<td>AP 2060</td>
<td>SIGNON_SESSION_USERID</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AP 2061</td>
<td>SIGNOFF_SESSION_USERID</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AP 2062</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AP 2063</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AP 2064</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AP 2065</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AP 2066</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AP 2067</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AP 2068</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AP 2069</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AP 206A</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AP 206B</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AP 206C</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AP 206D</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DFHSNTU</td>
<td>AP 2080</td>
<td>SIGNON_TERMINAL_USER</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AP 2081</td>
<td>SIGNOFF_TERMINAL_USER</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AP 2082</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AP 2083</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AP 2084</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AP 2085</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AP 2086</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AP 2087</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AP 2088</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AP 2089</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AP 208A</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AP 208B</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AP 208C</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AP 208D</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AP 208E</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AP 208F</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AP 2090</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AP 2091</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AP 2092</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AP 2093</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AP 2094</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AP 2095</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AP 2096</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AP 2097</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DFHSNUS</td>
<td>AP 2040</td>
<td>SIGNON_ATTACH_HEADER</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AP 2041</td>
<td>SIGNOFF_ATTACH_HEADER</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AP 2042</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AP 2043</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AP 2044</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AP 2045</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AP 2046</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AP 2047</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AP 2048</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AP 2049</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

DFHSNAS subroutine, SIGNON_ATI_SESSION function

The SIGNON_ATI_SESSION function of the DFHSNAS subroutine signs on the appropriate userid to a session when that session is being used by a trigger transaction specified in a DCT with DESTFAC=SYSTEM.
**Input parameters**

**SESSION_TCTTE_PTR**

is the address of the TCTTE for the session to be signed on.

**Output parameters**

**RESPONSE**

is the subroutine’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[**[REASON]**]

is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP, UNEXPECTED_REASON, CORRUPT_USER_TOKEN, USER_DOMAIN_FAILURE, USER_TOKEN_MISMATCH</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>INVALID_TERMINAL_TYPE, TERMINAL_ALREADY_SIGNED_ON, SURROGATE_TERMINAL, SECURITY_INACTIVE</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

**DFHSNAS subroutine, SIGNOFF_ATI_SESSION function**

The SIGNOFF_ATI_SESSION function of the DFHSNAS subroutine is used to reverse the effect of a SIGNON_ATI_SESSION.

**Input parameters**

**SESSION_TCTTE_PTR**

is the address of the session TCTTE to be signed off.

**Output parameters**

**RESPONSE**

is the subroutine’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[**[REASON]**]

is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP, UNEXPECTED_REASON, CORRUPT_USER_TOKEN, USER_DOMAIN_FAILURE, INVALID_USER_TOKEN</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>TERMINAL_ALREADY_SIGNED_ON, SURROGATE_TERMINAL, SECURITY_INACTIVE, INVALID_TERMINAL_TYPE</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

**DFHSNPU subroutine, SIGNON_PRESET_USERID function**

The SIGNON_PRESET_USERID function of the DFHSNPU subroutine is used to sign on the userid specified in a terminal definition when that terminal is installed.

**Input parameters**

**USERID**

is the userid to be assigned to the terminal.
Sign-on Component

USERID_LENGTH
is the length of the userid.

TCTTE_PTR
is the address of the TCTTE for the terminal to be given preset security.

[NATLANG_SUFFIX]
is an optional one-character national language code to be assigned to the terminal, which will override any national language associated with the userid.

[MESAGE]
is an optional parameter that specifies whether a message is to be issued when the sign on completes successfully. It can have either of these values: YES|NO

Output parameters

[SAF_RESPONSE]
is the optional 32-bit SAF response code to the call.

[SAF_REASON]
is the optional 32-bit SAF reason returned with SAF_RESPONSE.

[ESM_RESPONSE]
is the optional 32-bit ESM response code to the call.

[ESM_REASON]
is the optional 32-bit ESM reason returned with ESM_RESPONSE.

RESPONSE
is the subroutine’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP, EXCEPTION_UNKNOWN, CORRUPT_USER_TOKEN, MESSAGE_DOMAIN_FAILURE, USER_DOMAIN_FAILURE, GETMAIN_FAILED</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>INVALID_USERID, INVALID_NATIONAL_LANGUAGE, TERMINAL_ALREADY_SIGNED_ON, UNKNOWN_ESM_RESPONSE, SECURITY_INACTIVE, ESM_INACTIVE, TERMINAL_NOTAUTH, APPLICATION_NOTAUTH, USERID_REVOKED, TERMINAL_NOT_PRESET, GROUP_ACCESS_NOTAUTH, UNAVAILABLE_NATLANG, SECLABEL_CHECK_FAILED, ESM_TRANQUIL</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

DFHSNPU subroutine, SIGNOFF_PRESET_USERID function

The SIGNOFF_PRESET_USERID function of the DFHSNPU subroutine is used to sign off a preallocated userid from a terminal before it is deleted.

Input parameters

TCTTE_PTR
is the address of the TCTTE for the terminal from which preset security is to be removed.
[MESSAGE]
is an optional parameter that specifies whether a message is to be issued when the sign off completes successfully. It can have either of these values:

YES | NO

Output parameters

[SAF_RESPONSE]
is the optional 32-bit SAF response code to the call.

[SAF_REASON]
is the optional 32-bit SAF reason returned with SAF_RESPONSE.

[ESM_RESPONSE]
is the optional 32-bit ESM response code to the call.

[ESM_REASON]
is the optional 32-bit ESM reason returned with ESM_RESPONSE.

RESPONSE
is the subroutine’s response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP, CORRUPT_USER_TOKEN, FREEMAIN_FAILED</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>TERMINAL_NOT_SIGNED_ON, TERMINAL_NOT_PRESET, SECURITY_INACTIVE, ESM_INACTIVE, ESM_TRANQUIL, UNKNOWN_ESM_RESPONSE</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

DFHSNSG subroutine, SIGNOFF_SURROGATE function
The SIGNOFF_SURROGATE function of the DFHSNSG subroutine is used to sign off a userid from a surrogate terminal that is about to be deleted by the remote terminal builder. (The equivalent sign-on routine is always performed as an inline function, so no SIGNON call to DFHSNSG is ever traced.)

Input parameters
TCTTE_PTR
is the address of the TCTTE for the surrogate terminal being signed off.

SESSION_TCTTE_PTR
is the address of the TCTTE for the associated relay session.

Output parameters
RESPONSE
is the subroutine’s response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

DFHSNSU subroutine, SIGNON_SESSION_USERID function
The SIGNON_SESSION_USERID function of the DFHSNSU subroutine is used to sign on the USERID (from the SESSIONS definition) or the SECURITYNAME (from the CONNECTION definition) for IRC, LU6.1, and LU6.2 sessions.
Sign-on Component

**Input parameters**

- **[USERID]** is the userid to be signed on.
- **[USERID_LENGTH]** is the length of the userid to be signed on.
- **SESSION_TCTTE_PTR** is the address of the TCTTE for the session being signed on.

**Output parameters**

- **RESPONSE** is the subroutine’s response to the call. It can have any of these values:
  - OK
  - EXCEPTION
  - DISASTER
  - INVALID
  - KERNERROR
  - PURGED

- **[REASON]** is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP, USER_TOKEN_MISMATCH, MESSAGE_DOMAIN_FAILURE, SURROGATE_TERMINAL, USER_DOMAIN_FAILURE, XS_DOMAIN_FAILURE</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>INVALID_USERID, INVALID_TERMINAL_TYPE, TERMINAL_ALREADY_SIGNED_ON, UNKNOWN_ESM_RESPONSE, SECURITY_INACTIVE, ESM_INACTIVE, APPLICATION_NOTAUTH, USERID_REVOKED, GROUP_ACCESS_REVOKED, SECLABEL_CHECK_FAILED, ESM_TRANQUIL</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

**DFHSNSU subroutine, SIGNOFF_SESSION_USERID function**

The SIGNOFF_SESSION_USERID function of the DFHSNSU subroutine is used to reverse the effect of the SIGNON_SESSION_USERID function.

**Input parameters**

None

**Output parameters**

- **[SAF_RESPONSE]** is the optional 32-bit SAF response code to the call.
- **[SAF_REASON]** is the optional 32-bit SAF reason returned with SAF_RESPONSE.
- **[ESM_RESPONSE]** is the optional 32-bit ESM response code to the call.
- **[ESM_REASON]** is the optional 32-bit ESM reason returned with ESM_RESPONSE.
- **RESPONSE** is the subroutine’s response to the call. It can have any of these values:
  - OK
  - EXCEPTION
  - DISASTER
  - INVALID
  - KERNERROR
  - PURGED

- **[REASON]** is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:
### RESPONSE  Possible REASON values

<table>
<thead>
<tr>
<th>DISASTER</th>
<th>ABEND, LOOP, TERMINAL_NOT_SIGNED_ON, CORRUPT_USER_TOKEN, INVALID_TERMINAL_TYPE, SURROGATE_TERMINAL, SECOND_DELETE_FAILED, MESSAGE_DOMAIN_FAILURE, USER_DOMAIN_FAILURE</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>SECURITY_INACTIVE, ESM_INACTIVE, ESM_TRANQUIL, UNKNOWN_ESM_RESPONSE</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

**DFHSNTU subroutine, SIGNON_TERMINAL_USER function**

The SIGNON_TERMINAL_USER function of the DFHSNTU subroutine is used to implement the EXEC CICS SIGNON command and signs on a specific user at the principal facility terminal.

**Input parameters**

- **USERID**
  - is the userid being signed on to the principal facility terminal.

- **USERID_LENGTH**
  - is the length of the userid.

- **[PASSWORD]**
  - is the optional password associated with the userid. The external security manager determines whether the password is required or not.

- **[PASSWORD_LENGTH]**
  - is the length of the password.

- **[NEW_PASSWORD]**
  - is the optional new password that is to replace the existing password

- **[NEW_PASSWORD_LENGTH]**
  - is the length of the new password.

- **[OIDCARD]**
  - is the text obtained from an operator identification card. The external security manager determines whether operator identification card data, or a password, or both, or neither, are required.

- **[GROUPID]**
  - is the optional group name to be associated with the userid for this sign on.

- **[GROUPID_LENGTH]**
  - is the length of the group name.

- **[NATIONAL_LANGUAGE]**
  - is the optional three-letter national language code to be associated with the terminal for the duration of this sign on. The code should be one of those specified in [Table 93 on page 1023](#).

- **[SCOPE_CHECK]**
  - is an optional parameter that specifies whether this sign on is to be subject to the constraints imposed by the SNSCOPE system initialization parameter. It can have either of these values: YES | NO

**Output parameters**

- **[SAF_RESPONSE]**
  - is the optional 32-bit SAF response code to the call.
**Sign-on Component**

[Saf_reason] is the optional 32-bit SAF reason returned with SAF_RESPONSE.

[Esm_response] is the optional 32-bit ESM response code to the call.

[Esm_reason] is the optional 32-bit ESM reason returned with ESM_RESPONSE.

**Response**

is the subroutine’s response to the call. It can have any of these values:

**Ok** | **Exception** | **Disaster** | **Invalid** | **Kernerror** | **Purged**

[Reason] is returned when Response is Disaster, Exception, or Invalid.

Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disaster</td>
<td>ABEND, LOOP, ADD_USER_FAILURE, GETMAIN_FAILED, EXCEPTION_UNKNOWN, INQUIRE_DEFAULT_ERROR, MESSAGE_DOMAIN_FAILURE, USER_DOMAIN_FAILURE, XMIQ_FAILURE, TCAM_POOL_STATE_ERROR, CORRUPT_USER_TOKEN, SNXR_FAILURE, SUSX_FAILURE</td>
</tr>
<tr>
<td>Exception</td>
<td>INVALID_USERID, INVALID_PASSWORD, INVALID_NEW_PASSWORD, INVALID_OIDCARD, INVALID_GROUPID, USERID_NOT_IN_GROUP, INVALID_TERMINAL_TYPE, INVALID_NATIONAL_LANGUAGE, UNAVAILABLE_NATLANG, Terminal_ALREADY_SIGNED_ON, USERID_ALREADY_SIGNED_ON, SURROGATE_TERMINAL, PRESET_SECURITY_TERMINAL, NO_TERMINAL_WITH_TASK, USERID_REQUIRED, PASSWORD_REQUIRED, NEW_PASSWORD_REQUIRED, OIDCARD_REQUIRED, UNKNOWN_ESM_RESPONSE, SECURITY_INACTIVE, ESM_INACTIVE, TERMINAL_NOTAUTH, APPLICATION_NOTAUTH, USERID_REVOKED, GROUP_ACCESS_REVOKED, SECLABEL_CHECK_FAILED, ESM_TRANQUIL</td>
</tr>
<tr>
<td>Invalid</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

**DFHSNTU subroutine, Signoff_Terminal_User function**

The SIGNOFF_TERMINAL_USER function of the DFHSNTU subroutine is used to implement the EXEC CICS SIGNOFF command and reverses the effect of a SIGNON_TERMINAL_USER function. It effectively associates the terminal with the default userid specified in the DFLTUSER system initialization parameter.

**Input parameters**

[Tctte_ptr] is the optional TCTTE address of a terminal that is to be signed off.

**Output parameters**

[Saf_response] is the optional 32-bit SAF response code to the call.

[Saf_reason] is the optional 32-bit SAF reason returned with SAF_RESPONSE.

[Esm_response] is the optional 32-bit ESM response code to the call.

[Esm_reason] is the optional 32-bit ESM reason returned with ESM_RESPONSE.

**Response**

is the subroutine’s response to the call. It can have any of these values:
[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, FREEMAIN_FAILED, LOOP, XMIQ_FAILURE, ADD_TXN_USER_ERROR, TCAM_POOL_STATE_ERROR, INVALID_USER_TOKEN</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>INVALID_TERMINAL_TYPE, TERMINAL_NOT_SIGNED_ON, PRESET_SECURITY_TERMINAL, SURROGATE_TERMINAL, NO_TERMINAL_WITH_TASK, SECURITY_INACTIVE, ESM_INACTIVE, ESM_TRANQUIL, UNKNOWN_ESM_RESPONSE</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

**DFHSNUS subroutine, SIGNON_ATTACH_HEADER function**

The SIGNON_ATTACH_HEADER function of the DFHSNUS subroutine causes a sign on for the userid received in an LU6.2 function management header type 5, also known as an attach header or an FMH5.

**Input parameters**

**TCTTE_PTR**

is the address of the TCTTE for which the FMH5 sign on is being performed.

**USERID**

is the userid obtained from the FMH5, if any.

**USERID_LENGTH**

is the length of the userid.

**PASSWORD**

is the password obtained from the FMH5, if any.

**PASSWORD_LENGTH**

is the length of the password.

**GROUPID**

is the group name obtained from the profile name in the FMH5, if any.

**GROUPID_LENGTH**

is the length of the group name.

**ENTRY_PORT_NAME**

is the optional name of the entry port (terminal) at which the userid was signed on in the terminal-owning region.

**ENTRY_PORT_TYPE**

is the optional terminal type associated with the port of entry. It can have either of these values:

TERMINAL | CONSOLE

**APPLID**

is the optional applid at which the userid was signed on in the terminal-owning region.

**ATTACHSEC_TYPE**

specifies whether the ATTACHSEC associated with the connection is LOCAL or not. It can have either of these values:

LOCAL | NON_LOCAL
ALREADY_VERIFIED
specifies whether the already-verified indicator (AV) is present in the FMH5. It can have either of these values:
YES|NO

PERSISTENT_SIGNON
specifies whether the persistent-sign on indicator (PV2) is present in the FMH5. It can have either of these values:
YES|NO

PERSISTENT_VERIFY
specifies whether the persistent-verification indicator (PV1) is present in the FMH5. It can have either of these values:
YES|NO

**Output parameters**

[SAF_RESPONSE]
is the optional 32-bit SAF response code to the call.

[SAF_REASON]
is the optional 32-bit SAF reason returned with SAF_RESPONSE.

[ESM_RESPONSE]
is the optional 32-bit ESM response code to the call.

[ESM_REASON]
is the optional 32-bit ESM reason returned with ESM_RESPONSE.

RESPONSE
is the subroutine’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP, USER_DOMAIN_FAILURE, MESSAGE_DOMAIN_FAILURE, CORRUPT_USER_TOKEN, ZCUT_FAILURE</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>TERMINAL_ALREADY_SIGNED_ON, INVALID_USERID, INVALID_PASSWORD, INVALID_GROUPID, USERID_NOT_IN_GROUP, PRESET_SECURITY_TERMINAL, USERID_REQUIRED, PROTOCOL_VIOLATION, PASSWORD_REQUIRED, UNKNOWN_ESM_RESPONSE, SECURITY_INACTIVE, ESM_INACTIVE, TERMINAL_NOTAUTH, LUIT_ENTRY_NOT_FOUND, APPLICATION_NOTAUTH, USERID_REVOKED, GROUP_ACCESS_REVOKED, SECLABEL_CHECK_FAILED, SIGNON_SURROGATE_ERROR, ESM_TRANQUIL</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

**DFHSNUS subroutine, SIGNOFF_ATTACH_HEADER function**
The SIGNOFF_ATTACH_HEADER function of the DFHSNUS subroutine is used to reverse the effect of a SIGNON_ATTACH_HEADER function when the transaction initiated by the FMH5 attach header terminates.

**Input parameters**

TCTTE_PTR
is the address of the TCTTE for which the FMH5 sign off is being performed.
Output parameters

[SAF_RESPONSE] is the optional 32-bit SAF response code to the call.

[SAF_REASON] is the optional 32-bit SAF reason returned with SAF_RESPONSE.

[ESM_RESPONSE] is the optional 32-bit ESM response code to the call.

[ESM_REASON] is the optional 32-bit ESM reason returned with ESM_RESPONSE.

RESPONSE is the subroutine’s response to the call. It can have any of these values:
OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP, ZCUT_FAILURE, CORRUPT_USER_TOKEN</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>INVALID_TERMINAL_TYPE, PRESET_SECURITY_TERMINAL, SURROGATE_TERMINAL, SECURITY_INACTIVE, ESM_INACTIVE, ESM_TRANQUIL, UNKNOWN_ESM_RESPONSE</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

Modules

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHSNAS</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>SIGNON_ATI_SESSION</td>
</tr>
<tr>
<td></td>
<td>SIGNOFF_ATI_SESSION</td>
</tr>
<tr>
<td>DFHSNPU</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>SIGNON_PRESET_USERID</td>
</tr>
<tr>
<td></td>
<td>SIGNOFF_PRESET_USERID</td>
</tr>
<tr>
<td>DFHSNDUF</td>
<td>SN domain offline dump formatting routine</td>
</tr>
<tr>
<td>DFHSNSG</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>SIGNOFF_SURROGATE</td>
</tr>
<tr>
<td>DFHSNSU</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>SIGNON_SESSION_USERID</td>
</tr>
<tr>
<td></td>
<td>SIGNOFF_SESSION_USERID</td>
</tr>
<tr>
<td>DFHSNTU</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>SIGNON_TERMINAL_USER</td>
</tr>
<tr>
<td></td>
<td>SIGNOFF_TERMINAL_USER</td>
</tr>
<tr>
<td>DFHSNUS</td>
<td>Acts as a router to the other signon modules, and handles the following</td>
</tr>
<tr>
<td></td>
<td>requests directly:</td>
</tr>
<tr>
<td></td>
<td>SIGNON_ATTACH_HEADER</td>
</tr>
<tr>
<td></td>
<td>SIGNOFF_ATTACH_HEADER</td>
</tr>
<tr>
<td>DFHSNTRI</td>
<td>Interprets SN domain trace entries</td>
</tr>
</tbody>
</table>
Sign-on Component

Exits

There are two global user exit points in DFHSNUS: XSNON and XSNOFF.

For further information, see the CICS Customization Guide.

Trace

The point IDs for the sign on component are of the form AP xxxx; the corresponding trace levels are AP 1 and Exc.

For more information about the trace points, see the CICS Trace Entries. For more information about using traces in problem determination, see the CICS Problem Determination Guide.
Chapter 77. Socket domain (SO)

The socket domain provides TCP/IP services to the CICS Web Support and CICS IIOP Support components. It includes a TCP/IP listener system task, the TCPIPSERVICE RDO resource to manage the listener and domain gates to operate on a TCP/IP connection.

Socket domain’s specific gates

Table 96 summarizes the socket domain’s specific gates. It shows the level-1 trace point IDs of the modules providing the functions for the gate, the functions provided by the gate, and whether or not the functions are available through the exit programming interface (XPI).

Table 96. Socket domain’s specific gates

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>XPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOCK</td>
<td>SO 0201</td>
<td>SEND</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>SO 0202</td>
<td>SEND_SSL_DATA</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RECEIVE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RECEIVE_SSL_DATA</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CLOSE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LISTEN</td>
<td>NO</td>
</tr>
<tr>
<td>SORD</td>
<td>SO 0301</td>
<td>REGISTER</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>SO 0302</td>
<td>DEREGISTER</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IMMCLOSE</td>
<td>NO</td>
</tr>
<tr>
<td>SOIS</td>
<td>SO 0401</td>
<td>SET_PARAMETERS</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>SO 0402</td>
<td>INITIALIZE_ENVIRONMENT</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INQUIRE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SET</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INQUIRE_STATISTICS</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VERIFY</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EXPORT_CERTIFICATE_DATA</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IMPORT_CERTIFICATE_DATA</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DELETE_CERTIFICATE_DATA</td>
<td>NO</td>
</tr>
<tr>
<td>SOAD</td>
<td>SO 0601</td>
<td>ADD_REPLACE_TCPIPSERVICE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>SO 0602</td>
<td>DELETE_TCPIPSERVICE</td>
<td>NO</td>
</tr>
<tr>
<td>SOTB</td>
<td>SO 0701</td>
<td>INQUIRE_TCPIPSERVICE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>SO 0702</td>
<td>START_BROWSE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GET_NEXT</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>END_BROWSE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SET_TCPIPSERVICE</td>
<td>NO</td>
</tr>
<tr>
<td>SOSE</td>
<td>SO 0801</td>
<td>INITIALIZE_SSL</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>SO 0802</td>
<td>SECURE_SOC_INIT</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SECURE_SOC_READ</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SECURE_SOC_WRITE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SECURE_SOC_CLOSE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SECURE_SOC_RESET</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TERMINATE_SSL</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EXPORT_CERTIFICATE_DATA</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IMPORT_CERTIFICATE_DATA</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DELETE_CERTIFICATE_DATA</td>
<td>NO</td>
</tr>
</tbody>
</table>
**SOCK gate, SEND function**

The SEND function sends a buffer of data to a connected TCP/IP client.

**Input parameters**

SEND_BUFFER  
is the buffer of data to be sent.

**Output parameters**

RESPONSE  
is the domain’s response to the call. It can have any of these values:

- OK
- EXCEPTION
- INVALID
- DISASTER
- KERNERROR
- PURGED

[REASON] is returned when RESPONSE is EXCEPTION, INVALID or DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>UNKNOWN_SESSION_TOKEN, INSUFFICIENT_STORAGE, IO_ERROR, CONNECTION_CLOSED</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP, LOCK_FAILURE, SOCKET_IN_USE</td>
</tr>
</tbody>
</table>

**SOCK gate, SEND_SSL_DATA function**

The SEND_SSL_DATA function is called to send data to a connected TCP/IP client if the connection is secured using SSL.

**Input parameters**

STE_PTR  
is a pointer to the STE control block of the session.

SEND_BUFFER  
is the buffer of data to be sent.

**Output parameters**

RESPONSE  
is the domain’s response to the call. It can have any of these values:

- OK
- EXCEPTION
- INVALID
- DISASTER
- KERNERROR
- PURGED

[REASON] is returned when RESPONSE is EXCEPTION, INVALID or DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>UNKNOWN_SESSION_TOKEN, INSUFFICIENT_STORAGE, IO_ERROR, CONNECTION_CLOSED</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP, LOCK_FAILURE, SOCKET_IN_USE</td>
</tr>
</tbody>
</table>

**SOCK gate, RECEIVE function**

The RECEIVE function receives a buffer of data from a TCP/IP connected client.
**Input parameters**

**RECEIVE_BUFFER**

is the buffer to receive the data into.

**[TIMEOUT]**

is an optional parameter. It can take two values:

DEFAULT|SOCKETCLOSE

If not specified or a value of SOCKETCLOSE is specified then the timeout is taken from the tcpipservice definition. If DEFAULT is specified then the timeout is 30 seconds.

**Output parameters**

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|
KERNERROR|PURGED

**[REASON]**

is returned when RESPONSE is EXCEPTION, INVALID or DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>UNKNOWN_SESSION_TOKEN, INSUFFICIENT_STORAGE, IO_ERROR,CONNECTION_CLOSED</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT,INVALID_FUNCTION</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND,LOOP,LOCK_FAILURE,SOCKET_IN_USE</td>
</tr>
</tbody>
</table>

**SOCK gate, RECEIVE_SSL_DATA function**

The RECEIVE_SSL_DATA function is called to receive data from a connected TCP/IP client if the connection is secured using SSL.

**Input parameters**

**STE_PTR**

is a pointer to the STE control block of the session.

**RECEIVE_BUFFER**

is the buffer to receive data into.

**Output parameters**

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|
KERNERROR|PURGED

**[REASON]**

is returned when RESPONSE is EXCEPTION, INVALID or DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>UNKNOWN_SESSION_TOKEN, INSUFFICIENT_STORAGE, IO_ERROR,CONNECTION_CLOSED</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT,INVALID_FUNCTION</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND,LOOP,LOCK_FAILURE,SOCKET_IN_USE</td>
</tr>
</tbody>
</table>
Socket Domain (SO)

**SOCK gate, CLOSE function**

The CLOSE function is called to close the socket connection to the TCP/IP client.

**Output parameters**

RESPONSE

is the domain's response to the call. It can have any of these values:

OK | EXCEPTION | INVALID | DISASTER | KERNERROR | PURGED

[REASON]

is returned when RESPONSE is EXCEPTION, INVALID or DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>UNKNOWN_SESSION_TOKEN, INSUFFICIENT_STORAGE, IO_ERROR, CONNECTION_CLOSED</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP, LOCK_FAILURE, SOCKET_IN_USE</td>
</tr>
</tbody>
</table>

**SOCK gate, LISTEN function**

The LISTEN function is the main routine for the SO domain listener task CSOL. When the listener task starts it branches into the LISTEN function of the SOCK gate. This allows the listener code to be written at the domain level rather than the task level.

**Output parameters**

RESPONSE

is the domain's response to the call. It can have any of these values:

OK | EXCEPTION | INVALID | DISASTER | KERNERROR | PURGED

[REASON]

is returned when RESPONSE is INVALID or DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP, LOCK_FAILURE</td>
</tr>
</tbody>
</table>

**SORD gate, REGISTER function**

The REGISTER function is called to open a tcpipservice. It registers all the parameters of the service with the listener task.

**Input parameters**

PORT_NUMBER

is the TCP/IP port number to listen for new connection on.

SERVICE_NAME

is the name of the tcpipservice.

TRANID

is the transaction ID that is to be attached when a new connection is made to the listening port.
SSL specifies whether or not connections to this service are to be secured using the Secure Sockets Layer protocols.

BACKLOG is the value of the backlog parameter passed to the TCP/IP listen function for this service. It specifies how many connection requests TCP/IP will queue for this service.

URM is the name of a User Replacable Module program that the handler transaction for this service will invoke during request processing.

TSQPREFIX is the prefix for TS queues that are created by the programs handling requests for this service.

IPADDRESS is the specific IP address that the listener will bind to for this service.

[CERTIFICATE_LABEL] is the name of a certificate within the keyfile that this service will use to authenticate itself to clients with, if the SSL protocol is used.

RECV_TIMEOUT specifies whether or not receives should timeout, and if so, after how long.

Output parameters

LISTEN_TOKEN is a token representing the opened tcpipservice. This is subsequently used to close the service.

RESPONSE is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | INVALID | DISASTER | KERNERROR | PURGED

[REASON] is returned when RESPONSE is INVALID or DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>UNKNOWN_LISTEN_TOKEN, INSUFFICIENT_STORAGE, NOT_PERMITTED_TO_BIND, TCP/IP_SERVICE_ERROR, TCP/IP_CLOSED, TCP/IP_INACTIVE, UNKNOWN_ADDRESS, PORT_IN_USE</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP, LOCK_FAILURE</td>
</tr>
</tbody>
</table>

SORD gate, DEREGISTER function

The DEREGISTER function is called to close a tcpipservice. The listener task closes the listening socket and no more connections to the port are permitted. Any tasks handling existing connections are allowed to end normally.

Input parameters

LISTEN_TOKEN is a token representing the opened tcpipservice.
Socket Domain (SO)

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK | EXCEPTION | INVALID | DISASTER |
KERNERROR | PURGED

[REASON]
is returned when RESPONSE is INVALID or DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>UNKNOWN_LISTEN_TOKEN, INSUFFICIENT_STORAGE, NOT_PERMITTED_TO_BIND,</td>
</tr>
<tr>
<td></td>
<td>TCPIP_SERVICE_ERROR, TCPIP_CLOSED, TCPIP_INACTIVE, UNKNOWN_ADDRESS,</td>
</tr>
<tr>
<td></td>
<td>PORT_IN_USE</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP, LOCK_FAILURE</td>
</tr>
</tbody>
</table>

SORD gate, IMMCLOSE function

The IMMCLOSE function is called to immediately close a tcpipservice. The listener task closes the listening socket and no more connections to the port are permitted. All existing connections are closed and any tasks handling them are abended.

Input parameters

LISTEN_TOKEN
is a token representing the opened tcpipservice.

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:

OK | EXCEPTION | INVALID | DISASTER |
KERNERROR | PURGED

[REASON]
is returned when RESPONSE is INVALID or DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>UNKNOWN_LISTEN_TOKEN, INSUFFICIENT_STORAGE, NOT_PERMITTED_TO_BIND,</td>
</tr>
<tr>
<td></td>
<td>TCPIP_SERVICE_ERROR, TCPIP_CLOSED, TCPIP_INACTIVE, UNKNOWN_ADDRESS,</td>
</tr>
<tr>
<td></td>
<td>PORT_IN_USE</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP, LOCK_FAILURE</td>
</tr>
</tbody>
</table>

SOIS gate, SET_PARAMETERS function

The SET_PARAMETERS function is called during CICS initialisation when the SIT is processed. It sets the startup parameters for the SO domain.

Input parameters

TCPIP
is a YES or NO value indicating if the SO domain is to initialise in this CICS region.
SSLDELAY
is a the SSL timeout value.

SSLTCBS
specifies the number of S8 TCBs to be attached for SSL use.

ENCRYPTION
specifies the type of encryption that will be used by the system. The value can be NORMAL, STRONG or WEAK.

KEYFILE
specifies the name of the HFS keyring file that contains the keypairs and certificate data.

QUALIFIER
is actually the password that was used to secure the keyring file upon creation.

Output parameters
RESPONSE
is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|INVALID|DISASTER|
KERNERROR|PURGED

[REASON]
is returned when RESPONSE is INVALID or DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT,INVALID_FUNCTION</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND,LOOP,LOCK_FAILURE</td>
</tr>
</tbody>
</table>

SOIS gate, INITIALIZE_ENVIRONMENT function
The INITIALIZE_ENVIRONMENT function is called during SO domain startup to create and initialize the CEEPI LE pre-initialized environment for invocation of C functions.

Output parameters
RESPONSE
is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|INVALID|DISASTER|
KERNERROR|PURGED

[REASON]
is returned when RESPONSE is INVALID or DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT,INVALID_FUNCTION</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND,LOOP,LOCK_FAILURE</td>
</tr>
</tbody>
</table>

SOIS gate, INQUIRE function
The INQUIRE function is called by tasks that have been attached by the listener in response to a new TCP/IP connection. It provides TCP/IP and socket information about connection and the connected client.
Socket Domain (SO)

**Input parameters**

[LISTEN_TOKEN]

is a token representing an opened tcpipservice.

[CLIENT_HOSTNAME]

is a buffer in which the full hostname of the client is returned to the caller.

[SERVER_HOSTNAME]

is a buffer in which the full hostname of the CICS region is returned to the caller.

[GENERIC_HOSTNAME]

is a buffer in which the full generic hostname of the CICS region, as known to the DNS in a connection optimization environment, is returned to the caller.

**Output parameters**

[CLIENT_IP_ADDRESS]

is the text representation of the IP address of the client.

[CLIENT_BIN_IP_ADDRESS]

is the 32 bit binary IP address of the client.

[SERVER_IP_ADDRESS]

is the text representation of the IP address of the CICS region.

[SERVER_BIN_IP_ADDRESS]

is the 32 bit binary IP address of the CICS region.

[ListENER_PORT]

is the port number that the connection was received on.

[CLIENT_IP_ADDRESS_LEN]

is the length of the text representation of the client IP address.

[SERVER_IP_ADDRESS_LEN]

is the length of the text representation of the server IP address.

[CERTIFICATE_USERID]

is the userid associated with the certificate that was used to authenticate a client if this is an SSL connection.

[SSLTYPE]

returns whether or not SSL is being used to secure this connection.

[URM_NAME]

is the name of the URM program specified on the tcpipservice definition for this connection.

[TSQ_PREFIX]

is the TS queue prefix specified on the tcpipservice definition for this connection.

[ListENER_STATUS]

is the current status of the SO domain listener task.

[CONNECTIONS]

is either the number of connections for the service represented by the supplied LISTEN_TOKEN, or the total number of TCP/IP connections to all of the currently active services.
Socket Domain (SO)

[TCPIPSERVICE_NAME]
is the name of the service that attached the task, or the name associated with the supplied LISTEN_TOKEN.

[GROUP_NAME]
is the name of the dynamic DNS group that is registered with the MVS Work Load Manager for this service.

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK, EXCEPTION, INVALID, DISASTER, KERNERROR, PURGED

[REASON]
is returned when RESPONSE is INVALID or DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP, LOCK_FAILURE</td>
</tr>
</tbody>
</table>

SOIS gate, SET function

The SET function is called to open, close or immediately close the SO domain within a region. This is called in response to a SET TCPIP operator or SPI command.

Input parameters

[TCPIP_STATUS]
is either OPEN, CLOSED or IMMCLOSE.

Output parameters

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK, EXCEPTION, INVALID, DISASTER, KERNERROR, PURGED

[REASON]
is returned when RESPONSE is INVALID or DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP, LOCK_FAILURE</td>
</tr>
</tbody>
</table>

SOIS gate, INQUIRE_STATISTICS function

The INQUIRE_STATISTICS function returns gathered statistics about an open tcpipservice.

Input parameters

LISTEN_TOKEN
is the token representing the open tcpipservice, returned from the SORD REGISTER function.
Socket Domain (SO)

**RESET**

is a value indicating if the statistics should be reset.

**Output parameters**

**[ATTACH_COUNT]**

is the total number of tasks that have been attached to handle incoming connections.

**[PEAK_CONNECTIONS]**

is the high water mark for connections since that last reset.

**[SEND_COUNT]**

is the number of times TCP/IP send has been called.

**[SEND_BYTES]**

is the number of bytes that have been sent to TCP/IP.

**[RECV_COUNT]**

is the number of times TCP/IP receive has been called.

**[RECV_BYTES]**

is the number of bytes received from TCP/IP.

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|
KERNERROR|PURGED

**[REASON]**

is returned when RESPONSE is INVALID or DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT,INVALID_FUNCTION</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND,LOOP,LOCK_FAILURE</td>
</tr>
</tbody>
</table>

**SOIS gate, VERIFY function**

The VERIFY function checks an IP address string in the form nnn.nnn.nnn.nnn for validity and returns the binary form.

**Input parameters**

**SERVER_IP_ADDRESS**

is a char string in the form nnn.nnn.nnn.nnn representing an IP address.

**Output parameters**

**SERVER_BIN_IP_ADDRESS**

is the 32 bit binary number of the IP address.

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|
KERNERROR|PURGED

**[REASON]**

is returned when RESPONSE is INVALID or DISASTER. Possible values are:
### SOIS gate, EXPORT_CERTIFICATE_DATA function

The EXPORT_CERTIFICATE_DATA function saves a certificate in the sockets repository.

**Input parameters**

- **CERTIFICATE_INFORMATION**
  - A block representing the certificate.

**Output parameters**

- **REPOSITORY_TOKEN**
  - A token that represents the saved certificate data.

- **RESPONSE**
  - The domain's response to the call. It can have any of these values:
    - OK
    - EXCEPTION
    - INVALID
    - DISASTER
    - KERNERROR
    - PURGED

**[REASON]**

- Is returned when RESPONSE is INVALID or DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT,INVALID_FUNCTION</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND,LOOP,LOCK_FAILURE</td>
</tr>
</tbody>
</table>

### SOIS gate, IMPORT_CERTIFICATE_DATA function

The IMPORT_CERTIFICATE_DATA function imports certificate data from the sockets repository.

**Input parameters**

- **[REPOSITORY_TOKEN]**
  - A token representing a certificate exported to the repository.

- **CERTIFICATE_INFORMATION**
  - The block representing the certificate. The data is returned by the function.

**Output parameters**

- **CERTIFICATE_USERID**
  - The userid associated with the certificate.

- **RESPONSE**
  - The domain's response to the call. It can have any of these values:
    - OK
    - EXCEPTION
    - INVALID
    - DISASTER
    - KERNERROR
    - PURGED

**[REASON]**

- Is returned when RESPONSE is INVALID or DISASTER. Possible values are:
**Socket Domain (SO)**

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT,INVALID_FUNCTION</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP, LOCK_FAILURE</td>
</tr>
</tbody>
</table>

**SOIS gate, DELETE_CERTIFICATE_DATA function**

The DELETE_CERTIFICATE_DATA deletes certificate data from the sockets repository.

**Input parameters**

REPOSITORY_TOKEN
- a token representing a certificate exported to the repository.

**Output parameters**

RESPONSE
- is the domain’s response to the call. It can have any of these values:
  - OK
  - EXCEPTION
  - INVALID
  - DISASTER
  - KERNERROR
  - PURGED

[REASON]
- is returned when RESPONSE is INVALID or DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT,INVALID_FUNCTION</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP, LOCK_FAILURE</td>
</tr>
</tbody>
</table>

**SOAD gate, ADD_REPLACE_TCPIPSERVICE function**

The ADD_REPLACE_TCPIPSERVICE function is called at RDO time to install a tcpipservice definition. If the status is OPEN then the service is also opened using the SORD REGISTER function. A catalog entry is written to record the installed resource.

**Input parameters**

TCPIPSERVICE_NAME
- is the name of the tcpipservice.

URM_NAME
- is the name of the URM program.

STATUS
- is either OPEN or CLOSED.

SSL
- is either YES, NO or CLIENTAUTH.

TRANSACTION
- is the tranid of the transaction to attach for each connection to this service.

BACKLOG
- is the TCP/IP listen backlog parameter.

PORTNUMBER
- is the port number to listen on.
CERTIFICATE_LABEL is the name of the certificate from the keyfile to use to authenticate this service.

IPADDRESS is the IP address to bind this service to.

TSQPREFIX is the TS queue prefix to use for this service.

SOCKETCLOSE is the value of receive timeout for this service.

**Output parameters**

RESPONSE is the domain's response to the call. It can have any of these values:

- OK
- EXCEPTION
- INVALID
- DISASTER
- KERNERROR
- PURGED

[REASON] is returned when RESPONSE is INVALID or DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT,INVALID_FUNCTION</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND,LOOP,LOCK_FAILURE</td>
</tr>
</tbody>
</table>

**SOAD gate, DELETE_TCPIPSERVICE function**

The DELETE_TCPIPSERVICE function is called at RDO time to remove an installed tcpipservice definition. If the status is OPEN then the tcpipservice is not removed. The catalog entry is removed for the discarded resource.

**Input parameters**

TCPIPSERVICE_NAME is the name of the tcpipservice to remove.

**Output parameters**

RESPONSE is the domain's response to the call. It can have any of these values:

- OK
- EXCEPTION
- INVALID
- DISASTER
- KERNERROR
- PURGED

[REASON] is returned when RESPONSE is INVALID or DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT,INVALID_FUNCTION</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND,LOOP,LOCK_FAILURE</td>
</tr>
</tbody>
</table>

**SOTB gate, INQUIRE_TCPIPSERVICE function**

The INQUIRE_TCPIPSERVICE function is called by CEMT and the SPI for an INQUIRE TCPIPSERVICE function. It returns information about an installed tcpipservice.
Socket Domain (SO)

**Input parameters**

**TCPIPSERVICE_NAME**

is the name of the tcpipservice to inquire upon.

**Output parameters**

**[TRANSID]**

is the transaction ID associated with the service.

**[URM]**

is the URM name associated with the service.

**[PORT]**

is the port number associated with the service.

**[BACKLOG]**

is the backlog value associated with the service.

**[CONNECTIONS]**

is the current number of connections associated with the service.

**[IPADDRESS]**

is the IP address that the service is bound to.

**[TSQPREFIX]**

is the TS queue prefix associated with the service.

**[SOCKETCLOSE]**

is the receive timeout value associated with the service.

**RESPONSE**

is the domain's response to the call. It can have any of these values:

OK | EXCEPTION | INVALID | DISASTER | KERNERROR | PURGED

**[REASON]**

is returned when RESPONSE is INVALID or DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT,INVALID_FUNCTION</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND,LOOP,LOCK_FAILURE</td>
</tr>
</tbody>
</table>

**SOTB gate, START_BROWSE function**

The START_BROWSE function is called by CEMT and the SPI for an browsing tcpipservices.

**Output parameters**

**BROWSE_TOKEN**

is a token representing the browse.

**RESPONSE**

is the domain's response to the call. It can have any of these values:

OK | EXCEPTION | INVALID | DISASTER | KERNERROR | PURGED

**[REASON]**

is returned when RESPONSE is INVALID or DISASTER. Possible values are:
**SOTB gate, GET_NEXT function**

The GET_NEXT function is called by CEMT and the SPI for browsing tcpipserivces. It returns information about an installed tcpipserivce.

**Input parameters**

**BROWSE_TOKEN**

is a token representing the browse.

**Output parameters**

**TCPIPSERVICE_NAME**

is the name of the tcpipserivce.

**[TRANSID]**

is the transaction ID associated with the service.

**[URM]**

is the URM name associated with the service.

**[PORT]**

is the port number associated with the service.

**[BACKLOG]**

is the backlog value associated with the service.

**[CONNECTIONS]**

is the current number of connections associated with the service.

**[IPADDRESS]**

is the IP address that the service is bound to.

**[TSQPREFIX]**

is the TS queue prefix associated with the service.

**[SOCKETCLOSE]**

is the receive timeout value associated with the service.

**[STATUS]**

is the current status of the service:

- OPEN
- OPENING
- CLOSED
- CLOSING
- IMM CLOSING

**[SSL]**

is the SSL setting for the service:

- YES
- NO
- CLIAUTH

**[CERTIFICATE_LABEL]**

is the certificate label associated with the service.

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

- OK
- EXCEPTION
- INVALID
- DISASTER
- KERNERROR
- PURGED

**[REASON]**

is returned when RESPONSE is INVALID or DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP, LOCK_FAILURE</td>
</tr>
</tbody>
</table>
**Socket Domain (SO)**

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT,INVALID_FUNCTION</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND,LOOP,LOCK_FAILURE</td>
</tr>
</tbody>
</table>

**SOTB gate, END_BROWSE function**

The END_BROWSE function is called by CEMT and the SPI to end browsing tcpipservices.

**Input parameters**

**BROWSE_TOKEN**

is a token representing the browse.

**Output parameters**

**RESPONSE**

is the domain's response to the call. It can have any of these values:

- OK
- EXCEPTION
- INVALID
- DISASTER
- KERNERROR
- PURGED

**[REASON]**

is returned when RESPONSE is INVALID or DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT,INVALID_FUNCTION</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND,LOOP,LOCK_FAILURE</td>
</tr>
</tbody>
</table>

**SOTB gate, SET_TCPIPSERVICE function**

The SET_TCPIPSERVICE function is called by CEMT and the SPI to set tcpipservice parameters.

**Input parameters**

**TCPIPSERVICE_NAME**

is the name of the service to set.

**[STATUS]**

is the status to set for the service:

- OPEN
- CLOSED
- IMMCLOSED

**[URM]**

is the name of a new URM program to set.

**[BACKLOG]**

is the value of the new backlog parameter. This can only be set if the service is closed.

**Output parameters**

**RESPONSE**

is the domain's response to the call. It can have any of these values:

- OK
- EXCEPTION
- INVALID
- DISASTER
- KERNERROR
- PURGED

**[REASON]**

is returned when RESPONSE is INVALID or DISASTER. Possible values
### RESPONSE Possible REASON values

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP, LOCK_FAILURE</td>
</tr>
</tbody>
</table>

### SOSE gate, INITIALIZE_SSL function

The `INITIALIZE_SSL` function is called at SO domain initialization. It creates the environment necessary to perform Secure Sockets Layer communication.

**Output parameters**

- **GSK_RETURN_CODE**
  
  is the return code from the System SSL component of OS/390 that CICS uses to perform SSL communications.

- **RESPONSE**
  
  is the domain's response to the call. It can have any of these values:

  - OK
  - EXCEPTION
  - INVALID
  - DISASTER
  - KERNERROR
  - PURGED

- **[REASON]**
  
  is returned when RESPONSE is INVALID or DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP, CEEPIPI_ERROR, GSK_ERROR</td>
</tr>
</tbody>
</table>

### SOSE gate, SECURE_SOC_INIT function

The `SECURE_SOC_INIT` function is called when a new connection is established with the service and SSL is enabled. This function performs the SSL handshake to establish the security.

**Output parameters**

- **GSK_RETURN_CODE**
  
  is the return code from the System SSL component of OS/390 that CICS uses to perform SSL communications.

- **CERTIFICATE**
  
  is the certificate used by the client to authenticate itself.

- **CERTIFICATE_USERID**
  
  is the userid associated with the client certificate.

- **CIPHERS_SELECTED**
  
  represents the encryption cyphers that have been selected in negotiation with the client and server.

- **RESPONSE**
  
  is the domain's response to the call. It can have any of these values:

  - OK
  - EXCEPTION
  - INVALID
  - DISASTER
  - KERNERROR
  - PURGED

- **[REASON]**
  
  is returned when RESPONSE is INVALID or DISASTER. Possible values...
Socket Domain (SO)

are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>GSK_INACTIVE, INSUFFICIENT_THREADS, GETMAIN_FAILED, REPOSITORY_ERROR, CONNECTION_CLOSED, CLIENT_ERROR</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP, CEEPIPI_ERROR, GSK_ERROR</td>
</tr>
</tbody>
</table>

SOSE gate, SECURE_SOC_READ function

The SECURE_SOC_READ function is called to read data on a secure connection.

**Input parameters**

RECEIVE_BUFFER

is the buffer to hold the received data.

**Output parameters**

GSK_RETURN_CODE

is the return code from the System SSL component of OS/390 that CICS uses to perform SSL communications.

RESPONSE

is the domain's response to the call. It can have any of these values:

OK | EXCEPTION | INVALID | DISASTER | KERNERROR | PURGED

[REASON]

is returned when RESPONSE is INVALID or DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP, CEEPIPI_ERROR, GSK_ERROR, CONNECTION_CLOSED, HANDSHAKE_ERROR</td>
</tr>
</tbody>
</table>

SOSE gate, SECURE_SOC_WRITE function

The SECURE_SOC_WRITE function is called to send data on a secure connection.

**Input parameters**

SEND_BUFFER

is the buffer to holding the data to send.

**Output parameters**

GSK_RETURN_CODE

is the return code from the System SSL component of OS/390 that CICS uses to perform SSL communications.

RESPONSE

is the domain's response to the call. It can have any of these values:

OK | EXCEPTION | INVALID | DISASTER | KERNERROR | PURGED

[REASON]

is returned when RESPONSE is INVALID or DISASTER. Possible values are:
The `SECURE_SOC_CLOSE` function is called to close a secure connection.

**Output parameters**

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP, CEEPIPI_ERROR, GSK_ERROR, CONNECTION_CLOSED, HANDSHAKE_ERROR</td>
</tr>
</tbody>
</table>

**SOSE gate, `SECURE_SOC_RESET` function**

The `SECURE_SOC_RESET` function is called to reset a secure connection.

**Output parameters**

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP, CEEPIPI_ERROR, GSK_ERROR</td>
</tr>
</tbody>
</table>

**SOSE gate, `TERMINATE_SSL` function**

The `TERMINATE_SSL` function is to terminate all SSL operation in a region.

**Output parameters**

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP, CEEPIPI_ERROR, GSK_ERROR</td>
</tr>
</tbody>
</table>
**SOSE gate, EXPORT_CERTIFICATE_DATA function**

The `EXPORT_CERTIFICATE_DATA` function saves a certificate in the sockets repository.

**Input parameters**

- **CERTIFICATE_INFORMATION**
  is a block representing the certificate.

**Output parameters**

- **REPOSITORY_TOKEN**
  is a token that represents the saves certificate data.

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

- OK
- EXCEPTION
- INVALID
- DISASTER
- KERNERROR
- PURGED

**[REASON]**

is returned when RESPONSE is INVALID or DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP, CEEPIPI_ERROR</td>
</tr>
</tbody>
</table>

---

**SOSE gate, IMPORT_CERTIFICATE_DATA function**

The `IMPORT_CERTIFICATE_DATA` imports certificate data from the sockets repository.

**Input parameters**

- **[REPOSITORY_TOKEN]**
  a token representing a certificate exported to the repository.

- **CERTIFICATE_INFORMATION**
  is the block representing the certificate. The data is returned by the function.

**Output parameters**

- **CERTIFICATE_USERID**
  is the userid associated with the certificate.

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

- OK
- EXCEPTION
- INVALID
- DISASTER
- KERNERROR
- PURGED

**[REASON]**

is returned when RESPONSE is INVALID or DISASTER. Possible values are:
The DELETE_CERTIFICATE_DATA function deletes certificate data from the sockets repository.

**Input parameters**

REPOSITORY_TOKEN

a token representing a certificate exported to the repository.

**Output parameters**

RESPONSE

is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON]

is returned when RESPONSE is INVALID or DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT,INVALID_FUNCTION</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND,LOOP,LOCK_FAILURE</td>
</tr>
</tbody>
</table>

**Socket domain’s generic gates**

Table 97 summarizes the socket domain’s generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>SODM</td>
<td>SO 0101</td>
<td>INITIALIZE_DOMAIN</td>
<td>DMDM</td>
</tr>
<tr>
<td>SO 0102</td>
<td></td>
<td>QUIESCE_DOMAIN</td>
<td></td>
</tr>
<tr>
<td>SO 0102</td>
<td></td>
<td>TERMINATE_DOMAIN</td>
<td></td>
</tr>
<tr>
<td>SOXM</td>
<td>EM 0401</td>
<td>INQUIRE_DATA_LENGTH</td>
<td>XMXM</td>
</tr>
<tr>
<td></td>
<td>EM 0402</td>
<td>GET_DATA</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>DESTROY_TOKEN</td>
<td></td>
</tr>
</tbody>
</table>

For descriptions of these functions and their input and output parameters, refer to the §§ dealing with the corresponding generic formats:

| Functions and parameters | Format DMDM—Figure 116 on page 1333 | Format XMXM—Figure 89 on page 1153 |
In initialization, quiesce, and termination processing, the socket domain performs only internal routines.

## Modules

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHSODM</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>INITIALIZE_DOMAIN</td>
</tr>
<tr>
<td></td>
<td>QUIESCE_DOMAIN</td>
</tr>
<tr>
<td></td>
<td>TERMINATE_DOMAIN</td>
</tr>
<tr>
<td>DFHSOCK</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>LISTEN</td>
</tr>
<tr>
<td></td>
<td>SEND</td>
</tr>
<tr>
<td></td>
<td>RECEIVE</td>
</tr>
<tr>
<td></td>
<td>CLOSE</td>
</tr>
<tr>
<td></td>
<td>SEND_SSL_DATA</td>
</tr>
<tr>
<td></td>
<td>RECV_SSL_DATA</td>
</tr>
<tr>
<td>DFHSORD</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>REGISTER</td>
</tr>
<tr>
<td></td>
<td>DEREISTER</td>
</tr>
<tr>
<td></td>
<td>IMMCLOSE</td>
</tr>
<tr>
<td>DFHSOIS</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>INITIALIZE_ENVIRONMENT</td>
</tr>
<tr>
<td></td>
<td>INQUIRE</td>
</tr>
<tr>
<td></td>
<td>SET_PARAMETERS</td>
</tr>
<tr>
<td></td>
<td>INQUIRE_STATISTICS</td>
</tr>
<tr>
<td></td>
<td>VERIFY</td>
</tr>
<tr>
<td></td>
<td>EXPORT_CERTIFICATE_DATA</td>
</tr>
<tr>
<td></td>
<td>IMPORT_CERTIFICATE_DATA</td>
</tr>
<tr>
<td></td>
<td>DELETE_CERTIFICATE_DATA</td>
</tr>
<tr>
<td>DFHSOAD</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>ADD_REPLACE_TCPIPSERVICE</td>
</tr>
<tr>
<td></td>
<td>DELETE_TCPIPSERVICE</td>
</tr>
<tr>
<td>DFHSOTB</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>INQUIRE_TCPIPSERVICE</td>
</tr>
<tr>
<td></td>
<td>START_BROWSE</td>
</tr>
<tr>
<td></td>
<td>GET_NEXT</td>
</tr>
<tr>
<td></td>
<td>END_BROWSE</td>
</tr>
<tr>
<td></td>
<td>SET_TCPIPSERVICE</td>
</tr>
<tr>
<td>DFHSOSE</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>INITIALIZE_SSL</td>
</tr>
<tr>
<td></td>
<td>SECURE_SOC_INIT</td>
</tr>
<tr>
<td></td>
<td>SECURE_SOC_READ</td>
</tr>
<tr>
<td></td>
<td>SECURE_SOC_WRITE</td>
</tr>
<tr>
<td></td>
<td>SECURE_SOC_CLOSE</td>
</tr>
<tr>
<td></td>
<td>SECURE_SOC_RESET</td>
</tr>
<tr>
<td></td>
<td>TERMINATE_SSL</td>
</tr>
<tr>
<td></td>
<td>EXPORT_CERTIFICATE_DATA</td>
</tr>
<tr>
<td></td>
<td>IMPORT_CERTIFICATE_DATA</td>
</tr>
<tr>
<td></td>
<td>DELETE_CERTIFICATE_DATA</td>
</tr>
<tr>
<td>DFHSODUF</td>
<td>Formats the SO domain control blocks</td>
</tr>
<tr>
<td>DFHSOTRI</td>
<td>Interprets SO domain trace entries</td>
</tr>
</tbody>
</table>
Exits

No global user exit points are provided in this domain.
Chapter 78. Statistics domain (ST)

The statistics domain controls the collection of resource statistics for a CICS system (the monitoring domain collects task statistics). The statistics domain collects data at user-specified intervals, at system quiesce or logical end of day, and when requested by the user, and writes it to the statistics data sets in SMF format. This can subsequently be used by the statistics offline utility to produce formatted reports.

Statistics domain’s specific gate

Table 98 summarizes the statistics domain’s specific gate. It shows the level-1 trace point IDs of the modules providing the functions for the gate, the functions provided by the gate, and whether or not the functions are available through the exit programming interface (XPI).

Table 98. Statistics domain’s specific gate

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>XPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>STST</td>
<td>ST 0003</td>
<td>INQ_STATISTICS_OPTIONS</td>
<td>NO</td>
</tr>
<tr>
<td>STST</td>
<td>ST 0004</td>
<td>SET_STATISTICS_OPTIONS</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>REQUEST_STATISTICS</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RECORD_STATISTICS</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>STATISTICS_COLLECTION</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DISABLE_STATISTICS</td>
<td>NO</td>
</tr>
</tbody>
</table>

STST gate, INQ_STATISTICS_OPTIONS function

The INQ_STATISTICS_OPTIONS function of the STST gate is used to return information associated with the statistics domain options.

Input parameters
None.

Output parameters

COLLECT
indicates whether interval statistics are being collected (and their counts reset). It can have either of these values:
YES | NO

INTERVAL
is the interval at which statistics are being collected if COLLECT is YES.

EOD_TIME_OF_DAY
is the time of day at which end-of-day statistics are collected.

NEXT_COLLECTION_TIME
is the time of the next collection of statistics. If COLLECT is YES, it is the earlier of the next interval collection time and the logical end-of-day time; if COLLECT is NO, it is the logical end-of-day time.

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK | INVALID | KERNERROR | DISASTER
Statistics domain (ST)

[REASON] is returned when RESPONSE is DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
</tbody>
</table>

**STST gate, SET_STATISTICS_OPTIONS function**

The SET_STATISTICS_OPTIONS function of the STST gate is used to set statistics options.

**Input parameters**

[COLLECT] indicates whether interval statistics are to be collected (and their counts reset). It can have either of these values:

YES\|NO

[INTERVAL] is the interval at which statistics are to be collected if COLLECT is YES.

[EOD_TIME_OF_DAY] is the time of day at which end-of-day statistics are to be collected.

[COLLECT_UPDATE_ACTION] is the action to be taken when changing the COLLECT option value from NO to YES, or from YES to NO. It can have any one of these values:

NOACTION\|RESETNOW\|RECORDNOW\|RECORD_RESETNOW

**Output parameters**

RESPONSE is the domain’s response to the call. It can have any of these values:

OK\|INVALID\|KERNERROR\|DISASTER

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>COLL_ACTION_NO_UPDATE</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_COLLECT, INVALID_INTERVAL, INVALID_EOD_TIME_OF_DAY,</td>
</tr>
<tr>
<td></td>
<td>INV_COLL_UPDATE_ACTION</td>
</tr>
</tbody>
</table>

**STST gate, REQUEST_STATISTICS function**

The REQUEST_STATISTICS function of the STST gate is used to request a collection of statistics.

**Input parameters**

[DOMAIN_TOKEN] identifies the domain from which the statistics are to be collected.

[RESOURCE_TYPE] indicates the resource in the AP domain on which statistics are to be collected.
REQUEST_TOKEN
uniquely identifies the collection of statistics requested by the caller.

RESET
indicates whether certain statistics fields are to be reset.

Output parameters
RESPONSE
is the domain’s response to the call. It can have any of these values: OK|INVALID|EXCEPTION|PURGED|KERNERROR|DISASTER

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>TYPE_NOT_FOUND, NOT_AVAILABLE, INCOMPLETE_DATA</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_RESET</td>
</tr>
</tbody>
</table>

STST gate, RECORD_STATISTICS function
The RECORD_STATISTICS function of the STST gate is used to record statistics.

Input parameters
STATISTICS_DATA
specifies the address and length of data requested.

STATISTICS_TYPE
indicates the type of statistics collection, either a normal collection or unsolicited. It can have either of these values: COLLECTION|USS

Output parameters
RESPONSE
is the domain’s response to the call. It can have any of these values: OK|INVALID|KERNERROR|DISASTER

[REASON] is returned when RESPONSE is DISASTER or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_DATA_FORMAT</td>
</tr>
</tbody>
</table>

STST gate, STATISTICS_COLLECTION function
The STATISTICS_COLLECTION function of the STST gate is used to initiate a collection of statistics.

Input parameters
RESET
indicates whether certain statistics fields are to be reset.
Statistics domain (ST)

DATA indicates whether the domain being called is requested to return its statistics to the caller.

END_OF_DAY indicates whether all statistics fields are to be reset.

COLLECTION_TYPE indicates whether this is an interval collection or end-of-day collection of statistics. It can have either of these values:

INT|EOD

[SYSTEM_TERMINATING] indicates whether this is the last collection for the CICS run. It can have either of these values:

YES|NO

YES is used for the end-of-day collection that is taken when CICS is shut down.

Output parameters

RESPONSE is the domain’s response to the call. It can have any of these values:

OK|INVALID|KERNERROR|DISASTER

[REASON] is returned when RESPONSE is DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
</tbody>
</table>

STST gate, DISABLE_STATISTICS function

The DISABLE_STATISTICS function of the STST gate is used to disable statistics interval collections.

Input parameters

None.

Output parameters

RESPONSE is the domain’s response to the call. It can have any of these values:

OK|INVALID|KERNERROR|DISASTER

[REASON] is returned when RESPONSE is DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
</tbody>
</table>
Statistics domain’s generic gates

Table 99 summarizes the statistics domain’s generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 99. Statistics domain’s generic gates

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMDM</td>
<td>ST 0001</td>
<td>INITIALIZE_DOMAIN</td>
<td>DMDM</td>
</tr>
<tr>
<td></td>
<td>ST 0002</td>
<td>QUIESCE_DOMAIN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TERMINATE_DOMAIN</td>
<td></td>
</tr>
<tr>
<td>TISR</td>
<td>ST 0005</td>
<td>NOTIFY</td>
<td>TISR</td>
</tr>
<tr>
<td></td>
<td>ST 0006</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For descriptions of these functions and their input and output parameters, refer to the sections dealing with the corresponding generic formats:

Functions and parameters

Format DMDM—Domain manager domain’s generic formats" on page 377
Format TISR—Timer domain’s generic format” on page 1228

In initialization processing, the statistics domain sets the initial statistics options:
• Collecting interval
• Logical end of day
• Collecting status.

For a cold start, the collecting interval defaults to 3 hours, the logical end of day defaults to midnight, and the collecting status defaults to ON; for any other type of start, the information comes from the global catalog.

In quiesce processing, the statistics domain collects and records statistics from all other domains.

In termination processing, the statistics domain collects and records end-of-day statistics.

Statistics domain’s generic format

Table 100 summarizes the generic format owned by the statistics domain and shows the functions performed on the calls.

Table 100. Generic format owned by statistics domain

<table>
<thead>
<tr>
<th>Format</th>
<th>Calling module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>STST</td>
<td>DFHSTST</td>
<td>COLLECT_STATISTICS</td>
</tr>
<tr>
<td></td>
<td>DFHEIQMS</td>
<td>COLLECT_RESOURCE_STATS</td>
</tr>
</tbody>
</table>

In the descriptions of the format that follows, the “input” parameters are input not to statistics domain, but to the domain being called by the statistics domain. Similarly, the “output” parameters are output by the domain that was called by the statistics domain, in response to the call.
STST format, COLLECT_STATISTICS function

The COLLECT_STATISTICS function of the STST format is used by the statistics domain to ask a domain to collect its statistics.

**Input parameters**

**DATA** indicates whether the domain being called is requested to return its statistics to the caller. It can have either of these values:
- YES
- NO

**END_OF_DAY**
indicates whether all statistics fields are to be reset. It can have either of these values:
- YES
- NO

**RESET**
indicates whether certain statistics fields are to be reset. It can have either of these values:
- YES
- NO

**RESET_TIME**
is the time of day to be used as the time at which the statistics fields were last reset.

**[RESOURCE_TYPE]**
indicates the resource in the AP domain on which statistics are to be collected.

**Output parameters**

**RESPONSE**
is the domain’s response to the call. It can have any of these values:
- OK
- EXCEPTION
- INVALID
- PURGED
- KERNERROR
- DISASTER

**[REASON]**
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>TYPE_NOT_FOUND, NOT_AVAILABLE, INCOMPLETE_DATA</td>
</tr>
</tbody>
</table>

STST format, COLLECT_RESOURCE_STATS function

The COLLECT_RESOURCE_STATS function of the STST format is used by the EXEC API to ask a domain to collect its monitoring data collection information.

**Input parameters**

**[RESOURCE_TYPE]**
is the type of resource on which statistics are required.

**[RESOURCE_ID]**
specifies the address and length of the resource identifier.

**[RESOURCE_ID_2]**
specifies the address and length of the resource identifier.

**[RESOURCE_ID_3]**
specifies the address and length of the resource identifier.
[LONGRESOURCE_IDM] specifies the address and length of the resource identifier.

[RESID_TOKEN] a token representing the resource id required.

RESOURCE_STATISTICS_DATA specifies the address and length of the area into which the requested statistics are to be placed.

Output parameters

[LAST_RESET_TIME] indicates the time at which the statistics fields were last reset.

RESPONSE is the domain’s response to the call. It can have any of these values: OK|EXCEPTION|INVALID|PURGED|KERNERROR|DISASTER

[REASON] is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>TYPE_NOT_FOUND, ID_NOT_FOUND, NOT_AVAILABLE</td>
</tr>
</tbody>
</table>

Modules

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHSTDM</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>INITIALIZE_DOMAIN</td>
</tr>
<tr>
<td></td>
<td>QUIESCE_DOMAIN</td>
</tr>
<tr>
<td></td>
<td>TERMINATE_DOMAIN</td>
</tr>
<tr>
<td>DFHSTDF</td>
<td>Formats the ST domain control blocks in a CICS system dump</td>
</tr>
<tr>
<td>DFHSTST</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>INQ_STATISTICS_OPTIONS</td>
</tr>
<tr>
<td></td>
<td>RECORD_STATISTICS</td>
</tr>
<tr>
<td></td>
<td>REQUEST_STATISTICS</td>
</tr>
<tr>
<td></td>
<td>SET_STATISTICS OPTIONS</td>
</tr>
<tr>
<td></td>
<td>STATISTICS_COLLECTION</td>
</tr>
<tr>
<td></td>
<td>DISABLE_STATISTICS</td>
</tr>
<tr>
<td>DFHSTTI</td>
<td>Handles the NOTIFY request</td>
</tr>
<tr>
<td>DFHSTTRI</td>
<td>Interprets ST domain trace entries</td>
</tr>
<tr>
<td>DFHSTUE</td>
<td>Provides a SET_EXIT_STATUS routine to enable or disable a user exit.</td>
</tr>
</tbody>
</table>

Exits

There is one global user exit point in the statistics domain: XSTOUT. See the CICS Customization Guide for further information.

Trace

The point IDs for the statistics domain are of the form ST xxxx; the corresponding trace levels are ST 1, ST 2, and Exc.
Statistics domain (ST)

For more information about the trace points, see the CICS Trace Entries. For more information about using traces in problem determination, see the CICS Problem Determination Guide.
Chapter 79. Statistics utility program (DFHSTUP)

This chapter provides a general overview of the collection of CICS statistics as well as describing the operation of the offline statistics utility program (DFHSTUP). For more information about using the DFHSTUP utility program, see the CICS Operations and Utilities Guide.

An operator interface to all online statistics functions is provided by the CEMT transaction. The equivalent programmable interface is provided by the EXEC API.

Statistics may be collected at user-specified intervals from the startup to the shutdown of a CICS system. Statistics may also be requested, resulting in the collection of data for the period between the last time statistics were reset and the time the request was made.

Statistics are also collected at system quiesce or logical end of day; this data is written to the SMF data set as for a normal interval collection.

An option is provided by the statistics domain to allow the user to specify whether interval statistics are to be collected. The statistics domain calls each domain in turn to reset the statistics fields at every interval when statistics are collected. Statistics (particularly interval statistics) can be used for capacity planning and performance tuning. For further information about this aspect, see the CICS Performance Guide.

There is a great similarity between CICS statistics data and CICS performance class monitoring data. Statistics data is collected on a resource basis, whereas performance class monitoring collects similar data on a transaction basis. Statistics can therefore be viewed as resource monitoring.

Design overview

CICS statistics support is divided into the following components:

1. The operator interface. This component is responsible for interfacing to the various CICS-supported terminals, analyzing the input string and then invoking the statistics domain to perform the appropriate management operation. This function is provided by the CEMT transaction, and also by the EXEC API.

2. The statistics domain. This component is responsible for managing statistics interfaces, for example, SMF and EXEC API.

3. The statistics update logic. This code is inline in the relevant resource manager. In this way the control function of statistics is centralized, but the management and updating of the statistics fields is given to the resource owner.

4. The statistics data collection and reset. For all collection types except unsolicited (see below), the collection mechanism is the same. The owning domain is invoked by statistics domain to supply a record that contains the domain’s statistics. When this record has been formed, the domain then calls statistics domain to place the data on the SMF data set.

There are five types of collections:

a. Interval. The collection interval default is 3 hours. This may be changed by the user. The minimum value is 1 minute, the maximum 24 hours. On an
interval collection, each called domain collects and resets its statistics counters. No action is taken if the statistics recording status is OFF.

b. Requested. Statistics may be requested using the PERFORM STATISTICS function provided by the CEMT transaction or the EXEC API. The data recorded is for the period between the last time statistics were reset and the time the request was made. Statistics are reset on an interval, end-of-day, or requested-reset collection; they can also be reset, without a collection, when changing the statistics recording status from ON to OFF, or from OFF to ON.

This type of collection can obtain statistics from some or all domains, as requested. Each called domain collects, but does not reset, its statistics counters.

Requested statistics are collected even if the statistics recording status is OFF.

c. Requested-reset. This collection is similar to requested statistics, except that it always obtains statistics for all domains, and each called domain resets its statistics counters after collection.

Requested-reset statistics are collected even if the statistics recording status is OFF.

d. End-of-day. This collection occurs when the system is quiescing. A logical end-of-day time may be specified. The default time is midnight. This is primarily for continually running systems. The collection is then made at this time, and the called domain collects and resets its statistics counters.

End-of-day statistics are collected even if the statistics recording status is OFF.

Daily systems that are taken down after midnight should change the logical end of day to a time when the system is not operational.

If the user wishes to simulate shutdown statistics, the interval can be set to 24 hours. An end-of-day report, which contains total figures for the CICS run up to the end of the day, can then be printed by DFHSTUP.

e. Unsolicited. For dynamically allocated and deallocated resources, the resource records its statistics just before it is deleted; for example, an autoinstall terminal that logs off and is thereby deleted. USS statistics are written to SMF regardless of the statistics recording status (STATRCD).

5. The statistics formatting control. The offline utility DFHSTUP opens the statistics data set, which is an unloaded SMF data set, and the I/O interfaces to that data set. This routine then browses the data set and formats the statistics.

Reports may be produced for any or all of the five types of statistics collections. DFHSTUP also provides the option of producing a summary report for selected CICS applids. The summary report is constructed from all the statistics contained in the interval, requested-reset, end-of-day, and unsolicited collections. Requested statistics are not involved in the production of the summary report.

**DFHSTUP operation**

DFHSTUP runs as a separate MVS job and extracts all or selected entries from the unloaded SMF data set. The types of entries to be processed by this program are specified in the SYSIN data set. Entries that can be selected for processing include:

- All entries—the default
- Entries written for specified applids
- Entries written for specified resource types
Entries written for specified collection types, that is, interval, requested, requested-reset, end-of-day, or unsolicited

Entries written during a specified period of time.

You can also select:

- The page size; the default is 60 lines per page.
- Whether output is to be printed in mixed case or all uppercase; the default is to print in mixed case.
- The summary report option; by default, it is not selected.

Further information about using DFHSTUP is given in the CICS Operations and Utilities Guide.

### Modules

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHSTUP1</td>
<td>PRE_INITIALIZE</td>
</tr>
<tr>
<td>DFHSTE15</td>
<td>DFSORT interface to E15 user exit</td>
</tr>
<tr>
<td>DFHSTE35</td>
<td>DFSORT interface to E35 user exit</td>
</tr>
<tr>
<td>DFHSTIN</td>
<td>DFSORT E15 user exit input routine</td>
</tr>
<tr>
<td>DFHSTOT</td>
<td>DFSORT E35 user exit output routine</td>
</tr>
<tr>
<td>DFHSTRD</td>
<td>Read interface subroutines</td>
</tr>
<tr>
<td>DFHSTTQX</td>
<td>Transient data statistics summary formatter</td>
</tr>
<tr>
<td>DFHSTWR</td>
<td>Write interface subroutines</td>
</tr>
<tr>
<td>DFHSTUDB</td>
<td>DBCTL statistics formatter</td>
</tr>
<tr>
<td>DFHSTUDS</td>
<td>Dispatcher domain statistics formatter</td>
</tr>
<tr>
<td>DFHSTUDU</td>
<td>Dump domain statistics formatter</td>
</tr>
<tr>
<td>DFHSTUD2</td>
<td>CICS DB2 statistics formatter</td>
</tr>
<tr>
<td>DFHSTULD</td>
<td>Loader domain statistics formatter</td>
</tr>
<tr>
<td>DFHSTUKN</td>
<td>Monitoring domain statistics formatter</td>
</tr>
<tr>
<td>DFHSTUPG</td>
<td>Program manager domain statistics formatter</td>
</tr>
<tr>
<td>DFHSTURS</td>
<td>User domain statistics formatter</td>
</tr>
<tr>
<td>DFHSTUSM</td>
<td>Storage manager domain statistics formatter</td>
</tr>
<tr>
<td>DFHSTUST</td>
<td>Statistics domain statistics formatter</td>
</tr>
<tr>
<td>DFHSTUTQX</td>
<td>Transient data statistics formatter</td>
</tr>
<tr>
<td>DFHSTUUS</td>
<td>Autoinstall terminals unsolicited statistics formatter</td>
</tr>
<tr>
<td>DFHSTUXM</td>
<td>Transaction manager domain statistics formatter</td>
</tr>
<tr>
<td>DFHSTU03</td>
<td>VTAM statistics formatter</td>
</tr>
<tr>
<td>DFHSTU04</td>
<td>Autoinstall terminals statistics formatter</td>
</tr>
<tr>
<td>DFHSTU06</td>
<td>Terminal statistics formatter</td>
</tr>
<tr>
<td>DFHSTU08</td>
<td>LSRPOOL resource statistics formatter</td>
</tr>
<tr>
<td>DFHSTU09</td>
<td>LSRPOOL file statistics formatter</td>
</tr>
<tr>
<td>DFHSTU14</td>
<td>ISC/IRC statistics formatter</td>
</tr>
<tr>
<td>DFHSTU16</td>
<td>Table manager statistics formatter</td>
</tr>
<tr>
<td>DFHSTU17</td>
<td>File control statistics formatter</td>
</tr>
<tr>
<td>DFHSTU21</td>
<td>ISC/IRC attach-time statistics formatter</td>
</tr>
<tr>
<td>DFHSTU22</td>
<td>FEPI statistics formatter</td>
</tr>
<tr>
<td>DFHSTDBX</td>
<td>DBCTL statistics summary formatter</td>
</tr>
<tr>
<td>DFHSTDSX</td>
<td>Dispatcher domain statistics summary formatter</td>
</tr>
<tr>
<td>DFHSTDX</td>
<td>Dump domain statistics summary formatter</td>
</tr>
<tr>
<td>DFHSTD2X</td>
<td>CICS DB2 statistics summary formatter</td>
</tr>
<tr>
<td>DFHSTLDX</td>
<td>Loader domain statistics summary formatter</td>
</tr>
<tr>
<td>DFHSTMNX</td>
<td>Monitoring domain statistics summary formatter</td>
</tr>
<tr>
<td>DFHSTPGX</td>
<td>Program manager domain statistics summary formatter</td>
</tr>
</tbody>
</table>
### Statistics utility program (DFHSTUP)

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHSTSMX</td>
<td>Storage manager domain statistics summary formatter</td>
</tr>
<tr>
<td>DFHSTSTX</td>
<td>Statistics domain statistics summary formatter</td>
</tr>
<tr>
<td>DFHSTURX</td>
<td>User domain statistics summary formatter</td>
</tr>
<tr>
<td>DFHSTXMX</td>
<td>Transaction manager domain statistics summary formatter</td>
</tr>
<tr>
<td>DFHST03X</td>
<td>VTAM statistics summary formatter</td>
</tr>
<tr>
<td>DFHST04X</td>
<td>Autoinstall terminals statistics summary formatter</td>
</tr>
<tr>
<td>DFHST05X</td>
<td>Dynamic transaction backout statistics summary formatter</td>
</tr>
<tr>
<td>DFHST06X</td>
<td>Terminal statistics summary formatter</td>
</tr>
<tr>
<td>DFHST08X</td>
<td>LSRPOOL resource statistics summary formatter</td>
</tr>
<tr>
<td>DFHST09X</td>
<td>LSRPOOL file statistics summary formatter</td>
</tr>
<tr>
<td>DFHST12X</td>
<td>Temporary-storage statistics summary formatter</td>
</tr>
<tr>
<td>DFHST13X</td>
<td>Journal control statistics summary formatter</td>
</tr>
<tr>
<td>DFHST14X</td>
<td>ISC/IRC statistics summary formatter</td>
</tr>
<tr>
<td>DFHST16X</td>
<td>Table manager statistics summary formatter</td>
</tr>
<tr>
<td>DFHST17X</td>
<td>File control statistics summary formatter</td>
</tr>
<tr>
<td>DFHST21X</td>
<td>ISC/IRC attach-time statistics summary formatter</td>
</tr>
<tr>
<td>DFHST22X</td>
<td>FEPI statistics summary formatter</td>
</tr>
</tbody>
</table>
Chapter 80. Storage control macro-compatibility interface

DFHSMSCP is responsible for handling all requests for storage services that are made by using the routine addressed by CSASCNAC in the CICS common system area (CSA). DFHSMSCP is called by some parts of the CICS AP domain containing DFHSC macros.

DFHSMSCP converts all requests into calls to the storage manager domain, and its main function is to get or free storage.

Design overview

The input to DFHSMSCP, set up by the macro used for the invocation, or directly by the calling program, consists of the following TCA fields:

- TCASCTR—the storage control request byte. This can contain one of the following values:
  - X’80’ GETMAIN, in conjunction with:
    - X’40’ Initialize storage
    - X’20’ Conditional
  - Storage class in bits 3 through 7 (the resulting SMMC GETMAIN storage class name is given in parentheses where this differs from the first name):
    - X’00’ 1WD, treated as SHARED
    - X’04’ LINE
    - X’05’ TERMINAL or TERM
    - X’0C’ USER (becomes CICS24)
    - X’0D’ TRANSDATA or TD
    - X’13’ SHARED (becomes SHARED_CICS24)
    - X’14’ CONTROL
  - X’40’ FREEMAIN, in conjunction with:
    - X’01’ TCTTE address supplied.

- TCASCIB—the 1-byte value to which storage is to be initialized.

- TCASCNB—the 2-byte field giving the number of bytes requested on the GETMAIN.

- TCASCSA—the 4-byte address of the storage that was obtained or the storage to be freed.

Modules

DFHSMSCP

Exits

No global user exit points are provided for this function.

Trace

The point IDs for this function are of the form AP F1xx; the corresponding trace levels are SC 1 and Exc.
Storage control macro-compatibility

For more information about the trace points, see the CICS Trace Entries. For more information about using traces in problem determination, see the CICS Problem Determination Guide.
Chapter 81. Storage manager domain (SM)

The storage manager domain (also sometimes known simply as “storage manager”) manages virtual storage requests.

Storage manager domain’s specific gates

Table 101 summarizes the storage manager domain’s specific gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and whether or not the functions are available through the exit programming interface (XPI).

Table 101. Storage manager domain’s specific gates

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>XPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMAD</td>
<td>SM 0201</td>
<td>ADD_SUBPOOL</td>
<td>NO</td>
</tr>
<tr>
<td>SMAD</td>
<td>SM 0202</td>
<td>DELETE_SUBPOOL</td>
<td>NO</td>
</tr>
<tr>
<td>SMAR</td>
<td>SM 0F01</td>
<td>ALLOCATE_TRANSACTION_STG</td>
<td>NO</td>
</tr>
<tr>
<td>SMAR</td>
<td>SM 0F02</td>
<td>RELEASE_TRANSACTION_STG</td>
<td>NO</td>
</tr>
<tr>
<td>SMCK</td>
<td>SM 0901</td>
<td>CHECK_STORAGE</td>
<td>NO</td>
</tr>
<tr>
<td>SMCK</td>
<td>SM 0902</td>
<td>RECOVER_STORAGE</td>
<td>NO</td>
</tr>
<tr>
<td>SMGF</td>
<td>SM 0301</td>
<td>GETMAIN</td>
<td>NO</td>
</tr>
<tr>
<td>SMGF</td>
<td>SM 0302</td>
<td>FREEMAIN</td>
<td>NO</td>
</tr>
<tr>
<td>SMGF</td>
<td></td>
<td>INQUIRE_ELEMENT_LENGTH</td>
<td>NO</td>
</tr>
<tr>
<td>SMCC</td>
<td>SM 0601</td>
<td>INITIALIZE</td>
<td>NO</td>
</tr>
<tr>
<td>SMCC</td>
<td>SM 0602</td>
<td>GETMAIN</td>
<td>YES</td>
</tr>
<tr>
<td>SMCC</td>
<td>SM 0C01</td>
<td>FREEMAIN</td>
<td>YES</td>
</tr>
<tr>
<td>SMCC</td>
<td>SM 0C02</td>
<td>FREEMAIN</td>
<td>YES</td>
</tr>
<tr>
<td>SMCC</td>
<td>SM 0D01</td>
<td>FREEMAIN_ALL_TERMINAL</td>
<td>NO</td>
</tr>
<tr>
<td>SMCC</td>
<td>SM 0D02</td>
<td>INQUIRE_ELEMENT_LENGTH</td>
<td>YES</td>
</tr>
<tr>
<td>SMCC</td>
<td>SM 0E01</td>
<td>INQUIRE_TASK_STORAGE</td>
<td>YES</td>
</tr>
<tr>
<td>SMCC</td>
<td>SM 0E02</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SMSR</td>
<td>SM 0401</td>
<td>INQUIRE_DSA_SIZE</td>
<td>NO</td>
</tr>
<tr>
<td>SMSR</td>
<td>SM 0402</td>
<td>SET_DSA_LIMIT</td>
<td>NO</td>
</tr>
<tr>
<td>SMSR</td>
<td></td>
<td>INQUIRE_DSA_LIMIT</td>
<td>NO</td>
</tr>
<tr>
<td>SMSR</td>
<td></td>
<td>SET_STORAGE_RECOVERY</td>
<td>NO</td>
</tr>
<tr>
<td>SMSR</td>
<td></td>
<td>SET_STORAGE_PROTECT</td>
<td>NO</td>
</tr>
<tr>
<td>SMSR</td>
<td></td>
<td>INQUIRE_STORAGE_PROTECT</td>
<td>NO</td>
</tr>
<tr>
<td>SMSR</td>
<td></td>
<td>INQUIRE_ACCESS_TOKEN</td>
<td>NO</td>
</tr>
<tr>
<td>SMSR</td>
<td></td>
<td>INQUIRE_ACCESS</td>
<td>YES</td>
</tr>
<tr>
<td>SMSR</td>
<td></td>
<td>SET_REENTRANT_PROGRAM</td>
<td>NO</td>
</tr>
<tr>
<td>SMSR</td>
<td></td>
<td>SET_TRANSACTION_ISOLATION</td>
<td>NO</td>
</tr>
<tr>
<td>SMSR</td>
<td></td>
<td>INQUIRE_REENTRANT_PROGRAM</td>
<td>NO</td>
</tr>
<tr>
<td>SMSR</td>
<td></td>
<td>INQUIRE_TRANSACTION_ISOLATION</td>
<td>NO</td>
</tr>
<tr>
<td>SMSR</td>
<td></td>
<td>SWITCH_SUBSPACE</td>
<td>YES</td>
</tr>
<tr>
<td>SMSR</td>
<td></td>
<td>INQUIRE_SHORT_ON_STORAGE</td>
<td>YES</td>
</tr>
<tr>
<td>SMSR</td>
<td></td>
<td>UPDATE_SUBSPACE_TCB_INFO</td>
<td>NO</td>
</tr>
</tbody>
</table>
SMAD gate, ADD_SUBPOOL function

The ADD_SUBPOOL function of the SMAD gate is used to create a new subpool with given attributes.

Input parameters

**USAGE**
indicates whether the subpool is for task or domain use. It can have either of these values:

- TASK
- DOMAIN

**ELEMENT_TYPE**
indicates whether the subpool elements are of fixed or variable length. It can have either of these values:

- FIXED
- VARIABLE

**[FIXED_LENGTH]**
is the element length for a fixed-length subpool.

**ELEMENT_CHAIN**
indicates whether a chain of the addresses and lengths of the elements is to be kept. It can have either of these values:

- YES
- NO

**BOUNDARY**
is the boundary on which all elements within the subpool must be aligned. The boundary must be a power of two in the range 8 through 4096.

**LOCATION**
specifies whether all elements within the subpool must be allocated below the maximum 24-bit address, or may be allocated anywhere. It can have either of these values:

- BELOW
- ANY

**SUBPOOL_NAME**
is the 8-character name by which the subpool is known.

**INITIAL_FREE**
is the size of the initial free storage area for the subpool.

**[STORAGE_CHECK]**
indicates whether storage zone checking is to be enabled for this subpool. It can have either of these values:

- YES
- NO

Output parameters

**SUBPOOL_TOKEN**
is the token identifying the newly created subpool.

**[DSA_NAME]**
is the name of the CICS dynamic storage area (DSA) in which the subpool resides. It can have any of these values:

- CDSA
- UDSA
- SDSA
- RDSA
- ECDSA
- EUDSA
- ESDSA
- ERDSA

**RESPONSE**
is the domain’s response to the call. It can have any of these values:

- OK
- EXCEPTION
- DISASTER
- INVALID
- KERNERROR
- PURGED

**[REASON]**
is returned when RESPONSE is EXCEPTION or INVALID. Possible values
The DELETE_SUBPOOL function of the SMAD gate is used to delete a subpool.

**Input parameters**

**SUBPOOL_TOKEN**

is the token identifying the subpool to be deleted.

**Output parameters**

**RESPONSE**

is the domain’s response to the call. It can have any of these values: OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**[REASON]**

is returned when RESPONSE is INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>INVALID_SUBPOOL_TOKEN, NOT_SUBPOOL_OWNER, SUBPOOL_NOT_EMPTY</td>
</tr>
</tbody>
</table>

The ALLOCATE_TRANSACTION_STG function of the SMAR gate is used at task initialization to add the four task lifetime storage subpools.

**Input parameters**

**TASK_DATALOC**

indicates the location of task data for the transaction, as specified by the TASKDATALOC attribute on the associated TRANSACTION resource definition. It can have either of these values: BELOW|ANY

**TASK_DATAKEY**

indicates the storage key for the task-lifetime storage and program-related storage (for all programs that run under the transaction) for the transaction, as specified by the TASKDATAKEY attribute on the associated TRANSACTION resource definition. It can have either of these values: CICS|USER

**ISOLATE**

indicates whether CICS is to isolate the transaction’s user-key task-lifetime storage to provide application-to-application protection, as specified by the ISOLATE attribute on the associated TRANSACTION resource definition. It can have either of these values: YES|NO

**STORAGE_FREEZE**

indicates whether or not task-lifetime storage freemains should be delayed until task termination. It can have either of these values:
Storage manager domain (SM)

YES|NO

**STORAGE_CLEAR**
indicates whether task lifetime storage should be cleared to zeros when it is freemained. It can have either of these values:
YES|NO

**Output parameters**

**RESPONSE**
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, INSUFFICIENT_STORAGE</td>
</tr>
</tbody>
</table>

**SMAR gate, RELEASE_TRANSACTION_STG function**
The RELEASE_TRANSACTION_STG function of the SMAR gate is used at task termination to freemain all remaining task-lifetime storage and deletes the four task lifetime subpools.

**Input parameters**
None.

**Output parameters**

**RESPONSE**
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, DEACTIVATE_FAILURE, INSUFFICIENT_STORAGE, STORAGE_VIOLATION</td>
</tr>
</tbody>
</table>

**SMCK gate, CHECK_STORAGE function**
The CHECK_STORAGE function of the SMCK gate is used to check the storage check zones of task lifetime storage and the storage accounting areas (SAAs) of terminal storage for consistency.

**Input parameters**

**TASK_STORAGE**
specifies whether the storage check zones of task lifetime storage are to be checked for the current task or all tasks, or is not to be checked. It can have any one of these values:
NO|CURRENT_TASK|ALL_TASKS

**TP_STORAGE**
specifies whether the SAAs of terminal storage are to be checked for the current terminal, or is not to be checked. It can have either of these values:
NO|CURRENT_TERMINAL
Output parameters

**RESPONSE**

is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**[REASON]**

is returned when RESPONSE is DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP, STORAGE_VIOLATION</td>
</tr>
</tbody>
</table>

**SMCK gate, RECOVER_STORAGE function**

The RECOVER_STORAGE function of the SMCK gate is used to recover storage.

**Input parameters**

**TASK_STORAGE**

specifies whether or not the task lifetime storage for the current task is to be recovered. It can have any one of these values:

NO|CURRENT_TASK

**TP_STORAGE**

specifies whether or not the SAAs of terminal storage for the current terminal are to be recovered. It can have either of these values:

NO|CURRENT_TERMINAL

**Output parameters**

**RESPONSE**

is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**[REASON]**

is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>STORAGE_NOT_RECOVERED</td>
</tr>
</tbody>
</table>

**SMGF gate, GETMAIN function**

The GETMAIN function of the SMGF gate is used to allocate an element of storage from a subpool.

**Input parameters**

**Note:** Either STORAGE_CLASS or SUBPOOL_TOKEN, but not both, must be specified.

**[REMARK]**

is an optional 8-character field that is used to identify the GETMAIN operation for problem determination. This field is highlighted when the GETMAIN trace is interpreted. Typically, it is the name of the control block whose storage is being obtained.
Storage manager domain (SM)

[STORAGE_CLASS]
identifies the class of storage that is being allocated. It can have any
one of these values:
CICS|CICS24|USER|USER24|TASK|TASK24

[SUBPOOL_TOKEN]
is a token identifying the subpool within which the element is to be
allocated.

[GET_LENGTH]
is the length of the storage requested.

SUSPEND
If there is insufficient storage to satisfy the request, SUSPEND(YES)
causes the caller to be suspended until the request can be satisfied,
and SUSPEND(NO) causes REASON to be set to
INSUFFICIENT_STORAGE. It can have either of these values:
YES|NO

[INITIAL_IMAGE]
is an optional byte value to which every byte in the new element is
set.

Output parameters
ADDRESS
is the address of the new element.

[ELEMENT_LENGTH]
is the actual length of the new element (when it has been rounded up to a
multiple of the boundary for the subpool).

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID.
Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, ACTIVATE_FAILURE, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>INSUFFICIENT_STORAGE</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_SUBPOOL_TOKEN, INVALID_GET_LENGTH, INVALID_STORAGE_CLASS,</td>
</tr>
<tr>
<td></td>
<td>NO_GET_LENGTH, NOT_SUBPOOL_OWNER INVALID_INITIAL_IMAGE</td>
</tr>
</tbody>
</table>

SMGF gate, FREEMAIN function

The FREEMAIN function of the SMGF gate is used to release an element of storage
within a subpool.

Input parameters

Note: Either STORAGE_CLASS or SUBPOOL_TOKEN, but not both, must be
specified.

[REMARK]
is an optional 8-character field that is used to identify the
FREEMAIN operation for problem determination. This field is
highlighted when the FREEMAIN trace is interpreted. Typically, it is the name of the control block whose storage is being released.

**[STORAGE_CLASS]**
identifies the class of storage that is being released. It can have any one of these values:
CICS|CICS24|USER|USER24|TASK|TASK24

**[SUBPOOL_TOKEN]**
is a token identifying the subpool within which the element is to be released.

**ADDRESS**
is the address of the element to be released.

**[FREE_LENGTH]**
is the length of the element to be released.

**Output parameters**

**RESPONSE**
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**[REASON]**
is returned when RESPONSE is DISASTER or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, DEACTIVATE_FAILURE, LOOP</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_SUBPOOL_TOKEN, INVALID_ADDRESS, INVALID_FREE_LENGTH, INVALID_STORAGE_CLASS, NO_FREE_LENGTH, NOT_SUBPOOL_OWNER, SUBPOOL_EMPTY</td>
</tr>
</tbody>
</table>

**SMGF gate, INQUIRE_ELEMENT_LENGTH function**
The INQUIRE_ELEMENT_LENGTH function of the SMGF gate is used to return the length of an element of storage whose address is known.

**Input parameters**

**Note:** Either STORAGE_CLASS or SUBPOOL_TOKEN, but not both, must be specified.

**[STORAGE_CLASS]**
identifies the class of storage that is being inquired upon. It can have any one of these values:
CICS|CICS24|USER|USER24|TASK|TASK24

**[SUBPOOL_TOKEN]**
is a token identifying the subpool within which the element has been allocated.

**ADDRESS**
is the address of the element whose length is being inquired on.

**Output parameters**

**ELEMENT_LENGTH**
is the length of the element.
Storage manager domain (SM)

RESPONSE is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>ADDRESS_NOT_FOUND</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_STORAGE_CLASS, INVALID_SUBPOOL_TOKEN</td>
</tr>
</tbody>
</table>

SMMC gate, INQUIRE_ELEMENT_LENGTH function

The INQUIRE_ELEMENT_LENGTH function of the SMMC gate is used to obtain the start address and length of the storage element that contains the address that was specified on the input to the call. This function only searches the current task’s task-lifetime storage for the required storage element.

Input parameters

ADDRESS is the address to be searched for.

Output parameters

ELEMENT_LENGTH is the length of the storage element that contains the input address.

[ELEMENT_ADDRESS] is the start address of the element that contains the input address.

RESPONSE is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON] is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>INVALID_ADDRESS</td>
</tr>
</tbody>
</table>

SMMC gate, INQUIRE_TASK_STORAGE function

The INQUIRE_TASK_STORAGE function of the SMMC gate is used to obtain details of all the task-lifetime storage associated with the current task (if the input parameter TRANSACTION_NUMBER is omitted from the call) or for the specified task.

Input parameters

[TRANSACTION_NUMBER] indicates the transaction that you wish to obtain storage details about. If this parameter is omitted, the current task is assumed.
ELEMENT_BUFFER
is a buffer in which the storage manager lists the start addresses of all the specified task’s task-lifetime storage.

LENGTH_BUFFER
is a buffer in which the storage manager lists the lengths of all the specified task’s task-lifetime storage.

Output parameters

NUMBER_OF_ELEMENTS
is the number of elements in each buffer.

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>INSUFFICIENT_STORAGE, NO_TRANSACTION_ENVIRONMENT</td>
</tr>
</tbody>
</table>

SMMC gate, INITIALIZE function
The INITIALIZE function of the SMMC gate is used to perform macro-compatibility interface initialization.

Input parameters
None.

Output parameters

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
</tbody>
</table>

SMMC gate, GETMAIN function
The GETMAIN function of the SMMC gate is used to allocate an element of storage.

Input parameters

[REMARK]
is an optional 8-character field that is used to identify the GETMAIN operation for problem determination. This field is highlighted when the GETMAIN trace is interpreted. Typically, it is the name of the control block whose storage is being obtained.
Storage manager domain (SM)

GET_LENGTH
is the length of storage requested. For storage classes that have 8-byte SAAs, the requested length excludes the lengths of the initial and duplicate SAAs. However, for storage classes that have only a 4-byte SAA, the requested length must include the length of the SAA.

SUSPEND
specifies whether the request is to be suspended if there is insufficient storage to satisfy the request. It can have either of these values:
YES|NO

[INITIAL_IMAGE]
specifies a byte value to which the user’s part of the allocated storage element is to be set.

[TCTTE_ADDRESS]
is an optional field that must be specified for GETMAIN requests for the TERMINAL storage class.

STORAGE_CLASS
is the class of storage to be allocated. It can have any one of these values:
TERMINAL24|CICS|SHARED_CICS|LINE|TERMINAL|
TASK|TASK24|CICS24_SAA|SHARED_CICS24_SAA|
CICS24|TRANSDATA|TEMPSTG|USER|USER24|
SHARED_CICS24|CONTROL|TACLE|SHARED_USER24|
SHARED_USER

[CALLER]
can have any one of these values:
EXEC|MACRO|SYSTEM

Output parameters
ADDRESS
is the address of the allocated storage.

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, ACTIVATE_FAILURE, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>INSUFFICIENT_STORAGE, INVALID_GET_LENGTH</td>
</tr>
<tr>
<td>INVALID</td>
<td>NO_TCTTE_ADDRESS, INVALID_STORAGE_CLASS</td>
</tr>
</tbody>
</table>

SMMC gate, FREEMAIN function
The FREEMAIN function of the SMMC gate is used to release an element of storage.

Input parameters
[REMARK]
is an optional 8-character field that is used to identify the FREEMAIN operation for problem determination. This field is highlighted when the
**Storage manager domain (SM)**

FREEMAIN trace is interpreted. Typically, it is the name of the control block whose storage is being released.

**ADDRESS**

is the address of the storage to be freed.

**[TCTTE_ADDRESS]**

is an optional field that must be specified if the FREEMAIN is for storage of a LINE or TERMINAL class.

**[STORAGE_CLASS]**

is an optional field specifying the class of storage that is being freed. It can have any one of these values:

- TERMINAL24
- CICS
- SHARED_CICS
- LINE
- TERMINAL
- TASK
- TASK24
- CICS24_SA
- SHARED_CICS24_SA
- CICS24
- TRANSDATA
- TEMPSTG
- USER
- USER24
- SHARED_CICS24
- CONTROL
- TACLE
- SHARED_USER24
- SHARED_USER

**[CALLER]**

can have any one of these values:

- EXEC
- MACRO
- SYSTEM

**[EXEC_KEY]**

is the execution key of the program issuing the EXEC FREEMAIN request. It can have either of these values:

- CICS
- USER

**Output parameters**

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

- OK
- EXCEPTION
- DISASTER
- INVALID
- KERNERROR
- PURGED

**[REASON]**

is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, DEACTIVATE_FAILURE, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>INVALID_EXEC_KEY</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_ADDRESS, NO_TCTTE_ADDRESS</td>
</tr>
</tbody>
</table>

**SMMC gate, FREEMAIN_ALL_TERMINAL function**

The FREEMAIN_ALL_TERMINAL function of the SMMC gate is used to release all terminal storage.

**Input parameters**

**TCTTE_ADDRESS**

is the address of the TCTTE whose storage is to be freed.

**Output parameters**

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

- OK
- EXCEPTION
- DISASTER
- INVALID
- KERNERROR
- PURGED
Storage manager domain (SM)

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
</tbody>
</table>

**SMSR gate, INQUIRE_ISOLATION_TOKEN function**

The INQUIRE_ISOLATION_TOKEN function of the SMSR gate is used to return an isolation token which can be used on SWITCH_SUBSPACE calls.

**Input parameters**
None.

**Output parameters**

**ISOLATION_TOKEN**
an isolation token which can be used on SWITCH_SUBSPACE calls.

**RESPONSE**
is the domain’s response to the call. It can have any of these values:

OK|DISASTER|INVALID|KERNERROR

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
</tbody>
</table>

**SMSR gate, INQUIRE_REENTRANT_PROGRAM function**

The INQUIRE_REENTRANT_PROGRAM function of the SMSR gate is used to return whether the read-only DSAs, RDSA and ERDSA, have been allocated from read-only key-0 protected storage or CICS-key storage, as set by the RENTPGM system initialization parameter.

**Input parameters**
None.

**Output parameters**

**RENTPGM**
indicates whether CICS has obtained the storage for the read-only DSAs from key-0 non-fetch protected storage (PROTECT) or from CICS-key storage (NOPROTECT). It can have either of these values:

PROTECT|NOPROTECT

**RESPONSE**
is the domain’s response to the call. It can have any of these values:

OK|DISASTER|INVALID|KERNERROR

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
</tbody>
</table>
**SMSR gate, INQUIRE_SHORT_ON_STORAGE function**

The INQUIRE_SHORT_ON_STORAGE function of the SMSR gate is used to return whether or not CICS is currently short-on-storage.

**Input parameters**

None.

**Output parameters**

**SOS_BELOW_THE_LINE**
indicates whether or not CICS is short-on-storage below the 16MB line. It can have either of these values:

YES | NO

**SOS ABOVE THE LINE**
indicates whether or not CICS is short-on-storage above the 16MB line. It can have either of these values:

YES | NO

**RESPONSE**
is the domain’s response to the call. It can have any of these values:

OK | DISASTER | INVALID | KERNERROR

**[REASON]**
is returned when RESPONSE is DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
</tbody>
</table>

**SMSR gate, INQUIRE_DSA_SIZE function**

The INQUIRE_DSA_SIZE function of the SMSR gate is used to return the size of the CICS DSAs.

**Input parameters**

**DSA NAME**
is the name of the DSA whose size is being inquired on. It can have any of these values:

CDSA | UDSA | SDSA | RDSA | ECDSA | EUDSA | ESDSA | ERDSA

**Output parameters**

**DSA_SIZE**
is the size of the DSA.

**RESPONSE**
is the domain’s response to the call. It can have any of these values:

OK | DISASTER | INVALID | KERNERROR

**[REASON]**
is returned when RESPONSE is DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
</tbody>
</table>
SMSR gate, SET_STORAGE_RECOVERY function

The SET_STORAGE_RECOVERY function of the SMSR gate is used to set the storage recovery option.

Input parameters

RECOVERY
is the value to which the storage recovery option is to be set. It can have either of these values:
YES|NO

Output parameters

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|DISASTER|INVALID|KERNERROR

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
</tbody>
</table>

SMSR gate, SET_TRANSACTION_ISOLATION function

The SET_TRANSACTION_ISOLATION function of the SMSR gate is used to set whether or not you want transaction isolation in your CICS region. This value is initially set by the TRANISO system initialization parameter.

Input parameters

TRANSACTION_ISOLATION
indicates whether or not transaction isolation is active in your CICS region.
It can have either of these values:
ACTIVE|INACTIVE

Output parameters

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|DISASTER|INVALID|KERNERROR

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
</tbody>
</table>

SMSR gate, SWITCH_SUBSPACE function

The SWITCH_SUBSPACE function of the SMSR gate is used to change a task’s subspace.

Input parameters

SPACE
indicates the type of subspace you wish this task to execute in. It can have any of these the values:
Storage manager domain (SM)

BASESPACE|SUBSPACE|RESET_SPACE

[ISOLATION_TOKEN]
an isolation token which can be returned from an
INQUIRE_ISOLATION_TOKEN call.

[TRANSACTION_TOKEN]
a transaction manager token (which can be returned from an XMIQ
INQUIRE_TRANSACTION_TOKEN call) that represents the task whose
subspace you wish to change.

[WORK_REGISTER]
a work register.

Output parameters

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|DISASTER|INVALID|KERNERROR

REASON
is returned when RESPONSE is DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
</tbody>
</table>

SMSR gate, INQUIRE_DSA_LIMIT function

The INQUIRE_DSA_LIMIT function of the SMSR gate is used to return the DSA
storage limits above (EDSA) and below (DSA) the 16MB line. These limits are the
maximum amounts of storage that CICS can use for all the DSAs above and below
the 16MB line.

Input parameters

None.

Output parameters

[DSA_LIMIT]
indicates the DSA storage limit.

[EDSA_LIMIT]
indicates the EDSA storage limit.

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|DISASTER|INVALID|KERNERROR

REASON
is returned when RESPONSE is DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
</tbody>
</table>

SMSR gate, SET_DSA_LIMIT function

The SET_DSA_LIMIT function of the SMSR gate is used to set the DSA storage
limits above (EDSA) and below (DSA) the 16MB line. These limits are the
maximum amounts of storage that CICS can use for all the DSAs above and below
the 16MB line.
Storage manager domain (SM)

**Input parameters**

[DSA_LIMIT] indicates the DSA storage limit required.

[EDSA_LIMIT] indicates the EDSA storage limit required.

**Output parameters**

RESPONSE is the domain’s response to the call. It can have any of these values:

OK|DISASTER|INVALID|KERNERROR

REASON is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>INSUFFICIENT_STORAGE, INVALID_DSA_LIMIT</td>
</tr>
</tbody>
</table>

**SMSR gate, SET_STORAGE_PROTECT function**

The SET_STORAGE_PROTECT function of the SMSR gate is used to set the storage protection option.

**Input parameters**

STORAGE_PROTECT can have either of these values:

YES|NO

**Output parameters**

RESPONSE is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR

REASON is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>NO_HARDWARE_SUPPORT</td>
</tr>
</tbody>
</table>

**SMSR gate, INQUIRE_STORAGE_PROTECT function**

The INQUIRE_STORAGE_PROTECT function of the SMSR gate is used to return the current value of the storage protection option.

**Input parameters**

None.

**Output parameters**

STORAGE_PROTECT is the current storage protection mode. It can have either of these values:
RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|DISASTER|INVALID|KERNERROR

[REASON]
is returned when RESPONSE is DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
</tbody>
</table>

**SMSR gate, INQUIRE_ACCESS function**

The INQUIRE_ACCESS function of the SMSR gate is used to return the access key of an element of storage.

**Input parameters**

**[ACCESS_TOKEN]**
is the access token for the element of storage (returned by the INQUIRE_ACCESS_TOKEN function).

**ELEMENT_ADDRESS**
is the start address of the storage element.

**ELEMENT_LENGTH**
is the length of the storage element.

**Output parameters**

**ACCESS**
is the type of access for the storage element. It can have any of these values:
CICS|USER|READ_ONLY

**[DSA_NAME]**
is the name of the DSA in which the storage element resides.

**[DSA_EXTENT_START]**
indicates the start address of the DSA extent that contains the input address.

**[DSA_EXTENT_END]**
indicates the end address of the DSA extent that contains the input address.

**RESPONSE**
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR

[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>INVALID_ELEMENT</td>
</tr>
</tbody>
</table>
Storage manager domain (SM)

**SMSR gate, SET_REENTRANT_PROGRAM function**
The SET_REENTRANT_PROGRAM function of the SMSR gate is used to set the reentrant program option for the RDSA and the ERDSA.

**Input parameters**

**REENTRANT_PROGRAM**
is the reentrant program option for the RDSA and the ERDSA. It can have either of these values:

- PROTECT
- NOPROTECT

**Output parameters**

**RESPONSE**
is the domain’s response to the call. It can have any of these values:

- OK
- EXCEPTION
- DISASTER
- INVALID
- KERNERROR

[REASON] is returned when RESPONSE is DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
</tbody>
</table>

**SMSR gate, INQUIRE_ACCESS_TOKEN function**
The INQUIRE_ACCESS_TOKEN function of the SMSR gate is used to return the access token for a storage element.

**Input parameters**

None.

**Output parameters**

**ACCESS_TOKEN**
is the access token for the storage element.

**RESPONSE**
is the domain’s response to the call. It can have any of these values:

- OK
- EXCEPTION
- DISASTER
- INVALID
- KERNERROR

[REASON] is returned when RESPONSE is DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
</tbody>
</table>

**SMSR gate, UPDATE_SUBSPACE_TCB_INFO function**
The UPDATE_SUBSPACE_TCB_INFO function informs SM of the deletion of open TCBs which are associated with subspaces.

**Input parameters**

**SUBSPACE_TOKEN**
indicates the subspace which is associated with the deleted TCBs.

**OPEN_TCBS_DELETED**
is a 32-bit string indicating the mode(s) of deleted TCB(s).
Output parameters

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK|DISASTER|INVALID|KERNERROR

### Storage manager domain’s generic gates

Table 102 summarizes the storage manager domain’s generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

**Table 102. Storage manager domain’s generic gates**

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMDM</td>
<td>SM 0101</td>
<td>PRE_INITIALIZE</td>
<td>DMDM</td>
</tr>
<tr>
<td></td>
<td>SM 0102</td>
<td>INITIALIZE_DOMAIN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>QUIESCE_DOMAIN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TERMINATE_DOMAIN</td>
<td></td>
</tr>
<tr>
<td>STST</td>
<td>SM 0A01</td>
<td>COLLECT_STATISTICS</td>
<td>STST</td>
</tr>
<tr>
<td></td>
<td>SM 0A02</td>
<td>COLLECT_RESOURCE_STATS</td>
<td></td>
</tr>
</tbody>
</table>

For descriptions of these functions and their input and output parameters, refer to the sections dealing with the corresponding generic formats:

**Functions and parameters**

- **Format DMDM**—“Domain manager domain’s generic formats” on page 377
- **Format STST**—“Statistics domain’s generic format” on page 1087

In preinitialization processing, the storage manager domain sets the initial storage options:

- The amount of storage to be allocated to the dynamic storage area
- The amount of storage to be allocated to the extended dynamic storage area
- The storage recovery option
- The state of the storage protect, transaction isolation and the reentrant program option.

For a cold start, the information comes from the system initialization parameters; for any other type of start, the information comes from the local catalog, but is then modified by any relevant system initialization parameters.

Storage manager domain also issues console messages during preinitialization to report the amount of storage allocated above and below the line for DSA use.

In initialization, quiesce, and termination processing, the storage manager domain performs only internal routines.
Storage manager domain (SM)

Storage manager domain’s generic format

Table 103 shows the generic format owned by the storage manager domain, and shows the function performed on the call.

Table 103. Generic format owned by the storage manager domain

<table>
<thead>
<tr>
<th>Format</th>
<th>Calling module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMNT</td>
<td>DFHSMSY</td>
<td>STORAGE_NOTIFY</td>
</tr>
</tbody>
</table>

In the descriptions of the format that follow, the “input” parameters are input not to the storage manager domain, but to the domain being called by the storage manager. Similarly, the “output” parameters are output by the domain that was called by the storage manager domain, in response to the call.

Format SMNT, STORAGE_NOTIFY function

The STORAGE_NOTIFY function of SMNT format is used to notify free storage above and below the 16MB line.

Input parameters

DSAS_CONSTRAINED YES|NO
indicates whether any DSA is currently constrained due to lack of free storage.

FREE_BYTES_CDSA
is the largest free area available (in bytes) in the CICS DSA below the 16MB line (not including the cushion).

FREE_BYTES_UDSA
is the largest free area available (in bytes) in the user-key DSA below the 16MB line (not including the cushion).

FREE_BYTES_SDSA
is the largest free area available (in bytes) in the shared user-key DSA below the 16MB line (not including the cushion).

FREE_BYTES_RDSA
is the largest free area available (in bytes) in the read-only DSA below the 16MB line (not including the cushion).

FREE_BYTES_ECDSA
is the largest free area available (in bytes) in the CICS DSA above the 16MB line (not including the cushion).

FREE_BYTES_EUDSA
is the largest free area available (in bytes) in the user-key DSA above the 16MB line (not including the cushion).

FREE_BYTES_ESDSA
is the largest free area available (in bytes) in the shared user-key DSA above the 16MB line (not including the cushion).

FREE_BYTES_ERDSA
is the largest free area available (in bytes) in the read-only DSA above the 16MB line (not including the cushion).

Output parameters

RESPONSE
is the domain’s response to the call. It can have any of these values:
[REASON]
is returned when RESPONSE is DISASTER or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>LOOP, ABEND</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

### Modules

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHSMAD</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>ADD_SUBPOOL</td>
</tr>
<tr>
<td></td>
<td>DELETE_SUBPOOL</td>
</tr>
<tr>
<td>DFHSMAR</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>ALLOCATE_TRANSACTION_STG</td>
</tr>
<tr>
<td></td>
<td>RELEASE_TRANSACTION_STG</td>
</tr>
<tr>
<td>DFHSMCK</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>CHECK_STORAGE</td>
</tr>
<tr>
<td></td>
<td>RECOVER_STORAGE</td>
</tr>
<tr>
<td>DFHSMMDM</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>PRE_INITIALIZE</td>
</tr>
<tr>
<td></td>
<td>INITIALIZE_DOMAIN</td>
</tr>
<tr>
<td></td>
<td>QUIESCE_DOMAIN</td>
</tr>
<tr>
<td></td>
<td>TERMINATE_DOMAIN</td>
</tr>
<tr>
<td>DFHSMDFS</td>
<td>SM domain offline dump formatting routine</td>
</tr>
<tr>
<td>DFHSMGF</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>GETMAIN</td>
</tr>
<tr>
<td></td>
<td>FREEMAIN</td>
</tr>
<tr>
<td></td>
<td>INQUIRE_ELEMENT_LENGTH</td>
</tr>
<tr>
<td>DFHSMMC1</td>
<td>SM domain macro-compatibility interface INITIALISE function</td>
</tr>
<tr>
<td>DFHSMMC2</td>
<td>SM domain macro-compatibility interface which handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>FREEMAIN_ALL_TERMINAL</td>
</tr>
<tr>
<td></td>
<td>INQUIRE_ELEMENT_LENGTH</td>
</tr>
<tr>
<td></td>
<td>INQUIRE_TASK_STORAGE</td>
</tr>
<tr>
<td>DFHSMMF</td>
<td>SM domain macro-compatibility interface FREEMAIN function</td>
</tr>
<tr>
<td>DFHSMMG</td>
<td>SM domain macro-compatibility interface GETMAIN function</td>
</tr>
</tbody>
</table>
### Storage manager domain (SM)

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHSMR</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>INQUIRE_ACCESS</td>
</tr>
<tr>
<td></td>
<td>INQUIRE_ACCESS_TOKEN</td>
</tr>
<tr>
<td></td>
<td>INQUIRE_DSA_LIMIT</td>
</tr>
<tr>
<td></td>
<td>INQUIRE_DSA_SIZE</td>
</tr>
<tr>
<td></td>
<td>INQUIRE_REENTRANT_PROGRAM</td>
</tr>
<tr>
<td></td>
<td>INQUIRE_SHORT_ON_STORAGE</td>
</tr>
<tr>
<td></td>
<td>INQUIRE_STORAGE_PROTECT</td>
</tr>
<tr>
<td></td>
<td>INQUIRE_TRANSACTION_ISOLATION</td>
</tr>
<tr>
<td></td>
<td>SET_DSA_LIMIT</td>
</tr>
<tr>
<td></td>
<td>SET_REENTRANT_PROGRAM</td>
</tr>
<tr>
<td></td>
<td>SET_STORAGE_RECOVERY</td>
</tr>
<tr>
<td></td>
<td>SET_STORAGE_PROTECT</td>
</tr>
<tr>
<td></td>
<td>SWITCH_SUBSPACE</td>
</tr>
<tr>
<td>DFHSMST</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>COLLECT_STATISTICS</td>
</tr>
<tr>
<td></td>
<td>COLLECT_RESOURCE_STATS</td>
</tr>
<tr>
<td>DFHSMVC</td>
<td>Gets DSAs</td>
</tr>
<tr>
<td>DFHSMY</td>
<td>SM domain system task—issues STORAGE_NOTIFY requests</td>
</tr>
<tr>
<td>DFHSMTRI</td>
<td>Interprets SM domain trace entries</td>
</tr>
</tbody>
</table>

#### Exits

No global user exit points are provided in this domain.

#### Trace

The point IDs for the storage manager domain are of the form SM xxxx; the corresponding trace levels are SM 1 and Exc.

For more information about the trace points, see the [CICS Trace Entries](CICS_Trace_Entries). For more information about using traces in problem determination, see the [CICS Problem Determination Guide](CICS_Problem_Determination_Guide).
Chapter 82. Subsystem interface

The subsystem interface is a mechanism by which the MVS operating system communicates with its underlying subsystems at certain critical points in its processing.

CICS is required to be defined as a formal MVS subsystem for the following purposes:
• Multiregion operation (MRO)
• Shared database support
• Console message handling.

Functional overview

An MVS subsystem consists of two control blocks and a set of functional routines, all resident in common memory. The control blocks are:

SSCT  The subsystem communication table, which contains the 4-character name of the subsystem and a pointer to the SSVT.

SSVT  The subsystem vector table, which contains a list of the subsystem function codes that the subsystem supports, and the addresses of the functional routines that support them.

The subsystem is active when the SSCT contains a nonzero pointer to the SSVT, and inactive when the pointer is zero.

Subsystem definition

Each subsystem is defined to MVS by an entry in an IEFSSNxx member of SYS1.PARMLIB. (See the OS/390 MVS Initialization and Tuning Guide [SC28-1751].) Each subsystem can be defined with an initialization routine and some initialization parameters. The CICS subsystem is defined with an initialization routine of DFHSSIN, and an initialization parameter that specifies the name of an additional member of SYS1.PARMLIB, which contains further CICS-specific subsystem parameters. These parameters specify whether the console message handling facility is required.

Design overview

When the recommended initialization routine DFHSSIN is specified, the CICS subsystem is initialized during the master scheduler initialization phase of the MVS IPL. The CICS-specific subsystem parameters are read from SYS1.PARMLIB, and the subsystem vector table is created. The supporting subsystem function routines are loaded into common memory and their addresses are stored into the subsystem vector table. If everything is successful, the CICS subsystem is made active by storing the address of the subsystem vector table in the subsystem communication table.

Console message handling

At startup, a CICS region that supports console message handling notifies the CICS subsystem of its existence, by using the CICS SVC to issue a subsystem interface call for the ‘generic connect’ function with a CONNECT subfunction. The subsystem notes the creation of the new region and, if this is the first such CICS
region to become active, invokes a service of MVS console support called
“subsystem console message broadcasting”. The message broadcasting service
causes all subsequent console messages to be broadcast to all subsystems that have
expressed an interest in receiving them, including the CICS subsystem. This MVS
service can also be activated by other products, for example, NetView.

If the message broadcasting service has been activated, either by CICS or by
another product, the CICS subsystem examines all messages issued by WTO
macros in any address space, but it intercepts and modifies only the following:

- Messages beginning with “DFH” that are issued under any CICS TCB, including
  those CICS regions that do not have console message handling support.
  These messages are reformatted to contain the CICS applid for the region in a
  standard position in the message.

Because the CICS subsystem receives control after JES has recorded a console
message in the job’s message log, messages in the job log do not appear to be
reformatted. The messages are only reformatted on the operator consoles and in
the MVS system log.

If the original message is a long one, inserting the CICS applid can cause the
message to exceed the maximum length for an MVS console message. In this
case, the original message is suppressed (that is, does not appear on the
console), and the reformatted message is issued using the MVS multiple-line
console message service to split the message text over several lines. Both the
original message and perhaps several instances of the reformatted multiple-line
message appear in the job log, but only one copy of the reformatted message is
displayed on the console.

- Messages that redisplay, on operator consoles or in the MVS system log,
  MODIFY commands that are directed towards CICS and contain signon
  passwords for the CESN transaction.
  These messages are reformatted with the passwords replaced by asterisks, so
  that the original passwords are not exposed.

As each TCB terminates, it issues an ‘end of task’ subsystem call, which is
broadcast to all active subsystems. Likewise, as each address space terminates, it
issues an ‘end of memory’ subsystem call, which is also broadcast to all active
subsystems. When it receives either of these calls, the CICS subsystem first calls
the end-of-memory routine in DFHIRP; then, if the terminating address space is
known by the subsystem, it invokes the ‘generic connect’ function with a
DISCONNECT subfunction.

The DISCONNECT subfunction notes the termination of the CICS address space
and, if this is the last CICS containing console message handling support to
terminate, notifies the “subsystem console message broadcasting” support that the
CICS subsystem is no longer interested in receiving broadcast console messages.
Nevertheless, if another product has kept console message broadcasting active, the
CICS subsystem continues to reformat messages from CICS regions that do not
have console message handling support.
## Control Blocks

<table>
<thead>
<tr>
<th>DSECT</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHSABDS</td>
<td>The CICS subsystem anchor block (SAB). This is used to contain global subsystem-related information that is common to all CICS regions in the MVS image. It is used to record the options specified in the DFHSSInn member of SYS1.PARMLIB. It contains a pointer to a bit map that records which MVS address spaces contain an active CICS. It also contains the address of the subsystem control table extension (SCTE) used by IRC, and the address of the CEC status tracking information used by XRF.</td>
</tr>
<tr>
<td>IEFJSCVT</td>
<td>The subsystem communication table (SSCT). This is an MVS control block. There is one SSCT for each subsystem, including the primary job entry subsystem (JES) as well as CICS.</td>
</tr>
<tr>
<td>IEFJSSVT</td>
<td>The subsystem vector table (SSVT). This is an MVS control block. There is one SSVT for each active subsystem. It contains a lookup table for determining which function codes are supported by the subsystem, and a list of the entry points for all the supporting function routines.</td>
</tr>
</tbody>
</table>

*Figure 87 on page 1122* shows these control blocks.
Subsytem interface

<table>
<thead>
<tr>
<th>MVS CVT</th>
</tr>
</thead>
<tbody>
<tr>
<td>X'128' CVTJESCT</td>
</tr>
<tr>
<td>Address of JES communication table</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>JESCT</th>
</tr>
</thead>
<tbody>
<tr>
<td>X'18' JESSSCT</td>
</tr>
<tr>
<td>Address of first SSCT</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SSCT (for JES)</th>
</tr>
</thead>
<tbody>
<tr>
<td>X'04' SSCTSCTA</td>
</tr>
<tr>
<td>Address of next SSCT</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SSCT (for MSTR)</th>
</tr>
</thead>
<tbody>
<tr>
<td>X'04' SSCTSCTA</td>
</tr>
<tr>
<td>Address of next SSCT</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SSCT (for CICS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>X'04' SSCTSCTA</td>
</tr>
<tr>
<td>Address of next SSCT</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SSCT</th>
</tr>
</thead>
<tbody>
<tr>
<td>X'10' SSCTSSVT</td>
</tr>
<tr>
<td>Address of SSVT for CICS</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SAB (for CICS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>X'00' SABCDD</td>
</tr>
<tr>
<td>Address of CEC 'inoperative' data</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SAB (for MSTR)</th>
</tr>
</thead>
<tbody>
<tr>
<td>X'04' SABSCTE</td>
</tr>
<tr>
<td>Address of SCTE</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SAB (for JES)</th>
</tr>
</thead>
<tbody>
<tr>
<td>X'18' SABMAPPT</td>
</tr>
<tr>
<td>Address of bit map for active CICS regions</td>
</tr>
</tbody>
</table>

Figure 87. Control blocks associated with the subsystem interface

## Modules

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHSSIN</td>
<td>Subsystem initialization routine for the CICS subsystem. Reads in subsystem parameters from member DFHSSInn of SYS1.PARMLIB, creates SSVT, loads function modules into MVS common storage.</td>
</tr>
<tr>
<td>DFHSSEN</td>
<td>End-of-task and end-of-memory functional module. Calls DFHIRP’s EOT/EOM routine. Issues ‘generic connect’ if terminating region or job-step task is in the CICS address space map.</td>
</tr>
</tbody>
</table>
### Subsystem interface

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHSSGC</td>
<td>The generic connect functional module. CONNECT subfunction sets the bit for the current address space in the address space map. If this is the first CICS region to start, it invokes IEAVG700 to initiate message broadcasting. DISCONNECT subfunction unsets the bit for the current address space in the address space map. If this is the last CICS region to finish, it invokes IEAVG700 to terminate message broadcasting.</td>
</tr>
<tr>
<td>DFHSSMGP</td>
<td>Message routine for DFHSSIN.</td>
</tr>
<tr>
<td>DFHSSMGT</td>
<td>Message table for DFHSSIN.</td>
</tr>
<tr>
<td>DFHSSWT</td>
<td>Router module for the console message handler. Calls DFHSSWTO for messages beginning with DFH. Calls DFHSSWTF for messages that echo MODIFY commands.</td>
</tr>
<tr>
<td>DFHSSWTF</td>
<td>Suppresses passwords from the echoed copies of MODIFY CICS commands that contain signon passwords.</td>
</tr>
<tr>
<td>DFHSSWTO</td>
<td>Inserts the applid into all DFH messages issued under a TCB with a valid AFCB.</td>
</tr>
</tbody>
</table>

### Exits

There are no user exits in the subsystem interface support.

### Trace

No tracing is performed by the subsystem interface support.

### External interfaces

Module DFHSSIN invokes the MVS module IEEMB878 to read its initialization parameters from SYS1.PARMLIB.

Module DFHSSGC invokes the MVS module IEAVG700 to start and stop console message subsystem broadcasting.

Modules DFHCSCVC and DFHSSEN use the IEFSSREQ interface to communicate with the CICS subsystem.
Chapter 83. Subtask control

Subtask control is the interface between a CICS task and a subtask. It avoids suspending CICS execution, and improves the response time.

This function is invoked by the DFHSK macro with the following calls:
- CTYPE=PERFORM activates an exit routine under a new TCB.
- CTYPE=WAIT waits for subtask to complete.
- CTYPE=RETURN returns control to the main CICS TCB.

Design overview

Some synchronous operating system requests issued by CICS modules could cause CICS to be suspended until the requests had completed. To avoid the resulting response-time degradation, certain requests are processed by the general-purpose subtask control program, DFHSKP. A CICS module calls DFHSKP to execute a routine within the module under a subtask of the operating system.

DFHSKP does the following:
- Schedules a subtask to execute a routine (called an SK exit routine)
- Allows an SK exit routine to wait on an event control block (ECB) of the operating system
- Manages subtask creation, execution, and termination
- Handles program checks or abends within the SK exit routine.

DFHSKP consists of the DFHSKM, DFHSKC, and DFHSKE programs.

DFHSKM (subtask manager program)

A DFHSK macro invokes DFHSKM to cause a routine to be executed under a subtask of the operating system. DFHSKM chooses a subtask to execute the request unless the caller has specified a particular subtask.

DFHSKM determines whether the subtask is inoperative, not started, or running. The subtask is called inoperative if it has terminated itself, or could not be attached. If the subtask is inoperative and the user coded SYNC=YES in the DFHSK macro, the request is processed synchronously; that is, DFHSKM executes the request under the CICS task control block (TCB).

If the subtask has not started, DFHSKM attaches a CICS task specifying the entry point of DFHSKC to execute. DFHSKM then waits on an ECB in the subtask control area (SKA) for the subtask and continues when the ECB is posted by DFHSKC, indicating that the subtask has been initialized.

DFHSKM then creates a work queue element (WQE) that represents the work to be performed under a subtask. The WQE is added to the work queue for the subtask. When the work ECB of the subtask is posted, signaling work to do, DFHSKM issues a wait on the work-complete ECB in the WQE. This ECB is posted when the WQE has been processed by the subtask. DFHSKM returns control to the caller, indicating the outcome of the processing.
If the subtask processing the WQE fails before completion, DFHSKM is informed and attempts to execute the request synchronously if the caller so specified.

When CICS terminates, it issues a DFHSK CTYPE=TERMINATE macro to terminate the subtasking mechanism. DFHSKM sets a flag in each subtask control area (in DFHSDKP static storage) indicating that the subtask should terminate. DFHSKM then posts the subtask work ECB to signal the subtask to examine this flag.

DFHSKM is also invoked by deferred work element (DWE) processing.

When DFHSKM decides to process a WQE synchronously, control is passed to the routine specified by the caller. This routine may not complete normally and, so that DFHSKM does not lose the WQE because the task abended, it creates a DWE containing the address of the WQE. If the task abends, the DWE processor adds the WQE to the free queue.

**DFHSKC (subtask control program)**

DFHSDKM invokes DFHSKC using the DFHKCP attach logic to start a subtask of the operating system, and wait for its completion. DFHSKM passes the address of the subtask control area in the facility control area address (TCAFCAA) in the TCA.

DFHSKC issues an EXEC CICS GETMAIN for shared storage to pass to the subtask for use as its automatic storage. The length required is in a field in DFHSDK containing the automatic storage requirements. DFHSKC issues the ATTACH macro with the ECB option to attach the operating system subtask, and passes the address of the subtask control area.

DFHSKC issues the CICS SVC to authorize the TCB of the subtask to use the SVC.

DFHSKC issues a KC wait on the attach ECB. The module is suspended until subtask termination, when the ECB is posted. On termination, the subtask puts a return code in the subtask control area.

When the subtask completes, DFHSKC cleans up the subtask work queue. It then frees the automatic storage and terminates.

DFHSKC writes messages to CSMT from this module if it was unable to attach a subtask of the operating system subtask, or the subtask indicated that its termination was not normal.

**DFHSKE (subtask exit program)**

When the subtask manager DFHSDKM, executing on behalf of a CICS task, decides that a subtask is to be started, it attaches a CICS task using the DFHKC ATTACH macro and specifying the entry point of DFHSDKCNA. This CICS task attaches the subtask and waits for subtask completion by means of the ECB parameter coded in the ATTACH macro.

The ATTACH macro specifies an entry point in DFHSIP (known to MVS by an IDENTIFY macro issued in DFHSIP). DFHSIP then branches to the entry point of DFHSKE, whose address is in the subtask control area.

**Note:** DFHSIP remains in storage after initialization has completed.
The subtask reverses the order of the in-progress queue to service requests on a first-come, first-served basis. It then loops round the in-progress queue and, for each WQE, branches to the program specified in the WQE (the SK exit routine).

The exit routine returns control to DFHSKE, either indicating that the exit routine has completed by issuing a DFHSK CTYPE=RETURN macro or requesting that execution of the SK exit is suspended until an ECB specified by the exit is posted by some component of the operating system.

When a return is requested, the ECB in the WQE is posted, causing the dispatcher domain to resume the CICS task that was waiting for the SK exit to be complete. When a wait is requested, the WQE is added to the waiting queue, which is processed later.

When all WQEs in the in-progress queue have been processed, DFHSKE examines the waiting queue. If any WQEs are on this queue, their ECB addresses are inserted into an operating-system multiple-wait queue. The subtask work ECB (posted when a WQE is added to the work queue) is put at the top of this multiple-wait list. An operating-system multiple-wait is then issued.

When the subtask regains control, an ECB has been posted. This can be because more work has arrived or because an ECB belonging to an exit routine has been posted.

The WQEs on the waiting queue are scanned, and those whose ECB has been posted are moved to the in-progress queue, with a flag on indicating that an SK exit routine is to be resumed.

Control returns to the beginning of this program which examines the work queue and proceeds as described earlier.

DFHSKE handles program checks and operating system abends. If an abend exit is driven when processing a WQE, that WQE is blamed and processing of it terminates. The CICS task requesting the processing is informed of the problem.

If an abend exit is driven when a WQE is not being processed, it is assumed to be a problem in the subtasking program. The abend is handled, and a count of failures is increased. When a threshold is reached, the subtask terminates.

The MVS exits are ESTAE and SPIE.

For normal termination, DFHSKE loops, processing WQEs and waiting when there is no work to do. The subtask checks a flag in the subtask control area to see if it has been requested to terminate. If the flag is set, the subtask terminates, indicating normal termination by setting a response code in the subtask control area for the attacher, DFHSKC.

Abnormal termination may occur when the error threshold has been reached. The subtask terminates, but sets an error return code in the subtask control area for the attacher to see. The attacher, DFHSKC, then cleans up any outstanding WQEs on the subtask queues.

Control blocks

This function has the following control blocks:
Subtask control

- SK static storage contains pointers to free work queue elements (WQEs) and to work queue elements.
- SKRQLIST is the parameter area passed to DFHSKP on a request. It contains the address of the code to be executed, and the address of the ECB.
- DFHSKWPS is the WKE structure containing the address of the next WQE in the chain, the contents of the parameter field from CTYPE=PERFORM, the save area for registers, and the work-complete ECB.
- DFHSKAPS is the subtask control area. Each instance of this control block describes the state of one subtask and contains the address of automatic storage to be used by DFHSKE, pointers to the WQE used by the subtask, the current WQE being processed, and the ECB for work and completion.

See the [CICS Data Areas](#) manual for a detailed description of these control blocks.

### Modules

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHSKC</td>
<td>The subtask control program is invoked by DFHSKM to start up a subtask of the operating system</td>
</tr>
<tr>
<td>DFHSKE</td>
<td>The general-purpose multitask program is executed as a subtask of the operating system</td>
</tr>
<tr>
<td>DFHSKM</td>
<td>The subtask manager program causes the routine to execute under a subtask.</td>
</tr>
</tbody>
</table>

### Exits

No global user exit points are provided for this function.

### Trace

The following point ID is provided for this function:

- AP 00DE, for which the trace level is AP 1.

For more information about the trace points, see the [CICS Trace Entries](#). For more information about using traces in problem determination, see the [CICS Problem Determination Guide](#).

### External interfaces

The following external calls are made by DFHSKC:

- **MVS ATTACH**  
  To attach a new TCB
- **MVS DETACH**  
  To detach a TCB
- **MVS POST**  
  To post a CICS TCB.

The following external calls are made by DFHSKE:

- **MVS ESTAE**  
  To establish an error exit
- **MVS WAIT**  
  To synchronize with the TCB
MVS SETRP
To retry after a failure.
Subtask control
Chapter 84. Syncpoint program

This allows the user to specify logical units of work by means of syncpoints. Any processing performed between syncpoints (provided the resources are declared as recoverable) can be reversed in the event of an error; but after a given syncpoint has been reached, the processing performed before that syncpoint cannot be reversed.

A syncpoint is also taken automatically at the end of each task.

Design overview

The syncpoint program works in conjunction with the Recovery Manager domain to provide the user with the ability to establish points in application programs at which all recoverable updates are committed. (The user can, at any time, back out any uncommitted changes by means of the rollback function.)

The syncpoint interface is provided by the DFHSP module. DFHSP is invoked, via the EXEC Interface module DFHEISP, when an application program issues an EXEC CICS SYNCPOINT or SYNCPOINT ROLLBACK command. It is also called from other CICS modules, such as DFHIRMS.

Further important information about syncpoint processing is given in “Chapter 42 Function shipping” on page 673 and “Chapter 68, Recovery Manager Domain (RM)” on page 919.

DFHSP implements syncpoint calls by in turn calling the Recovery Manager domain with DFHRMUWM COMMIT_UOW or BACKOUT_UOW requests. RM calls its clients with prepare, commit, start backout etc. calls. One of RM’s clients is ‘APUS’, serviced by module DFHAPRC. Depending on the call from RM DFHAPRC calls DFHSP or DFHDDBP to process Deferred Work Elements (DWEs). DWEs provide a mechanism whereby resource owners can record their need to perform actions at a syncpoint. Most resource owners provide their own RM client routines, but a few, such as interval control, use DWEs.

Note that the implicit syncpoint or rollback performed at task termination is effected by a direct call to the RM domain, not by issuing a DFHSP macro.

Task-related user exit resynchronization

The purpose of task-related user exit resynchronization is to allow a resource manager to ask CICS for the resolution of UOWs about which it is in-doubt. Task related user exit resynchronization is called as a result of an EXEC CICS RESYNC command to restore the CICS end of the thread that was interrupted by the failure of the connection with the resource manager.

DFHRMSY is passed a parameter list by DFHERMRS which consists of the following: rmi entryname (8 bytes) - the name of the TRUE to be called for resync. rmi qualifier (8 bytes) - the qualifier to the name of the TRUE to be called for resync. uowid (8 bytes) - the id of the UOW to be resynchronized resync type (1 byte) - a flag indicating whether this is a resync as a result of an EXEC CICS RESYNC command or due to a Recovery manager domain unshunt.
DFHRMSY’s job is to call the named TRUE with a resync call giving the resolution of the named UOW. The resolution can be commit, backout, should not be indoubt or lost to initial start. (Lost to initial start means that a START=INITIAL has been performed subsequent to the indoubt UOW being created. Initial start clears the log and the catalog meaning that Recovery Manager has no knowledge of the UOW.)

In order to find the outcome of the UOW, DFHRMSY issues a INITIATE_RECOVERY call to Recovery manager domain for the named UOW, which returns the UOW status. DFHRMSY then builds the resync plist to pass to the TRUE, and calls the TRUE using a DFHRMCAL macro. On return from the TRUE, if the TRUE returns an OK response indicating that it has successfully resynced with its resource manager, then DFHRMSY issues a TERMINATE_RECOVERY call to RECOVERY manager domain specifying FORGET(YES). This tells RM domain it can remove this TRUE’s involvement in the UOW. If no other components or TRUEs are waiting resync for the UOW, then RM domain will delete it’s knowledge of the UOW. If the TRUE does not return with an ok response, FORGET(NO) is specified on the TERMINATE_RECOVERY call, and RM domain retains this UOW for this TRUE. A subsequent resync will be required.

Control blocks

This section describes the control blocks used by the syncpoint program:

- Deferred work element (DWE)

See the CICS Data Areas manual for a detailed description.

Deferred work element (DWE)

A deferred work element (DWE) is created and placed on a DWE chain to save information about actions that must be taken when the unit of work terminates. These actions may depend upon whether the UOW commits or backouts.

DWEs are created by CICS control modules, and chained off field TCADWLBA in the task’s TCA using DWECHAN as the chain field. The module that creates a DWE inserts the entry address of a DWE processor in field DWESVMNA of that DWE. Control is passed to this DWE processor by the syncpoint program at the end of the task or UOW.

DWEs can be used for work to be done before or after the syncpoint is logged or in the event of transaction backout.

The layout of DWEs is defined by the DFHDWEPS structure and by the DFHDWEDS assembler DSECT.

Modules

DFHSPP, DFHAPRC, DFHDBP

DFHSPP

DFHSPP can be invoked by the following macros:

**DFHSP TYPE=USER**

Take a syncpoint
**Syncpoint program**

**DFHSP TYPE=ROLLBACK**
- Roll back the current unit of work

**DFHSP TYPE=PHASE_1**
- Do DWE processing for prepare

**DFHSP TYPE=PHASE_2**
- Do DWE processing for commit

When DFHSPP is called by means of a DFHSP TYPE=USER or TYPE=ROLLBACK macro the request is converted into a call to the Recovery Manager domain to commit or backout the current UOW. If the RM request fails SPP calls DFHAPAC to select an abend code corresponding to the failure reported by RM (for example ASP1 for an in-doubt failure) and, in most cases, issues a PC ABEND with this abend code.

In the case of a commit or backout failure, however, no PC ABEND is issued and the transaction continues normally. In these cases CICS has, for the present, been unable to bring all local resources to the committed state for this unit of work. It has recorded any data necessary to re-attempt this at some later time, and has retained any locks necessary to preserve data integrity until then.

When DFHSPP is called by means of a DFHSP TYPE=PHASE_1 or TYPE=PHASE_2 macro SPP processes any DWEs in the DWE chain (TCADWLBA). The TYPE=PHASE_1 call is issued by DFHAPRC in response to an RM prepare or end_backout request. For each DWE in the chain that is not marked as cancelled (DWECNLM ON) or phase_2 only (DWEPHS2 ON) the DWE processor (entry address DWESVMNA) is called. In the prepare case SPP collects ‘votes’ and may return a YES, NO or READ-ONLY vote to its caller. Also, if necessary, a DL/I TERM call is issued to allow DFHDLI to perform end-of-UOW actions. The TYPE=PHASE_2 call is issued by DFHAPRC in response to an RM commit or shunt request. For each DWE in the chain that is marked phase 2 and not cancelled the DWE processor is called. In the shunt case any DWE that is marked for shunting (DWESHUNT ON) is retained in the DWE chain. All other DWEs are freed.

**DFHDBP**

DFHDBP is link-edited with DFHAPRC and is called by DFHAPRC in response to an RM start_backout request. For each DWE in the task’s DWE chain that is not marked cancelled it marks the DWE as ‘backout’ (DWEDYNB ON). For any BMS DWE it issues a DFHBMS TYPE=PURGE request to discard the incomplete message, otherwise it calls the DWE processor then marks the DWE as cancelled.

**DFHAPRC**

DFHAPRC is the module which provides the gate for the ‘APUS’ Recovery Manager client. It provides keypoint and restart support for user written log records, which is described elsewhere, and syncpoint support where it serves as a receiver for RMRO calls from the RM domain for prepare, commit, etc. which it converts into appropriate calls to SPP or DBP described above.

**Exits**

No global user exit points are provided for this function.
Trace

The following point IDs are provided for this function:

- AP 00CB, for which the trace level is AP 1.
- AP D8xx, for which the trace level is AP 1.

For more information about the trace points, see the CICS Trace Entries. For more information about using traces in problem determination, see the CICS Problem Determination Guide.
Chapter 85. System dump formatting program

The system dump formatting program is for use on MVS system dump (SYS1.DUMP) data sets that record system dumps requested by CICS via the MVS SDUMP macro.

The program is invoked via the interactive problem control system (IPCS). You can use IPCS either interactively or from an MVS batch job.

The CICS-supplied sample system dump formatting program for use with CICS Transaction Server for z/OS, Version 2 Release 1 control blocks is called DFHPD510. In earlier CICS releases, the CICS dump formatting program for use under the IPCS is supplied as DFHPDX.

For further information about the system dump formatting programs, about using IPCS to format and analyze CICS dumps, and about the dump exit parameters available, see the CICS Operations and Utilities Guide.

Design overview

The system dump formatting program produces a formatted listing of CICS control blocks grouped within functional area. CICS dump exit parameters can be specified on the IPCS VERBEXIT subcommand to indicate whether the control block output is to be produced or suppressed for each functional (component) area. Summary reports are available for certain of the functional areas, and the dump exit parameters can also indicate whether these are to be produced or suppressed.

Modules

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHAIUF</td>
<td>Autoinstall terminal model manager formatter</td>
</tr>
<tr>
<td>DFHAPTRA</td>
<td>Application domain multiregion operation trace interpreter</td>
</tr>
<tr>
<td>DFHAPTRB</td>
<td>Application domain extended recovery facility trace interpreter</td>
</tr>
<tr>
<td>DFHAPTRC</td>
<td>Application domain user exit trace interpreter</td>
</tr>
<tr>
<td>DFHAPTRD</td>
<td>Application domain trace interpreter</td>
</tr>
<tr>
<td>DFHAPTRE</td>
<td>Application domain data tables trace interpreter</td>
</tr>
<tr>
<td>DFHAPTRF</td>
<td>Application domain SAA Communications and Resource Recovery interfaces trace interpreter</td>
</tr>
<tr>
<td>DFHAPTRG</td>
<td>Application domain ZC exception and VTAM exit trace interpreter</td>
</tr>
<tr>
<td>DFHAPTRI</td>
<td>Application domain trace interpretation router</td>
</tr>
<tr>
<td>DFHAPTRJ</td>
<td>Application domain ZC VTAM interface trace interpreter</td>
</tr>
<tr>
<td>DFHAPTRL</td>
<td>Application domain CICS OS/2 LU2 mirror trace interpreter</td>
</tr>
<tr>
<td>DFHAPTRN</td>
<td>Application domain autoinstall terminal model manager trace interpreter</td>
</tr>
<tr>
<td>DFHAPTRO</td>
<td>Application domain LU6.2 application request logic trace interpreter</td>
</tr>
<tr>
<td>DFHAPTRP</td>
<td>Application domain program control trace interpreter</td>
</tr>
<tr>
<td>DFHAPTRR</td>
<td>Application domain partner resource manager trace interpreter</td>
</tr>
<tr>
<td>DFHAPTRS</td>
<td>Application domain DFHEISR trace interpreter</td>
</tr>
<tr>
<td>DFHAPTRV</td>
<td>Application domain DFHSRP trace interpreter</td>
</tr>
<tr>
<td>DFHAPTRW</td>
<td>Front End Programming Interface feature trace interpreter</td>
</tr>
<tr>
<td>DFHAPTR0</td>
<td>Application domain old-style trace entry interpreter</td>
</tr>
<tr>
<td>DFHAPTR2</td>
<td>Application domain statistics trace interpreter</td>
</tr>
</tbody>
</table>
## System dump formatting program

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHAPTR4</td>
<td>Application domain transaction manager trace interpreter</td>
</tr>
<tr>
<td>DFHAPTR5</td>
<td>Application domain file control trace interpreter</td>
</tr>
<tr>
<td>DFHAPTR6</td>
<td>Application domain DBCTL DL/1 trace interpreter</td>
</tr>
<tr>
<td>DFHAPTR7</td>
<td>Application domain LU6.2 transaction routing trace interpreter</td>
</tr>
<tr>
<td>DFHAPTR8</td>
<td>Application domain security trace interpreter</td>
</tr>
<tr>
<td>DFHAPTR9</td>
<td>Application domain interval control trace interpreter</td>
</tr>
<tr>
<td>DFHCCTRI</td>
<td>CICS catalog trace interpreter</td>
</tr>
<tr>
<td>DFHCSDUF</td>
<td>SAA Communications and Resource Recovery interfaces formatter</td>
</tr>
<tr>
<td>DFHCSDUF</td>
<td>CSA and CSA optional features list formatter</td>
</tr>
<tr>
<td>DFHDDBUF</td>
<td>DBCTL and remote DL/1 dump formatter</td>
</tr>
<tr>
<td>DFHDCCUF</td>
<td>CICS catalog formatter</td>
</tr>
<tr>
<td>DFHCSTRI</td>
<td>CICS catalog trace interpreter</td>
</tr>
<tr>
<td>DFHCSDUF</td>
<td>CSA and CSA optional features list formatter</td>
</tr>
<tr>
<td>DFHDUDUF</td>
<td>Dump domain formatter</td>
</tr>
<tr>
<td>DFHDUF</td>
<td>Formatting router</td>
</tr>
<tr>
<td>DFHDUFUT</td>
<td>Service functions routine</td>
</tr>
<tr>
<td>DFHDTRI</td>
<td>Dump domain trace interpreter</td>
</tr>
<tr>
<td>DFHFCDUF</td>
<td>Error message index processor</td>
</tr>
<tr>
<td>DFHFCEUT</td>
<td>File control formatter</td>
</tr>
<tr>
<td>DFHFREDUF</td>
<td>File control recoverable work elements formatter</td>
</tr>
<tr>
<td>DFHFICDF</td>
<td>Interval control formatter</td>
</tr>
<tr>
<td>DFHHPSP</td>
<td>Table of CICS entries for the IPCS exit control table</td>
</tr>
<tr>
<td>DFHHPDFU</td>
<td>Kernel stack internal procedure formatter</td>
</tr>
<tr>
<td>DFHKEDUF</td>
<td>Kernel domain formatter</td>
</tr>
<tr>
<td>DFHKLOC</td>
<td>Routine for locating domain anchors</td>
</tr>
<tr>
<td>DFHKETRI</td>
<td>Kernel domain trace interpreter</td>
</tr>
<tr>
<td>DFHLDDUF</td>
<td>Loader domain formatter</td>
</tr>
<tr>
<td>DFHLSTRI</td>
<td>Loader domain trace interpreter</td>
</tr>
<tr>
<td>DFHLMDFU</td>
<td>Lock manager formatter</td>
</tr>
<tr>
<td>DFHLMTRI</td>
<td>Lock manager trace interpreter</td>
</tr>
<tr>
<td>DFHMEDFU</td>
<td>Message domain formatter</td>
</tr>
<tr>
<td>DFHMETRI</td>
<td>Message domain trace interpreter</td>
</tr>
<tr>
<td>DFHMDUF</td>
<td>Monitoring domain formatter</td>
</tr>
<tr>
<td>DFHMNTRI</td>
<td>Monitoring domain trace interpreter</td>
</tr>
<tr>
<td>DFHMURDF</td>
<td>Multiregion operation formatter</td>
</tr>
<tr>
<td>DFHNMDFU</td>
<td>Control block index processor</td>
</tr>
<tr>
<td>DFHPADFU</td>
<td>Parameter manager formatter</td>
</tr>
<tr>
<td>DFHPAURI</td>
<td>Parameter manager trace interpreter</td>
</tr>
<tr>
<td>DFHPKPS</td>
<td>Input parameter string validation routine</td>
</tr>
<tr>
<td>DFHPSX1</td>
<td>Control program</td>
</tr>
<tr>
<td>DFHPGDFU</td>
<td>Program manager formatter</td>
</tr>
<tr>
<td>DFHPGTRI</td>
<td>Program manager trace interpreter</td>
</tr>
<tr>
<td>DFHRMDFU</td>
<td>Partner resource manager formatter</td>
</tr>
<tr>
<td>DFHRMDUF</td>
<td>Program control table formatter</td>
</tr>
<tr>
<td>DFHMDTRI</td>
<td>Resource recovery manager formatter</td>
</tr>
<tr>
<td>DFHMDUF</td>
<td>Storage manager formatter</td>
</tr>
<tr>
<td>DFHMSMTRI</td>
<td>Storage manager trace interpreter</td>
</tr>
<tr>
<td>DFHSNTRI</td>
<td>Application domain signon trace interpreter</td>
</tr>
<tr>
<td>DFHSNDUF</td>
<td>Static storage area formatter</td>
</tr>
<tr>
<td>DFHSTDFU</td>
<td>Statistics domain formatter</td>
</tr>
</tbody>
</table>
Module Function

DFHSTTRI Statistics domain trace interpreter
DFHSUDUF Dump domain summary formatter
DFHSUTRI Subroutine trace interpreter
DFHSZDUF Front End Programming Interface feature dump formatter
DFHTCDUF Terminal control formatter
DFHTDDUF Transient data formatter
DFHTDTRI Transient data trace interpreter
DFHTIDUF Timer domain formatter
DFHTITRI Timer domain trace interpreter
DFHTMDUF Table manager formatter
DFHTRDUF Trace domain formatter
DFHTRFFD Trace entry data field formatter
DFHTRFFE Trace entry formatter
DFHTRFPB Routine to process blocks of trace entries
DFHTRFPP Routine for selecting trace entries to be printed
DFHTRIB Trace entry interpretation string builder
DFHTRTRI Trace domain trace interpreter
DFHTSDUF Temporary-storage formatter
DFHUEDUF User exit formatter
DFHUSDUF User domain dump formatter
DFHUSTRI User domain trace interpreter
DFHXMDUF Transaction manager domain formatter
DFHXMTRI Transaction manager domain trace interpreter
DFHXSDUF Security domain dump formatter
DFHXSTRI Security domain trace interpreter
DFHXRDUF Extended recovery facility (XRF) formatter
DFHZXDUF XRF ZCP queue formatter

Exits

Global user exit points are not applicable to offline utilities.

Trace

Trace points are not applicable to offline utilities. However, the output obtained and any messages issued by the system dump formatting program may provide clues to problems associated with corrupted data.

For more information about the trace points, see the [CICS Trace Entries](#). For more information about using traces in problem determination, see the [CICS Problem Determination Guide](#).

External interfaces

The following external calls are used by the system dump formatting program:

- MVS GETMAIN and FREEMAIN for storage management
- OPEN SVC to open DFHSNAP
- CLOSE SVC to close DFHSNAP
- MVS IPCS service routines.
System dump formatting program
Chapter 86. System recovery program

The system recovery programs, DFHSR1, DFHSRP, and DFHSRLI, together form the default CICS recovery routine for the application (AP) domain. This routine is, in particular, the recovery routine for program checks, operating system abends, and runaway tasks that occur in user application code.

Design overview

The CICS kernel intercepts program checks, runaway tasks, operating system abends and some other internal errors for all CICS domains. The kernel then selects which CICS recovery routine to pass control to. The selected recovery routine can then process the error as appropriate.

The DFHSR1 module is the default recovery routine for the application domain. It receives control if any of the above errors occur in CICS system application programs, user application programs and some CICS nucleus modules. It processes internal errors itself but, when dealing with program checks, operating system abends, and runaway task abends, it calls the DFHSRP module. The DFHSRP module, in turn, converts the error into a transaction abend, if possible; if not possible, it terminates CICS. The DFHSRP module uses subroutines in DFHSRLI.

The transaction abend codes that may be issued are:

- **AEYD**  
  error detected by command protection
- **AICA**  
  task runaway
- **AKEF**  
  domain gate not active
- **AKEG**  
  kernel stack storage GETMAIN failure.
- **ASRA**  
  program check
- **ASRB**  
  operating system abend
- **ASRD**  
  illegal macro call or attempt to access the CSA or TCA
- **ASRK**  
  TCA not available
- **xxxx**  
  as set by issuers of deferred abend

The processing associated with each of these abends is described in "Error handling" on page 1140.

For further information about the abends, see the CICS Messages and Codes manual.

System recovery table

Associated with DFHSRP is the system recovery table (SRT). This is a table that the user can provide, containing operating system abend codes. It controls whether CICS recovers from program checks and operating system abends in noncritical code.
System recovery program

You specify the name of the system recovery table by the SRT system initialization parameter, as either SRT=NO or SRT=xx, where xx is the two-character suffix of the SRT:

- If NO is coded, CICS does not recover from program checks or operating system abends, and terminates if one occurs.
- If a suffix is coded, CICS attempts to recover from all types of program check, but can only recover from an operating system abend if the abend code appears in the SRT identified by the suffix (for example, DFHSRT1A where 1A is the suffix). If the abend code is not in the SRT, CICS terminates.

For information about how to create the SRT, see the CICS Resource Definition Guide.

Recovery initialization

The DFHSII1 module calls the DFHSRI1 module during AP Domain initialization. The DFHSRI1 module tells the Kernel that it is the default recovery routine for the AP domain and adds the ABAB gate.

If any error occurs when informing the kernel, CICS is terminated with message DFHSR0605 and a system dump because it is not possible to run CICS without AP domain recovery.

Error handling

The DFHSRI1 module gets control from the kernel or from other AP domain modules. It decides whether it is dealing with an internal error or an external error such as a program check. Internal errors are dealt with by exiting from the recovery environment and issuing the appropriate kernel call. If either of the DFHXFP or DFHEMS modules has caused a program check, the DFHSRI1 module exits from the recovery environment and passes control to DFHXFP or DFHEMS.

All other external errors are passed on to the DFHSRP module. If control returns from the DFHSRP module, DFHSRI1 issues a transaction abend. If control returns from the abend call, it is because the XPCTA exit has requested retry; in which case, DFHSRI1 restores the registers etc and branches to the resume address.

The DFHSRP module makes an exception trace entry, ensures it is running on the QR TCB and then deals with one of the following:

- Program check (see “Program check” on page 1141)
- Operating system abend (see “Operating system abend” on page 1142)
- Runaway task (see “Runaway task” on page 1142)
- Kernel gate error (see “Kernel gate error” on page 1143)
- Deferred abend. (see “Deferred abend” on page 1143).

Note: The kernel recovery environment is terminated very soon after DFHSRP receives control. This ensures that DFHSRP gets driven again if a subsequent error occurs in DFHSRP itself (rather than the kernel percolating the error to the next kernel stack entry). DFHSRP is therefore in a position to detect such recursive errors, and can take the appropriate action.

If DFHSRP can abend the transaction, it builds a Transaction Abend Control Block (TACB) to describe the abend. The TACB is a task-lifetime control block that records details of a transaction abend. This TACB may be used by the rest of AP domain that needs information about the abend. DFHSRP builds the TACB, rather than letting Program Control build it as part of DFHPC TYPE=ABEND processing.
This enables DFHSRP to include extra information in the TACB that would otherwise be lost, such as GP registers, PSW, and FP registers at the time of the error.

**Program check**
The following processing takes place for a program check, in the order given:

1. If this program check occurred while DFHSRP was in the middle of processing a previous program check, then CICS is terminated with message DFHSR0602 and a system dump. Otherwise DFHSRP may get caught in a recursive loop.
2. If this program check occurred while DFHSRP was in the middle of processing an operating system abend, then CICS is terminated with message DFHSR0615 and a system dump. This traps program checks in global user exit XSRAB.
3. If DFHEIP hired gun checking caused the program check, create an abend record for abend code AEYD and return to DFHSR1.
4. If the program check was an 0C4 protection exception, DFHSRP diagnoses the 0C4 further in order to establish whether it was caused by an attempt to access or overwrite CICS-managed protected storage. Such storage is as follows:
   - The fetch-protected dummy CSA block
   - The CDSA
   - The ECDSA
   - The ERDSA.
   - The EUDSA.
   - The RDSA.
   - The UDSA.
   Of the above, it should be noted that one can only 0C4 on the CDSA or ECDSA if storage protection is active, while 0C4 on the UDSA or EUDSA can only be obtained if transaction isolation is active.
   This diagnosis is accomplished by disassembling the failing instruction, and examining the instruction operands in conjunction with the execution conditions at the time of the 0C4 (such as execution key). If the dummy CSA caused the 0C4 (that is, an attempt was made to access the CSA or TCA, or an illegal macro call was issued), message DFHSR0618 is issued. If a DSA caused the 0C4, message DFHSR0622 is issued.
5. If the SRT=NO system initialization parameter was specified, you have disabled recovery, and CICS terminates with message DFHSR0603 and a system dump.
6. If a CICS system task was in control at the time of the program check, indicated by a non-numeric transaction number, CICS is terminated with message DFHSR0601 and a system dump.
7. Some special processing is performed which applies only to PL/I programs.
8. DFHSRLI is called to determine the following information:
   - The program in which the program check occurred
   - The offset in that program
   - The execution key.
9. The results of the diagnosis (program, offset, execution key, and, if an 0C4 abend, any “hit” DSA) are output in an exception trace.
10. Message DFHAP0001 or DFHSR0001 is issued and a system dump is taken. (See also [“System dump suppression” on page 1143](#).)
Whether message DFHAP0001 or DFHSR0001 is issued is governed by the execution key at the time of the program check. If the program was running in user key, message DFHSR0001 is issued; otherwise, message DFHAP0001 is issued.

11. Finally, DFHSRP creates an abend record and returns to DFHSR1.

**Operating system abend**

The following processing takes place for an operating system abend, in the order given:

1. If this abend occurred while DFHSRP was in the middle of processing a previous operating system abend, then CICS is terminated with message DFHSR0612 and a system dump. Otherwise, DFHSRP may get caught in a recursive loop.

2. If the SRT=NO system initialization parameter was specified, you have disabled recovery, and CICS terminates with message DFHSR0606. A system dump may be taken, if specified on the operating system abend.

3. If the SRT=xx system initialization parameter was specified, DFHSRP searches the SRT with the suffix xx (that is, DFHSRTxx) for the abend code. If it does not find the abend code, CICS terminates with message DFHSR0606. A system dump may be taken, if specified on the operating system abend.

4. When the abend code has been located, the next check is to see if the operating system abend occurred in a CICS system task, indicated by a non-numeric transaction number. If so, CICS terminates with message DFHSR0613 and a system dump.

5. Otherwise, the default decision is to abend the transaction with code ASRB. However, you can modify this decision by coding a global user exit program at exit point XSRAB. In addition to performing any processing that might be required for particular operating system abends, the XSRAB exit point allows you to specify whether to:
   • Terminate CICS
   • Abend the transaction ASRB
   • Abend the transaction ASRB, but cancel any active HANDLE ABEND exits.

6. If you choose to terminate CICS, CICS terminates with message DFHSR0606. A system dump may be taken, if specified on the operating system abend.

7. DFHSRLI is called to determine the following information:
   • The program in which the program check occurred
   • The offset in that program
   • The execution key.

8. The results of the diagnosis (program, offset, and execution key) are output in an exception trace.

9. Message DFHAP0001 or DFHSR0001 is issued and a system dump is taken. (See also [System dump suppression](#) on page 1143.) Whether message DFHAP0001 or DFHSR0001 is issued is governed by the execution key at the time of the program check. If the program was running in user key, message DFHSR0001 is issued; otherwise, message DFHAP0001 is issued.

10. Finally, DFHSRP The DFHSRP module creates an abend record with abend code ASRB returns to DFHSR1.

**Runaway task**

One of the following processing options takes place for a runaway task:
System recovery program

- If this runaway task occurred while DFHSRP was in the middle of processing an operating system abend, CICS terminates with message DFHSR0612 and a system dump. This traps runaway tasks caused by errors in global user exit XSRAB.
- Otherwise, the DFHSRP module creates an abend record with abend code AICA and returns to DFHSR1.

Kernel gate error
One of the following processing options takes place for a kernel gate error:

- If this error occurred while DFHSRP was in the middle of processing an operating system abend, CICS terminates with message DFHSR0612 and a system dump. This traps kernel gate errors from XPI calls in global user exit XSRAB.
- Otherwise, the DFHSRP module issues message DFHAP0001, creates an abend record with abend code AKEF, and returns to DFHSR1.

Kernel stack GETMAIN error
The processing that takes place for a kernel stack GETMAIN error is identical to the processing for a kernel gate error, except that the transaction is abended with abend code AKEG.

Deferred abend
The DFHSRP module creates an abend record using the abend code set by the code that issued the deferred abend and returns to DFHSR1.

DFHSRLIM interface

This interface is used to call program DFHSRLI. It provides the following functions for DFHSRP:

**INVOKE_XSRAB**
This function invokes global user exit XSRAB if active, passing to it structure SRP_ERROR_DATA which contains details of the operating system abend that occurred. The abend recovery option selected by the exit is returned, which is either to terminate CICS, abend the transaction ASRB, or abend the transaction ASRB and cancel any active abend exits.

**DIAGNOSE_ABEND**
This function diagnoses a program check, operating system abend, or other error, to establish the location of the error. It returns the program in which the error occurred, the offset within that program, and whether the error occurred in CICS or user application code. (A decision based on the execution key; user key implies user application code.)

System dump suppression
When message DFHAP0001 or DFHSR0001 is issued before the transaction is abended with ASRA, ASRB, ASRD, AKEF, or AKEG, the default is to take a system dump with dumpcode AP0001 or SR0001 respectively. Message DFHSR0001 is issued if CICS is running with storage protection active and is running in user key at the time of the error; otherwise, message DFHAP0001 is issued.

Therefore, it is possible to suppress the system dumps taken for errors occurring in code that is being run in user key (user application code), while retaining system dumps for errors occurring in code that is being run in CICS key (CICS code), by adding SR0001 to the dump table specifying that no system dump is to be taken.
Note that the XDUREQ Global User Exit can be used to distinguish between AP0001 situations in application and non-application code. This allows selective dump suppression when storage protection is not active or when it is active but some applications run in CICS key.

### Modules

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHSRP</td>
<td>Called by DFHSR1 to process program checks, operating system abends, runaway tasks, and so on.</td>
</tr>
<tr>
<td>DFHSRL1</td>
<td>Provides functions for DFHSRP, via the DFHSRLIM interface.</td>
</tr>
<tr>
<td>DFHSR1</td>
<td>The default recovery routine for the AP Domain.</td>
</tr>
</tbody>
</table>

### Exits

There is one global user exit point in DFHSR1: XSRAB. This exit can be called if an operating system abend has occurred and the abend code is in the SRT.

For further information about using the XSRAB exit, see the CICS Customization Guide.

### Trace

The following trace point IDs are provided for DFHSRP and DFHSRL1:

- AP 0701, for which the trace entry level is AP 2
- AP 0702, for which the trace entry level is AP 2
- AP 0780, for which the trace entry level is Exc
- AP 0781, for which the trace entry level is Exc
- AP 0782, for which the trace entry level is Exc
- AP 0783, for which the trace entry level is Exc.
- AP 0790, for which the trace entry level is Exc
- AP 0791, for which the trace entry level is Exc
- AP 0792, for which the trace entry level is Exc
- AP 0793, for which the trace entry level is Exc.
- AP 0794, for which the trace entry level is Exc
- AP 0795, for which the trace entry level is Exc
- AP 0796, for which the trace entry level is Exc
- AP 0797, for which the trace entry level is Exc.
- AP 0798, for which the trace entry level is Exc
- AP 0799, for which the trace entry level is Exc.
- AP 079A, for which the trace entry level is Exc.

For more information about the trace points, see the CICS Trace Entries. For more information about using traces in problem determination, see the CICS Problem Determination Guide.
Chapter 87. System spooler interface

A system programmer can communicate with the local system spooler and, consequently, with other system spoolers via the system spooler network facilities. The system spooler interface single-threads its input, and it is the user’s responsibility to see that all transactions get the chance to run. One high-priority transaction should not use the interface exclusively.

Further information about the system spooler interface is given in the CICS Application Programming Reference.

Design overview

The system spooler interface program opens a system spooler file for either input or output, reads or writes a file, and closes a file. These functions are for system programmer use. The input is single-threaded, so only one transaction can use it at a time.

An application can send files to a remote location by specifying the node of the location, and the userid (or external writer name) of the user at that location. To retrieve a file at the remote location, you specify the external writer name, and you can then retrieve reports from that writer. For security reasons, the external writer name must begin with the same four characters as the CICS applid. The remote system to which a file or report is sent, or from which it is received, must have JES under MVS, or VM.

System spooler interface modules

The SPOOLOPEN command dynamically allocates input or output files using the CICS SVC, and an application control block (ACB) is opened to process the file. For an input file, the IEFSSREQ macro is also issued to determine which file to process. The SPOOLREAD or SPOOLWRITE commands cause GETs or PUTs to be issued using the ACB. The SPOOLCLOSE command dynamically deallocates a file, and causes it to be either transmitted or deleted. All processing which could cause CICS to be suspended is performed under an operating system subtask which is initiated by subtask control, DFHSKP.

DFHPSPST runs under CICS, but DFHPSPSS, and modules called as a result, run under the subtask.

Normal flow

When a system spooler interface command is executed, the normal sequence of invocation of modules is:
1. DFHEIP
2. DFHEPS
3. DFHPSP
4. DFHPSPSS
5. DFHPSPST
6. DFHPSSVC.

DFHPSP is called by:
• Application programs via DFHEPS issuing the DFHPS macro.
System spooler interface

- Syncpoint program and dynamic transaction backout program to the deferred work element (DWE) module (DFHPSPDW). The entry address of DFHPSPDW is stored in the DWE. DFHPSPDW then calls DFHPSFST via DFHPS.

Abnormal flow

If a user transaction terminates without issuing a SPOOLCLOSE command, DFHPSPDW is invoked to process a DWE that was set up when the SPOOLOOPEN command was processed. This closes the file in the usual way.

Modules

<table>
<thead>
<tr>
<th>Module</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHEIP</td>
<td>DFHEIP initializes the EXEC interface structure (EIS) and then invokes the application program. Each EXEC CICS command invokes DFHEIP (nucleus) which in turn invokes the appropriate interface processor. DFHEIP also returns information to the application program through EIB (within EIS).</td>
</tr>
<tr>
<td>DFHEPS</td>
<td>DFHEPS is the link between DFHEIP and the JES interface program, DFHPS.</td>
</tr>
<tr>
<td>DFHPS</td>
<td>DFHPS is the system spooler interface control module.</td>
</tr>
<tr>
<td>DFHSPCK</td>
<td>DFHSPCK is the JES interface termination processor.</td>
</tr>
<tr>
<td>DFHPSPDW</td>
<td>DFHPSPDW is the DWE processor.</td>
</tr>
<tr>
<td>DFHPSPSS</td>
<td>The system spooler interface subtask module attaches a subtask to check that a writer name and a token have been supplied. It opens and closes JES data sets, reads a record, and writes a record.</td>
</tr>
<tr>
<td>DFHPSFST</td>
<td>DFHPSFST is the JES interface controller.</td>
</tr>
<tr>
<td>DFPSSVC</td>
<td>DFPSVC is the system spooler interface module that retrieves a data set name for a given external writer name, dynamically allocates it, and returns its DDNAME.</td>
</tr>
</tbody>
</table>

Exits

No global user exit points are provided for this interface.

Trace

The following point ID is provided for this interface:
- AP 00E3, for which the trace level is AP 1.

For more information about the trace points, see the CICS Trace Entries. For more information about using traces in problem determination, see the CICS Problem Determination Guide.
Chapter 88. Table manager

The table manager controls the locating, adding, deleting, locking, and unlocking of entries in certain CICS tables. These operations can be performed while CICS is running.

Design overview

Locating, adding, deleting, locking, and unlocking entries in tables such as the file control table (FCT), application file control table (AFCT), data set name block table (DSNT), and terminal control table (TCT) are performed by the table manager program, DFHTMP. Entries in these tables are also called “resources”. Because the structures of tables vary as entries are added or deleted, and a quick random access is required, a hash table mechanism is used to reference the table entries. In addition because fast access is needed for generic locates and ordered lists of entries, a getnext chain with a range table is used.

Hash table

The hash table is a set of pointers that are the addresses of directory elements of table entries. A directory element is a set of pointers; one of these pointers is the address of the table entry, the remaining pointers are the addresses of the next elements of various chains used in the different operations of the table manager. An example of a hash table is shown in Figure 88 on page 1143.

The table manager logically combines the characters of the name of the resource, and transforms the result to give an integer that is evenly distributed over the hash table size.

When an entry is located or added, the table manager places it at the head of its chain. Thus frequently used entries tend to have the minimum search times.

If the hash chains become very long, the table manager creates a larger hash table if storage is available. The hash table is enqueued before and dequeued after the reorganization, so that no references to the table can be made during reorganization.

Note: Certain TMP hash tables are not reorganized because they are also used in VTAM SRB exits.

Range table and getnext chain

Some requests to TMP are not full key locates, but rather generic locates with a partial key. For example, requests to find all terminals whose Termid starts with two specified characters. To enable these requests, a getnext chain is maintained which orders all the directory elements alphabetically by key. There is also a ‘range table’ which holds pointers to certain elements along the getnext chain and a count of how many intermediate elements there are in each range.

This range table is hunted with a binary search to find the range in which a given key (full or partial) will reside, and then the getnext chain is used to find a match (if one exists) for the search condition.
Table manager

A range will be split into two equal ranges if the number of intermediate elements rises above a threshold which depends on the number of ranges and the number of elements in the table. So the ranges are dynamic, and do not depend on any particular key distribution.

The number of ranges in the table is determined when the hash table is created, and if all the ranges are full, but a range should be split, a reorganization of the ranges takes place, which increases the range threshold by a factor of 2.

Secondary indexes

A separate hash table, called the secondary index, is created for certain TMP tables, which allows the same entry to be located by another key. In certain secondary indexes, the names do not need to be unique (whereas in the primary index the name is always unique). The secondary index entry is deleted at the same time the entry in the primary index is deleted.

For example, a secondary index is created for DSNAME blocks. This allows table entries to be accessed via secondary keys, using the DSNAME block number in the case of DSNAME blocks.
Certain tables also have aliases as distinct from secondary indexes. These are alternative names for the table entry, which can be used to locate a table entry. They exist in the same index as the primary name, and are not included in a getnext chain, rather they form an alias chain from the primary entry.

Figure 88. Example of a hash table
Functions of the table manager

The table manager performs the following functions:

**Locate table entry**
For a given name, find the address of the table entry.

**Get next table entry**
For a given name, find the address of the next table entry in collating sequence. This can be used repeatedly to find all entries in a range (or all elements in the whole table).

**Add table entry**
For a given table entry, add it into the table.

**Quiesce a table entry**
For a given name, mark its directory segment as busy.

**Unquiesce a table entry**
For a given name, remove its directory segment from the ‘quiesce’ state.

**Delete a table entry**
For a given name, delete it and any associated alias. The entry must have been quiesced first.

**Create an index for a table**
Create a hash table of a given type.

**Add a name into a secondary index**
Given a primary name and a secondary name, add the names to the secondary index.

**Add an alias name**
For a given name, assign an alias name.

**Get next alias name**
For a given a name, find the next alias name (if any).

**Lock a table entry**
For a given a name, assign a read lock to it.

**Unlock a directory entry**
For a given a name, remove the associated read lock.

**Reset lock slots**
For a given name, reset the lock slots.

**Transfer lock to target task**
For a given a name and the address of a target TCA, transfer the read lock to the target task.

**Process deferred work element**
Make the changes made by the logical unit of work (LUW) visible at task syncpoint time.

Read locks

Read locks are used to prevent a table entry being deleted by the table manager.

A read lock is a fullword of storage. When DFHKCP attaches a task, it allocates storage for a number of local read locks; this storage is addressed by TCATMRLP in the TCA. Local read locks are not acquired for table entries that cannot be deleted.
Global read locks are used by the CICS modules that are executed independently of any task. They reside in the table manager static storage area (TMS) that is addressed by SSATMP in the static storage address list (SSA).

These locks are released by:
- an Unlock call,
- a Getnext call,
- a Reset call,
- the termination of the task,
- or a DWE call.

Read locks are always obtained against the primary index entry even if the request is against a secondary index or an alias.

**Browse token**

For Getnext requests on secondary indexes, a browse token is used to hold the name of the previously found entry. The token consists of the name found in the secondary index (which may not be unique) and the name in the primary index (which is unique).

The address of the directory entry cannot be used instead of this logical name because the entry may be returned unlocked, and so may be deleted when the next getnext request is received.

The getnext consists of locating the entry in the secondary index which has the correct primary index, if it exists, and then moving forward in the getnext chain. If it does not, an entry with a matching secondary index name, but a higher primary index name is located, if one exists. If that also does not exist, an entry with a higher name in the secondary index is located. This requires that entries on the getnext chain for ordered both by secondary index name and also when identical secondary index names exist, by primary index name.

**Quiesce state**

A table entry is moved into quiesce state by a quiesce request if no read locks (including ones obtained by the issuing task) exist for the entry. When a table entry moves into quiesced state, it is unable to be located. Locating tasks can choose to ignore or wait for quiesced entries to be unquiesced or deleted.

If the quiesce request is performed with the commit option, the only ways to release the quiesced state are:
- Unquiesce
- Delete

For commit requests, the delete takes place immediately the request completes. Otherwise, if an entry is not deleted or unquiesced by the end of the UOW the TM DWE will unquiesce the entry. In this case, a delete does not take effect until the end of the UOW.

**Finding FCT, or TCT entries in a partition dump**

Figure 89 on page 1153 shows the relationship of the table manager control blocks. A general procedure for finding the required table entries in a partition dump is as follows:

1. Find the CSA.
2. Find the CSA optional features list, CSAOPFL, from its address in field CSAOPFLA (offset X'C8') in the CSA.
3. Find the static storage area address list (SSA) from its address in field CSASSA (offset X'1C0') in the CSAOPFL.
4. Find the table manager static storage area (TMS) from its address in field SSATMP (offset X'14') in the SSA.
5. Look at TMS in the CICS Data Areas manual. The fields TMASKT1 through TMASKT24 hold the addresses of the hash tables for various control blocks. Find the hash table for the control block you are interested in:
   - TMASKT1 = reserved
   - TMASKT2 = reserved
   - TMASKT3 = reserved
   - TMASKT4 = addr of profile table (PFT) entries
   - TMASKT5 = addr of FCT entries
   - TMASKT7 = addr of local terminal (TCTE) entries
   - TMASKT8 = addr of remote terminal and connection (TCNT) entries
   - TMASKT9 = addr of local connection (TCTS) entries
   - TMASKT10 = addr of AFCT entries
   - TMASKT11 = addr of DNAME entries (by name)
   - TMASKT12 = addr of DNAME entries (by block ID)*
   - TMASKT13 = addr of partner resource table (PRT) entries
   - TMASKT14 = reserved
   - TMASKT15 = addr of local terminal NETNAME table (TCNT) entries
   - TMASKT16 = addr of autoinstall terminal model (AITM) table entries
   - TMASKT17 = addr of signon table (SNT) entries
   - TMASKT18 = addr of session (TCSE) entries
   - TMASKT19 = addr of remote connection entries (TCSR)*
   - TMASKT20 = addr of indirect connection entries (TCSI)*
   - TMASKT21 = addr of connection NETNAME (TCSN) entries*
   - TMASKT22 = addr of remote terminal entries (TCTR)*
   - TMASKT23 = addr of generic connection NETNAME (TCSM) entries*
   - TMASKT24 = addr of remote terminal NETNAME (TCNR) entries*
   
   * = Secondary index

   Use the following formula to find the offset of the individual scatter table:
   \[ \text{Length(TMATTV)} \times (n-1) + \text{X'08'} \]

   Where n = position in table (see above - TMASKTn)

   To find Length(TMATTV) (and the value of n) see the CICS Data Areas manual.
6. Find the first directory element from its address in field SKTFDEA (offset X'10') in the hash table area.
7. Directory elements are chained together in alphabetic order. The address of the next element is in field DIRGNCHN (offset X'10').
8. Look at each directory element until you find the name of the control block you are looking for. The name is in field DIRKEY (offset X'18'). Field DIRTEA (offset X'0') holds the address of the desired control block.

**Control blocks**

Figure 89 on page 1153 shows the table manager control blocks.
See the CICS Data Areas manual for a detailed description of these control blocks.

### Modules

DFHTMP

### Exits

No global user exit points are provided for this function.

### Trace

The following point ID is provided for this function:
- AP 00EA, for which the trace level is AP 1.

See the CICS Trace Entries for further information.
The statistics utility program, DFHSTUP, provides, for table management, statistics (for each table) on the amount of storage (expressed in bytes) used by the table manager to support each table (excluding storage used for the tables themselves).

For more information about the trace points, see the CICS Trace Entries. For more information about using traces in problem determination, see the CICS Problem Determination Guide.
Chapter 89. Task-related user exit control

Task-related user exit support in CICS, also known as the resource manager interface (RMI), provides an interface that non-CICS resource managers can use to communicate with CICS applications. The exit program can be enabled or disabled dynamically, and useful information can be transferred to a user work area.

Functional overview

The following operations may be performed on a task-related user exit from application programs:

**ENABLE**

This is a global operation that names the task-related user exit and causes the task-related user exit to be loaded into storage, if it has not already been loaded. It also causes the exit program control block (EPB), which represents the task-related user exit, and the exit’s global storage to be set up by the user exit manager module, DFHUEM. The EPB also holds a TALENGTH argument and a bit-string profile for use in an exit operation. The ENABLE operation does not pass control to the task-related user exit.

DFHUEM is used to enable both global user exits and task-related user exits.

The ENABLE operation is performed in two stages:

1. **ENABLE**
2. **START**.

An exit is not made available for execution until it has been both enabled and started.

You can use the TASKSTART keyword on the ENABLE command to enable a task-related user exit so that it is invoked at task start for all tasks in the CICS system.

You can also enable a task-related user exit with the FORMATEDF keyword, which means that the task-related user exit can provide formatted screens for EDF to display, whenever a DFHRMCAL request to the task-related user exit takes place.

The task-related user exit is invoked in the addressing mode of its original caller unless the LINKEDITMODE keyword is specified on the ENABLE command, in which case the exit is invoked in its own link-edit AMODE. LINKEDITMODE is only valid on the first ENABLE command for an exit program.

**EXTRACT**

Information concerning an “enabled and started” task-related user exit is returned to an application when it issues this command.

**DISABLE**

This is a global operation which in general terms is the reverse of an ENABLE request. The DISABLE operation can be performed in two stages:

1. **STOP**: This is the reverse of the START keyword on the ENABLE request. It causes the task-related user exit to remain in main storage.
Task-related user exit control

together with all its associated control blocks; however it is not available for execution until an ENABLE command with the START option is specified.

2. EXITALL: This causes the EXIT and its control blocks to be deleted from main storage. The EPB however is added to a chain of re-usable EPB’s anchored in the UETH.

DFHRMCAL
After an exit has been enabled and started, it can be invoked from an application using a DFHRMCAL request directly, or by passing control to a stub which performs the DFHRMCAL request. A register 1 parameter list may be supplied to the task-related user exit from the application.

The task interface element (TIE) control block is created for the task and task-related user exit combination when the task issues its first DFHRMCAL request, unless the TIE has already been created because the task-related user exit was enabled for TASKSTART.

When a DFHRMCAL request is issued, control passes to DFHEIP, to DFHERM (the external resource manager interface program), and then to the task-related user exit. DFHERM manages the TIEs.

ENABLE, DISABLE, and EXTRACT are all EXEC CICS requests. DFHRMCAL is a macro.

A task-related user exit can “express interest” in certain types of events, and be invoked when these events take place. These events are:

• Application invocations (DFHRMCAL mentioned above), associated with which are optionally the EDF screen format invocations
• System Programming interface events i.e. INQUIRE EXITPROGRAM commands
• Syncpoint related events
• Task termination events
• CICS termination.

By default, it is assumed that task-related user exits are interested in application invocations only.

Design overview

The task-related user exit interface is comparable with the EXEC interface. When an application program requests the services of a non-CICS resource manager, it does so by a module called the task-related user exit. The exit receives arguments from the application program, and passes them on to the resource manager in a suitable form.

The advantage of this method is that if the resource manager is changed, the application program that invokes the resource manager should not need to be changed too.

The exit is part of the resource manager programs. The name of the exit, or the name of the entry to the exit, is specified by the resource manager, and each application program that invokes the resource manager has to be link-edited with an application program stub that refers to that name.
The exit is enabled and disabled using the user exit manager (DFHUEM). For enabling, the resource manager can specify the size of a task-related work area that it requires.

The exit, when enabled and subsequently driven, receives arguments in the form specified by the DFHUEXIT TYPE=RM parameter list (see the CICS Customization Guide or the manual). Register 1 points to this parameter list. Register 13 points to the address of a save area, rather than the address of the CSA. The save area is 18 words long, with registers 14 through 12 stored in the fourth word onward.

Responses to the request are indicated by values placed in register 15, and also by means that are specific to the architecture of the application interface, for example, by moving data into storage areas passed by the call, or into the caller’s register 15.

The main control blocks used by the task-related interface are the task interface element (TIE):

- A TIE is created by DFHERM on the first call by a task to each resource manager, and it is chained to the TCA for that task.

**Task-related user exit implementation**

The state of an exit is managed by DFHUEM, which is described under “Chapter 105. User exit control” on page 1359. For an exit, the TALENGTH argument and a profile in the form of a bit-string are held in the exit program block (EPB). These arguments are not processed until the occurrence of an application program CALL that explicitly names the exit, unless the TASKSTART keyword is used on the ENABLE request.

Entry to the exit is through the task-related user exit interface, which comprises:

- An application stub provided with the exit, but generated using the CICS-provided macro DFHRMCAL. It is this stub which explicitly names the exit, and which is link-edited with each application program that uses the application program interface (API) of the resource manager.
- DFHEIP, which is entered at DFHEIPCN by the application stub, in much the same way as EXEC CICS commands are routed at execution time.
- DFHERM, which receives control when DFHEIP discovers that the call is not for a CICS control function, but for a named exit.

DFHERM receives a set of registers (those of the caller, for example, the application program), and a routing argument which names the exit. This routing argument is constructed by DFHRMCAL, in the application stub, and is not normally visible to the application programmer. DFHERM retrieves the name of the requested exit from the routing argument, and scans any existing task interface elements (TIEs) that are chained from the task’s TCA, looking for a TIE associated with the named exit. If such a TIE is not found, it searches the installed exits on a chain of EPBs, looking for the matching name. On finding a match, DFHERM constructs a TIE to represent the connection between that task and the exit. The TIE is initialized from information provided in the EPB; the TALENGTH argument defines the size of a task-local work area which can be thought of as a logical extension of the TIE. The profile string is also copied into the TIE.

DFHERM stacks (stores in a last-in, first-out manner) various parts of the program execution environment—the status of HANDLE commands, file browse cursors, the EXEC interface block (EIB), and so on—and builds a parameter structure which
is essentially a superset of that built by DFHUEH. Additional arguments include the task-local work area, the profile referred to above, and an 8-byte UOW identifier supplied by Recovery Manager.

DFHERM then passes control to the exit’s entry point using standard CALL conventions, in which register 13 addresses a save area for DFHERM’s own registers, register 14 addresses DFHERM’s next sequential instruction, and register 1 addresses the passed parameters. This is a vector of addresses which include that of the caller’s register save area. Any changes the exit makes to arguments of the application program interface (API), or to the contents of the caller’s register save area, are not examined by DFHERM when it regains control, because they are not part of the CICS task-related user exit interface—rather they are the concern of the caller and the exit. However, the exit can request DFHERM to schedule certain actions by means of the profile argument. For example, the exit can request that it be informed (driven) when commitment of resources (syncpointing) is taking place, or the exit can request that DFHERM no longer routes API calls to it from this task.

Finally, on regaining control from the exit, DFHERM unstacks the objects that it had previously stacked, and returns to the caller. The state of the cursors, HANDLE labels, and so on, is apparently unchanged by the actions of DFHERM or the exit. Note that the exit may have used EXEC CICS HANDLE commands; this does not interfere with the caller’s HANDLE status.

In the discussion of DFHERM so far, the term “caller” has been used for the application program. However, a caller can be a function such as syncpoint (DFHERMSp), task control (DFHAPXM or DFHERMSp), system programming interface (DFHUEIQ), CICS termination (DFHAPDM or DFHSTP) or EDF (DFHERM). The exit can set appropriate bits in the profile (schedule flag word) so that, if the corresponding function is subsequently invoked, it in turn calls the exit. The exit can determine the identity of the caller from the first argument (called the “function definition”). This argument, passed by DFHERM, always has its first byte equal to X’00’. (If the first byte is other than X’00’, the exit has been entered from DFHUEH as a global user exit.) DFHERM sets the second byte of this argument according to the type of caller, thus indicating which interface is addressed by the caller’s register save area. The second byte is:

X’01′  For system programming interface
X’02′  For an application program
X’04′  For the syncpoint program
X’08′  For CICS task control
X’0A′  For a CICS termination call
X’0C′  For an EDF call.

Any remaining arguments are specific to each individual caller.

The flow of control for the task-related user exit interface is shown in Figure 90 on page 1153.
Processors

The term “processor” is used to refer to two different types of object:

1. For the EXEC interface, it refers to the function-dependent modules associated with the EXEC interface nucleus, DFHEIP. These processors usually have names such as DFHEPC, DFHETC, DFHETD, and so on, and each of these is invoked by DFHEIP. DFHERM is also a processor of this type.

2. In various contexts, including task-related user exits, it refers to a piece of code that is link-edited with an application program and serves the dual function of:
   - Satisfying the CALL requirement for a target address—its entry resolves a V-type ADCON
   - Finding the entry point of DFHEIP.

Both these types of processor are part of the path between an application call and the functional control module that supports the request. This path appears as follows:

Application call
  Application processor (type 2)
  DFHEIP
  EXEC interface processor (type 1)
  Functional control module

Examples of the interface are:

EXEC CICS SYNCPOINT ... CICS API
DFHECI  CICS COBOL EIP router
DFHEIP
DFHEISP  CICS syncpoint router
DFHSPP  CICS syncpoint manager
      CICS Recovery manager domain

EXEC DLI TERM ... DLI HLPI
DFHECI  CICS COBOL EIP router
DFHEIP

Figure 90. Task-related user exit control flow
Task-related user exit control

DFHERM CICS RMI module
DFHEDP DLI HLPI manager
(implemented as a task-related user exit)

Control blocks

The control blocks used in task-related user exit control are the exit program control block (DFHEPB), the task interface element (DFHTIEDS).

Figure 91 shows the main control blocks associated with task-related user exits.

![Figure 91. Control blocks associated with task-related user exits](image)

Field CSAUETBA in the CSA points to the user exit table (UET); UETHEPBC in the UET points to the first exit program block (EPB); and EPBCHAIN in each EPB points to the next EPB in the chain.

Each EPB holds:
- The address of the exit’s entry point (EPBEPN)
- The address of the global work area
- The halfword length of the global work area
- The halfword length of the task-local work area.

One EPB is associated with each enabled task-related user exit program or entry name.

EPBs used for global user exits and for task-related user exits are held on the same EPB chain.

The task-related user exit’s global storage is optional. It is associated with an individual enabled task-related user exit program or entry name. Several task-related user exit programs or entry names can share the same global storage.
Task-related user exit control

For full details of the EPB, see the CICS Data Areas manual.

The task interface element (TIE) is associated with each associated pair of CICS task and task-related user exit. The first time a CICS task passes control to a particular task-related user exit, a TIE is created. The TIE lasts until task termination.

Note that all TIEs relating to a single task are chained together (more than one TIE is set up when a single CICS task makes use of more than one task-related user exit). The TIEs corresponding to a single EPB (that is, to a single task-related user exit program or entry name) are not chained together.

A global user exit may only use global storage; a task-related user exit may use both global storage and task-local work area.

Field TCATIEBA in the TCA points to the first TIE, and TIECHNA in each TIE points to the next TIE in the chain.

The TIE holds information relevant to all invocations of the task-related user exit for the task concerned. For example, TIEFLAGS holds information concerning the events for which the task-related user exit should be invoked, for example, API calls, syncpoint, and task start.

Figure 92 gives a closer look at the TIE control block chain that is used during the lifetime of a task-related user exit.

Figure 92. Control blocks used during the lifetime of a task-related user exit
Task-related user exit control

For full details of the TIE control blocks, see the CICS Data Areas manual.

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHUEM</td>
<td>The EXEC interface processor for the ENABLE, DISABLE, and EXTRACT user exit commands.</td>
</tr>
<tr>
<td>DFHERM</td>
<td>Interfaces with task-related user exit.</td>
</tr>
<tr>
<td>DFHTIEM</td>
<td>Handles the TIE subpools.</td>
</tr>
</tbody>
</table>

Exits

No global user exit points are provided for this function.

Trace

The following point ID is provided for this function:
- AP 2520 ) for which the trace level is RI 1.
- AP 2521 )
- AP 2522 ) for which the trace level is RI 2.
- AP 2523 )

For more information about the trace points, see the CICS Trace Entries. For more information about using traces in problem determination, see the CICS Problem Determination Guide.

External interfaces

Calls are made to the task-related user exit via DFHEIP and DFHERM from the following modules:
- DFHAPXM: Task start
- DFHERMSP: Task end
- DFHERMSP: Syncpoint and backout
- DFHRMSY: For syncpoint resynchronization
- DFHAPDM: CICS termination
- DFHSTP: CICS termination
- DFHUEIQ: System programming interface for inquire exitprogram calls

Applications
- Application calls to resource manager
- DFHERM: EDF invocations for application calls to resource manager
Chapter 90. Task-related user exit recovery

Task-related user exit recovery, also known as the resource manager interface (RMI) recovery, ensures that changes to recoverable resources performed by an external resource manager in a logical unit of work are either all committed or all backed out.

Design overview

During the execution of a CICS task, the CICS recovery manager communicates with the resource manager task-related user exit to prepare to commit, to commit unconditionally, or to back out. The purpose of these calls is to ensure that changes to recoverable resources performed in a unit of work (UOW) are either all committed or all backed out, if there is a failure anywhere in CICS or in any of the external resource managers.

Each UOW created by Recovery Manager Domain is identified by a UOW_ID and a Local UOW_ID. The LOCAL UOWID is an eight byte value whose format is easy for CICS to identify whether the UOW originated before or after an initial start.

When the resource manager receives the call to commit unconditionally or to back out, it takes the corresponding irreversible step, if possible. If the action is successful, the resource manager sends the appropriate return code. If not, it sends a return code which requests that CICS record the state of the UOW, and tries to resolve the status at a later time.

Recovery manager domain maintains the status of UOWs that require resynchronization, until all participants in the UOW have successfully resynchronized. Recovery manager domain maintains these UOWs across cold, warm and emergency start of CICS. An initial start of CICS however will mean that Recovery manager domain will lose this information and resynchronization will not be possible.

The RMI also supports an optimized syncpoint process to improve performance under certain conditions where a single-phase commit can be used. With single phase commit Recovery manager does not have to maintain resynchronization information for the RMI. This optimized process is described in more detail later in this section.

The two-phase commit process

The RMI supports the two-phase commit process. The following is a brief summary of the two-phase commit process and other related processing as seen from the RMI’s point of view.

- When a unit of work is first created, Recovery manager creates local_uow_id which will be used by the RMI.
- When the task syncpoints, a prepare-to-commit request is then issued to each task-related user exit used during the current UOW. For each task-related user exit, issuing the prepare request indicates the start of phase 1 of commit processing from CICS’s point of view.
Task-related user exit recovery

- If all syncpoint participants vote ‘YES’ to the prepare requests, then Recovery manager will commit the UOW. CICS then invokes each task-related user exit with a commit request. This indicates the start of phase 2 commit processing for the task-related user exit.
  
  If the task-related user exit is unable to commit the UOW, Recovery manager will maintain a record of the UOW’s status so that the task related user exit can resync later.

- If one or more of the task-related user exits votes ‘NO’ to the prepare-to-commit request, all the task’s recoverable resources are backed out.

Resolution of in-doubts

An external resource manager can be left in doubt about the disposition of UOWs, for example, if the resource manager abnormally terminated after receiving a prepare request for an UOW, but before receiving the commit or backout request. The resource manager, at any time while interfacing with CICS, can supply a list of recovery tokens representing the in-doubt UOWs to the task-related user exit. The task-related user exit (or other related code) can then issue an EXEC CICS RESYNC request with the in-doubt list and the name of the task-related user exit as parameters.

As a result of a the EXEC CICS RESYNC command, DFHERMRS initiates a CRSY task (running program DFHRMSY) for each UOW named in the indoubt list passed from the TRUE. DFHRMSY interfaces with Recovery manager to find out the status of the UOW, and calls the task-related user exit with the appropriate resolution, for example ‘Commit’, ‘Backout’ and so on. For each successful commit or backout, DFHRMSY informs Recovery manager that it can delete the TRUEs involvement in the UOW. When all interested parties in a UOW complete such processing, Recovery manager deletes its record of the UOW.

If an EXEC CICS RESYNC request is issued without an in-doubt list or with an in-doubt list of length zero, then DFHERMRS informs Recovery manager that it can remove the TRUE (identified by its name and qualifier) from all UOWs in the resynchronization set, i.e. delete all resync information for a TRUE.

A resynchronization set is first established when a TRUE is enabled. The next resynchronization set is identified on completion of an EXEC CICS RESYNC command, and is used for the next RESYNC command. A resynchronization bounds how many UOWs resync information is deleted for because RESUNC commands execute at the same time as new work is processed by a TRUE. A RESYNC command with a zero list should not delete resync information new UOW created since the resync command was issued.

The single-phase commit process

The RMI also supports the single-phase commit process for UOWs that are read-only, and for UOWs where CICS detects that only one external resource manager has been called for update requests. The task-related exit must indicate to the RMI that it is capable of processing single-phase commit requests; otherwise, a two-phase commit is used. Use of single-phase commit improves performance, because CICS does not perform any logging and the task-related user exit is called only once during syncpoint processing.

Single-phase commit for read-only UOWs

To take advantage of single-phase commit for read-only UOWs, the external resource manager must return to the task-related user exit an indicator that the UOW is read-only. This can be done by the resource manager returning a flag.
indicating the “history” of the UOW so far (that is, whether it is read-only so far), or returning information about the current request. In the latter case, it is the responsibility of the task-related user exit to keep a “history” of the UOW so far. After each request, the task-related user exit must return to CICS with a flag set in the parameter list indicating this history.

At syncpoint time, if CICS detects that the UOW is read-only, it invokes the task-related user exit with an “End-UOW” request instead of the normal prepare and commit requests associated with a two-phase commit. This means that the task-related user exit is invoked only once during syncpoint. The “End-UOW” request is issued during phase 2 syncpoint processing. On receiving an “End-UOW” request, the task-related user exit should invoke the resource manager for single-phase commit. There are no return codes associated with the “End-UOW” request, and CICS does not perform any logging for this type of request.

**Single-phase commit for the single updater**

To take advantage of single-phase commit for the single-update situation, the task-related user exit must indicate to the RMI that it knows the single-update protocol. It does this by setting a flag in the parameter list at the same time as it expresses an interest in syncpoint.

At syncpoint time, if CICS detects that only resources owned by one external resource manager were updated in the UOW, and if the task-related user exit has indicated that it understands the protocol, CICS invokes the task-related user exit with an ‘Only’ request, instead of the normal prepare and commit requests associated with a two-phase commit. This means that the task-related user exit is invoked only once during syncpoint. The ‘Only’ request is issued during phase 1 syncpoint processing. CICS does not perform any logging for this type of request. When invoked for an ‘Only’ request, the task-related user exit should invoke the resource manager for single-phase commit.

There are two architected responses to the ‘Only’ request: ‘OK’ and ‘Backed-out’. ‘OK’ means that the UOW was committed; ‘Backed-out’ means that the single-phase commit failed and the updates were backed out. It is important to note that, unlike the two-phase commit, there is no equivalent ‘Remember’ response. If a task-related user exit calls a resource manager for single-phase commit and, for example, the resource manager abends while processing this request, the task-related user exit is left in doubt as to the outcome of the request. The task-related user exit cannot return to CICS in this case, but instead must output diagnostic messages as appropriate, and then abend the transaction.

Recovery manager does not keep resynchronization information for UOWs using single phase commit. Because the resource manager is the only updater in the UOW, CICS is not in doubt about any of its resources. The resource manager has either committed or backed out the updates. The messages output by the task-related user exit, in conjunction with any messages output by the resource manager, can be used to determine the outcome of the UOW.
Task-related user exit recovery

Modules

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHERMRS</td>
<td>DFHERMRS is invoked by DFHEISP as a result of an EXEC CICS RESYNC command. It attaches a CRSY task for each UOW identified in the IDLIST. Calls Recovery manager to delete unwanted resynchronization information.</td>
</tr>
<tr>
<td>DFHRMSY</td>
<td>A CRSY task (running program DFHRMSY) is attached for each indoubt UOW appearing in the in-doubt list for an EXEC CICS RESYNC command. This program then issues the appropriate ‘phase 2 of syncpoint’ request, that is, commit or backout, to the external resource manager that issued the EXEC CICS RESYNC.</td>
</tr>
</tbody>
</table>

Exits

No global user exit points are provided for this function.

Trace

The following point IDs are provided for this function:
- AP 2540 ) For trace level RI Level 1
- AP 2541 )
- AP 2548 ) For trace level RI level 2
- AP 2549 )
- AP 2560 ) For trace level RI level 1
- AP 2561 )

For more information about the trace points, see the CICS Trace Entries. For more information about using traces in problem determination, see the CICS Problem Determination Guide.

External interfaces

Calls are made from DFHRMSY, via DFHERM, to the task-related user exit to provide information about the disposition of the UOW, when resynchronization of in-doubts is taking place.
Chapter 91. Temporary storage domain (TS)

The temporary storage domain manages temporary storage requests.

## Temporary storage domain’s specific gates

Table 104 summarizes the temporary storage domain’s specific gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and whether or not the functions are available through the exit programming interface (XPI).

### Table 104. Temporary storage domain’s specific gates

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>XPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSQR</td>
<td>TS 0201</td>
<td>WRITE</td>
<td>NO</td>
</tr>
<tr>
<td>TSQR</td>
<td>TS 0202</td>
<td>REWRITE</td>
<td>NO</td>
</tr>
<tr>
<td>TSQR</td>
<td>TS 0202</td>
<td>READ_INTO</td>
<td>NO</td>
</tr>
<tr>
<td>TSQR</td>
<td>TS 0202</td>
<td>READ_SET</td>
<td>NO</td>
</tr>
<tr>
<td>TSQR</td>
<td>TS 0202</td>
<td>READ_NEXT_INTO</td>
<td>NO</td>
</tr>
<tr>
<td>TSQR</td>
<td>TS 0202</td>
<td>READ_NEXT_SET</td>
<td>NO</td>
</tr>
<tr>
<td>TSQR</td>
<td>TS 0202</td>
<td>DELETE</td>
<td>NO</td>
</tr>
<tr>
<td>TSPT</td>
<td>TS 0301</td>
<td>PUT</td>
<td>NO</td>
</tr>
<tr>
<td>TSPT</td>
<td>TS 0302</td>
<td>PUT_REPLACE</td>
<td>NO</td>
</tr>
<tr>
<td>TSPT</td>
<td>TS 0302</td>
<td>GET</td>
<td>NO</td>
</tr>
<tr>
<td>TSPT</td>
<td>TS 0302</td>
<td>GET_SET</td>
<td>NO</td>
</tr>
<tr>
<td>TSPT</td>
<td>TS 0302</td>
<td>GET_RELEASE</td>
<td>NO</td>
</tr>
<tr>
<td>TSPT</td>
<td>TS 0302</td>
<td>GET_RELEASE_SET</td>
<td>NO</td>
</tr>
<tr>
<td>TSSH</td>
<td>TS 0A01</td>
<td>INITIALIZE</td>
<td>NO</td>
</tr>
<tr>
<td>TSSH</td>
<td>TS 0A01</td>
<td>INQUIRE_POOL_TOKEN</td>
<td>NO</td>
</tr>
<tr>
<td>TSSH</td>
<td>TS 0A01</td>
<td>WRITE</td>
<td>NO</td>
</tr>
<tr>
<td>TSSH</td>
<td>TS 0A01</td>
<td>REWRITE</td>
<td>NO</td>
</tr>
<tr>
<td>TSSH</td>
<td>TS 0A01</td>
<td>READ_INTO</td>
<td>NO</td>
</tr>
<tr>
<td>TSSH</td>
<td>TS 0A01</td>
<td>READ_SET</td>
<td>NO</td>
</tr>
<tr>
<td>TSSH</td>
<td>TS 0A01</td>
<td>READ_NEXT_INTO</td>
<td>NO</td>
</tr>
<tr>
<td>TSSH</td>
<td>TS 0A01</td>
<td>READ_NEXT_SET</td>
<td>NO</td>
</tr>
<tr>
<td>TSSH</td>
<td>TS 0A01</td>
<td>DELETE</td>
<td>NO</td>
</tr>
<tr>
<td>TSSH</td>
<td>TS 0A01</td>
<td>INQUIRE_SYSID_TABLE_TOKEN</td>
<td>NO</td>
</tr>
<tr>
<td>TSSH</td>
<td>TS 0A01</td>
<td>START_BROWSE</td>
<td>NO</td>
</tr>
<tr>
<td>TSSH</td>
<td>TS 0A01</td>
<td>GET_NEXT</td>
<td>NO</td>
</tr>
<tr>
<td>TSSH</td>
<td>TS 0A01</td>
<td>END_BROWSE</td>
<td>NO</td>
</tr>
<tr>
<td>TSSH</td>
<td>TS 0A01</td>
<td>INQUIRE_QUEUE</td>
<td>NO</td>
</tr>
<tr>
<td>TSSR</td>
<td>TS 0601</td>
<td>SET_START_TYPE</td>
<td>NO</td>
</tr>
<tr>
<td>TSSR</td>
<td>TS 0602</td>
<td>SET_BUFFERS</td>
<td>NO</td>
</tr>
<tr>
<td>TSSR</td>
<td>TS 0602</td>
<td>SET_STRINGS</td>
<td>NO</td>
</tr>
<tr>
<td>TSBR</td>
<td>TS 0701</td>
<td>INQUIRE_QUEUE</td>
<td>NO</td>
</tr>
<tr>
<td>TSBR</td>
<td>TS 0702</td>
<td>START_BROWSE</td>
<td>NO</td>
</tr>
<tr>
<td>TSBR</td>
<td>TS 0702</td>
<td>GET_NEXT</td>
<td>NO</td>
</tr>
<tr>
<td>TSBR</td>
<td>TS 0702</td>
<td>END_BROWSE</td>
<td>NO</td>
</tr>
<tr>
<td>TSBR</td>
<td>TS 0702</td>
<td>CHECK_PREFIX</td>
<td>NO</td>
</tr>
</tbody>
</table>
Temporary storage domain (TS)

TSQR gate, WRITE function

If the queue does not exist, this function creates a queue with the single item provided, and the queue’s “read cursor” is set to zero.

If the queue already exists, the item provided is appended to the queue, and the read cursor left unchanged.

Input parameters

QUEUE_NAME
is the name of the queue being created or appended to.

ITEM_DATA
is the address and length of the item being written.

[BMS] indicates whether or not BMS owns this queue. It can have either of these values:
YES|NO

SUSPEND
indicates whether or not the request will be suspended if there is insufficient auxiliary storage to satisfy the request. This option is ignored if the queue is in main storage.

STORAGE_TYPE
indicates whether the queue is to be created in main or auxiliary storage. Note that this option is ignored if the queue already exists.

[CALLER]
indicates whether this request originated from an EXEC or macro call. The default is MACRO. It can have either of these values:
EXEC|MACRO

[FMH]
indicates whether the data contains an FMH. It can have either of these values:
YES|NO

Output parameters

[TOTAL_ITEMS]
is the total number of items in the queue on completion of the operation.

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>QUEUE_FULL, INSUFFICIENT_STORAGE, INVALID_LENGTH, IO_ERROR,</td>
</tr>
<tr>
<td></td>
<td>INVALID_QUEUE_TYPE, LOCKED, INVALID_QUEUE_NAME, QUEUE_DELETED,</td>
</tr>
<tr>
<td></td>
<td>QUEUE_REMOTE</td>
</tr>
</tbody>
</table>

TSQR gate, REWRITE function

This function updates the specified item in an existing queue. The read cursor is unchanged.
Input parameters

QUEUE_NAME
is the name of the queue being updated.

ITEM_NUMBER
is the number of the item to be updated.

ITEM_DATA
is the address and length of the item being written.

SUSPEND
indicates whether the request will be suspended if there is insufficient auxiliary storage to satisfy the request. This option is ignored if the queue is in main storage.

[CALLER]
indicates whether this request originated from an EXEC or macro call. The default is MACRO. It can have either of these values:
EXEC|MACRO

[FMH]
indicates whether the data contains an FMH. It can have either of these values:
YES|NO

Output parameters

[TOTAL_ITEMS]
is the total number of items in the queue.

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>INSUFFICIENT_STORAGE, INVALID_LENGTH, IO_ERROR, INVALID_QUEUE_TYPE, QUEUE_NOT_FOUND, ITEM_NOT_FOUND, LOCKED, INVALID_QUEUE_NAME, QUEUE_DELETED, QUEUE_REMOTE</td>
</tr>
</tbody>
</table>

TSQR gate, READ_INTO function

This function reads the specified queue item into a buffer provided by the caller. The read cursor for the queue is set to the item number provided. The caller provides the address (item_buffer_p) and buffer length (item_buffer_m). The actual length of the record is returned in item_buffer_n. If item_buffer_n is greater than item_buffer_m, the data is truncated (but an OK response is returned).

Input parameters

QUEUE_NAME
is the name of the queue being read.

ITEM_NUMBER
is the number of the item to be read.
**ITEM_BUFFER**
specifies the address (item_buffer_p) and maximum length (item_buffer_m) of the data area into which the data will be read. The actual data length is returned in item_buffer_n.

**[CALLER]**
indicates whether this request originated from an EXEC or macro call. The default is MACRO. It can have either of these values:
EXEC|MACRO

**Output parameters**

**[TOTAL_ITEMS]**
returns the total number of items in the queue.

**[FMH]**
indicates whether the data contains an FMH. It can have either of these values:
YES|NO

**RESPONSE**
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**[REASON]**
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>IO_ERROR, INVALID_QUEUE_TYPE, QUEUE_NOT_FOUND, ITEM_NOT_FOUND, INVALID_QUEUE_NAME</td>
</tr>
</tbody>
</table>

**TSQR gate, READ_SET function**

This function reads the specified queue item into a storage area obtained by TS. The read cursor for the queue is set to the input item number.

**Input parameters**

**QUEUE_NAME**
is the name of the queue being read.

**ITEM_NUMBER**
is the number of the item to be read.

**[TCTTE_ADDRESS]**
is the address of the TCTTE - required if
SET_STORAGE_CLASS(TERMINAL) is specified.

**[SET_STORAGE_CLASS]**
specifies the class of storage into which the item will be read. This may be either TASK (the default) or TERMINAL. If TERMINAL is specified, the item is read into a TIOA. It can have either of these values:
TASK|TERMINAL

**[CALLER]**
indicates whether this request originated from an EXEC or macro call. The default is MACRO. It can have either of these values:
EXEC|MACRO
Output parameters

ITEM_DATA
returns the address and length of the item data.

[TOTAL_ITEMS]
returns the total number of items in the queue.

[FMH]
indicates whether the data contains an FMH. It can have either of these values:
YES|NO

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>IO_ERROR, INVALID_QUEUE_TYPE, QUEUE_NOT_FOUND, ITEM_NOT_FOUND, INVALID_QUEUE_NAME</td>
</tr>
</tbody>
</table>

TSQR gate, READ_NEXT_INTO function

This function increments the read cursor by one and reads that item number into the buffer provided by the caller. The caller provides the address (item_buffer_p) and buffer length (item_buffer_m). The actual length of the record is returned in item_buffer_n. If item_buffer_n is greater than item_buffer_m, the data will have been truncated.

Input parameters

QUEUE_NAME
is the name of the queue being read.

ITEM_BUFFER
specifies the address (item_buffer_p) and maximum length (item_buffer_m) of the data area into which the data will be read. The actual data length is returned in item_buffer_n.

[CALLER]
indicates whether this request originated from an EXEC or macro call. The default is MACRO. It can have either of these values:
EXEC|Macro

ITEM NUMBER
returns the number of the item just read.

Output parameters

[TOTAL_ITEMS]
returns the total number of items in the queue.

[FMH]
indicates whether the data contains an FMH. It can have either of these values:
YES|NO
**Temporary storage domain (TS)**

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**[REASON]**

is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>IO_ERROR, INVALID_QUEUE_TYPE, QUEUE_NOT_FOUND, ITEM_NOT_FOUND, INVALID_QUEUE_NAME</td>
</tr>
</tbody>
</table>

**TSQR gate, READ_NEXT_SET function**

This function increments the queue’s read cursor by one and reads that item number into a storage area obtained by TS.

**Input parameters**

**QUEUE_NAME**

is the name of the queue being read.

**[TCTTE_ADDRESS]**

is the address of the TCTTE - required if SET_STORAGE_CLASS(TERMINAL) is specified.

**[SET_STORAGE_CLASS]**

specifies the type of storage into which the item will be read. This may be either TASK (the default) or TERMINAL. If TERMINAL is specified, the item is read into a TIOA. It can have either of these values:

TASK|TERMINAL

**[CALLER]**

indicates whether this request originated from an EXEC or macro call. The default is MACRO. It can have either of these values:

EXEC|MACRO

**Output parameters**

**ITEM_DATA**

returns the address and length of the item data.

**[ITEM_NUMBER]**

returns the number of the item just read.

**[TOTAL_ITEMS]**

returns the total number of items in the queue.

**[FMH]**

indicates whether the data contains an FMH. It can have either of these values:

YES|NO

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**[REASON]**

is returned when RESPONSE is EXCEPTION. Possible values are:
**TSQR gate, DELETE function**

This function deletes the specified queue.

**Input parameters**

**QUEUE_NAME**

is the name of the queue to be deleted. the request.

**[CALLER]**

indicates whether this request originated from an EXEC or macro call. The default is MACRO. It can have either of these values:

EXEC | MACRO

**RESPONSE**

is the domain's response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

**[REASON]**

is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>INVALID_QUEUE_TYPE, QUEUE_NOT_FOUND, LOCKED, INVALID_QUEUE_NAME, QUEUE_DELETED</td>
</tr>
</tbody>
</table>

---

**TSQR gate, ALLOCATE_SET_STORAGE function**

This function allocates set storage of the requested length.

**Input parameters**

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

**[REASON]**

is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>INVALID_QUEUE_TYPE, QUEUE_NOT_FOUND, LOCKED, INVALID_QUEUE_NAME, QUEUE_DELETED</td>
</tr>
</tbody>
</table>

---

**TSPT gate, PUT function**

If the queue does not already exist, this function creates a queue with the single item provided.

If the queue already exists, and is recoverable, a duplicate_name exception is returned. Otherwise, the item is appended to the queue.

**Input parameters**

**QUEUE_NAME**

is the name of the queue being created or appended to.
**Temporary storage domain (TS)**

**ITEM_DATA**

is the address and length of the item being written.

**[IC_DATA]**

is the address and length of an optional ICE.

**[BMS]**

this option indicates whether or not BMS owns this queue. If the queue already exists and is a BMS queue then BMS(YES) must be specified on the request. Otherwise an INVALID response is returned. It can have either of these values:

YES|NO

**[IC]**

this option indicates whether or not Interval Control owns this queue. If the queue already exists and is an IC queue then IC(YES) must be specified on the request. Otherwise an INVALID response is returned. It can have either of these values:

YES|NO

**[FMH]**

indicates whether the data contains an FMH. It can have either of these values:

YES|NO

**SUSPEND**

indicates whether the request is to be suspended if there is insufficient auxiliary storage to satisfy the request.

**Output parameters**

**RECOVERABLE**

returns whether the queue is recoverable or not.

**QUEUE_CREATION_TIME**

returns the store clock time at which the queue was created.

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**[REASON]**

is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>INSUFFICIENT_STORAGE, QUEUE_FULL, DUPLICATE_NAME, INVALID_LENGTH, IO_ERROR,</td>
</tr>
<tr>
<td></td>
<td>INVALID_QUEUE_TYPE, LOCKED, INVALID_QUEUE_NAME, QUEUE_DELETED, QUEUE_REMOTE</td>
</tr>
</tbody>
</table>

**TSPT gate, PUT_REPLACE function**

If the queue does not exist, this function creates the queue with the item provided. If the queue does exist, the first item in the queue is replaced by the item provided.

**Input parameters**

**QUEUE_NAME**

is the name of the queue being created or written to.

**ITEM_DATA**

is the address and length of the data item being written.
RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>INVALID_LENGTH, IO_ERROR, INVALID_QUEUE_TYPE, LOCKED, INVALID_QUEUE_NAME, QUEUE_DELETED, QUEUE_REMOTE</td>
</tr>
</tbody>
</table>

** TSPT gate, GET function  
This function retrieves the first item in a “put” queue.

**Input parameters**

**QUEUE_NAME**
is the name of the queue being accessed.

**ITEM_BUFFER**
specifies the address (item_buffer_p) and maximum length (item_buffer_m) of the data area into which the data will be read. The actual data length is returned in item_buffer_n.

**Output parameters**

**[FMH]**
indicates whether the data contains an FMH. It can have either of these values:
YES|NO

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>IO_ERROR, INVALID_QUEUE_TYPE, QUEUE_NOT_FOUND, INVALID_QUEUE_NAME</td>
</tr>
</tbody>
</table>

** TSPT gate, GET_SET function  
This function retrieves the first item in a “put” queue into a set storage area.

**Input parameters**

**QUEUE_NAME**
is the name of the queue being accessed.

**OUTPUT parameters**

**ITEM_DATA**
returns the address and length of the item in set storage.

**[FMH]**
indicates whether the data contains an FMH. It can have either of these values:
YES|NO
Temporary storage domain (TS)

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>IO_ERROR, INVALID_QUEUE_TYPE, QUEUE_NOT_FOUND, INVALID_QUEUE_NAME</td>
</tr>
</tbody>
</table>

TSPT gate, GET_RELEASE function

This function retrieves and deletes the first item in a “put” queue. If the queue has one item, the queue is deleted.

Input parameters

QUEUE_NAME
is the name of the queue being accessed.

ITEM_BUFFER
specifies the address (item_buffer_p) and maximum length (item_buffer_m) of the data area into which the data will be read. The actual data length is returned in item_buffer_n.

Output parameters

[FMH]
indicates whether the data contains an FMH. It can have either of these values:
YES|NO

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>IO_ERROR, INVALID_QUEUE_TYPE, QUEUE_NOT_FOUND, LOCKED, INVALID_QUEUE_NAME, QUEUE_DELETED</td>
</tr>
</tbody>
</table>

TSPT gate, GET_RELEASE_SET function

This function retrieves the first item in a “put” queue into set storage and then deletes it. If the queue has one item, the queue is deleted.

Input parameters

QUEUE_NAME
is the name of the queue being accessed.

Output parameters

ITEM_DATA
returns the address and length of the item in set storage.
Temporary storage domain (TS)

[FMH] indicates whether the data contains an FMH. It can have either of these values:

YES | NO

RESPONSE is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>IO_ERROR, INVALID_QUEUE_TYPE, QUEUE_NOT_FOUND, LOCKED, INVALID_QUEUE_NAME, QUEUE_DELETED</td>
</tr>
</tbody>
</table>

TSPT gate, RELEASE function

This function deletes a "put" queue.

Input parameters

QUEUE_NAME is the name of the queue being deleted.

RESPONSE is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>INVALID_QUEUE_TYPE, QUEUE_NOT_FOUND, LOCKED, INVALID_QUEUE_NAME, QUEUE_DELETED</td>
</tr>
</tbody>
</table>

TSSH gate, INITIALIZE function

Initialize the Shared TS interface.

Input parameters

TSSH gate, INQUIRE_POOL_TOKEN function

Return token for the pool corresponding to the sysid provided.

Input parameters

RESPONSE is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>SYSID_NOT_FOUND</td>
</tr>
</tbody>
</table>
Temporary storage domain (TS)

**TSSH gate, WRITE function**

If the queue does not exist, this function creates a queue with the single item provided, and the queue’s “read cursor” is set to zero.

If the queue already exists, the item provided is appended to the queue, and the read cursor left unchanged.

**Input parameters**

- **[POOL_TOKEN]** is a token for the shared TS pool.
- **QUEUE_NAME** is the name of the queue being created or appended to.
- **ITEM_DATA** is the address and length of the item being written.
- **SUSPEND** indicates whether or not the request will be suspended if there is insufficient storage to satisfy the request.
- **FMH** indicates whether the data contains an FMH.
- **[TRANSID]** is the id of the transaction which issued this request.
- **[TRANSACTION_NUMBER]** is the 4-byte transaction number (in packed-decimal format).

**Output parameters**

- **TOTAL_ITEMS** is the total number of items in the queue on completion of the operation.
- **RESPONSE** is the domain’s response to the call. It can have any of these values:
  - OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED
- **[REASON]** is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>SERVER_ERROR, IO_ERROR, QUEUE_FULL, INSUFFICIENT_STORAGE, INVALID_LENGTH, INVALID_QUEUE_NAME, MAXIMUM_QUEUES_REACHED</td>
</tr>
</tbody>
</table>

**TSSH gate, REWRITE function**

This function updates the specified item in an existing queue. The read cursor is unchanged.

**Input parameters**

- **[POOL_TOKEN]** is a token for the shared TS pool.
- **QUEUE_NAME** is the name of the queue being updated.
- **ITEM_NUMBER** is the number of the item to be updated.
ITEM_DATA
is the address and length of the item being written.

SUSPEND
indicates whether the request will be suspended if there is insufficient storage to satisfy the request.

FMH indicates whether the data contains an FMH.

[TRANSACTION_NUMBER]
is the 4-byte transaction number (in packed-decimal format).

Output parameters

TOTAL_ITEMS
is the total number of items in the queue.

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>SERVER_ERROR, IO_ERROR, INSUFFICIENT_STORAGE, INVALID_LENGTH,</td>
</tr>
<tr>
<td></td>
<td>QUEUE_NOT_FOUND, ITEM_NOT_FOUND, INVALID_QUEUE_NAME</td>
</tr>
</tbody>
</table>

TSSH gate, READ INTO function

This function reads the specified queue item into a buffer provided by the caller. The read cursor for the queue is set to the item number provided. The caller provides the address (item_buffer_p) and buffer length (item_buffer_m). The actual length of the record is returned in item_buffer_n. If item_buffer_n is greater than item_buffer_m, the data is truncated (but an OK response is returned).

Input parameters

[POOL_TOKEN]
is a token for the shared TS pool.

QUEUE_NAME
is the name of the queue being read.

ITEM_NUMBER
is the number of the item to be read.

ITEM_BUFFER
specifies the address (item_buffer_p) and maximum length (item_buffer_m) of the data area into which the data will be read. The actual data length is returned in item_buffer_n.

[TRANSACTION_NUMBER]
is the 4-byte transaction number (in packed-decimal format).

Output parameters

TOTAL_ITEMS
returns the total number of items in the queue.

FMH indicates whether the data contains an FMH.
Temporary storage domain (TS)

RESPONSE

is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON]

is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>SERVER_ERROR, IO_ERROR, QUEUE_NOT_FOUND, ITEM_NOT_FOUND,</td>
</tr>
<tr>
<td></td>
<td>INVALID_QUEUE_NAME</td>
</tr>
</tbody>
</table>

TSSH gate, READ_SET function

This function reads the specified queue item into a storage area obtained by TS. The read cursor for the queue is set to the input item number.

Input parameters

[POOL_TOKEN]

is a token for the shared TS pool.

QUEUE_NAME

is the name of the queue being read.

ITEM_NUMBER

is the number of the item to be read.

[TRANSACTION_NUMBER]

is the 4-byte transaction number (in packed-decimal format).

Output parameters

ITEM_DATA

returns the address and length of the item data.

TOTAL_ITEMS

returns the total number of items in the queue.

FMH

indicates whether the data contains an FMH.

RESPONSE

is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON]

is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>IO_ERROR, SERVER_ERROR, QUEUE_NOT_FOUND, ITEM_NOT_FOUND,</td>
</tr>
<tr>
<td></td>
<td>INVALID_QUEUE_NAME</td>
</tr>
</tbody>
</table>

TSSH gate, READ_NEXT_INTO function

This function increments the read cursor by one and reads that item number into the buffer provided by the caller. The caller provides the address (item_buffer_p) and buffer length (item_buffer_m). The actual length of the record is returned in item_buffer_n. If item_buffer_n is greater than item_buffer_m, the data will have been truncated.
Input parameters

[POOL_TOKEN]
   is a token for the shared TS pool.

QUEUE_NAME
   is the name of the queue being read.

ITEM_BUFFER
   specifies the address (item_buffer_p) and maximum length (item_buffer_m) of the data area into which the data will be read. The actual data length is returned in item_buffer_n.

[TRANSACTION_NUMBER]
   is the 4-byte transaction number (in packed-decimal format).

ITEM_NUMBER
   returns the number of the item just read.

Output parameters

TOTAL_ITEMS
   returns the total number of items in the queue.

FMH
   indicates whether the data contains an FMH.

RESPONSE
   is the domain’s response to the call. It can have any of these values: OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
   is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>SERVER_ERROR, IO_ERROR, QUEUE_NOT_FOUND, ITEM_NOT_FOUND,</td>
</tr>
<tr>
<td></td>
<td>INVALID_QUEUE_NAME</td>
</tr>
</tbody>
</table>

TSSH gate, READ_NEXT_SET function

This function increments the queue’s read cursor by one and reads that item number into a storage area obtained by TS.

Input parameters

[POOL_TOKEN]
   is a token for the shared TS pool.

QUEUE_NAME
   is the name of the queue being read.

[TRANSACTION_NUMBER]
   is the 4-byte transaction number (in packed-decimal format).

Output parameters

ITEM_DATA
   returns the address and length of the item data.

ITEM_NUMBER
   returns the number of the item just read.

TOTAL_ITEMS
   returns the total number of items in the queue.
Temporary storage domain (TS)

FMH indicates whether the data contains an FMH.

RESPONSE is the domain’s response to the call. It can have any of these values: OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>SERVER_ERROR, IO_ERROR, QUEUE_NOT_FOUND, ITEM_NOT_FOUND, INVALID_QUEUE_NAME</td>
</tr>
</tbody>
</table>

TSSH gate, DELETE function
This function deletes the specified queue.

Input parameters

[POOL_TOKEN] is a token for the shared TS pool.

QUEUE_NAME is the name of the queue to be deleted. the request.

[TRANSACTION_NUMBER] is the 4-byte transaction number (in packed-decimal format).

RESPONSE is the domain’s response to the call. It can have any of these values: OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>SERVER_ERROR, IO_ERROR, QUEUE_NOT_FOUND, INVALID_QUEUE_NAME</td>
</tr>
</tbody>
</table>

TSSH gate, INQUIRE_SYSID_TABLE_TOKEN function

Input parameters

[POOL_TOKEN] is a token for the shared TS pool.

QUEUE_NAME is the name of the queue to be deleted. the request.

RESPONSE is the domain’s response to the call. It can have any of these values: OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>SERVER_ERROR, IO_ERROR, QUEUE_NOT_FOUND, INVALID_QUEUE_NAME</td>
</tr>
</tbody>
</table>
TSSB gate, START_BROWSE function

Input parameters
RESPONSE
is the domain’s response to the call. It can have any of these values:
OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>QUEUE_NOT_FOUND, BROWSE_END, SERVER_ERROR, IO_ERROR</td>
</tr>
</tbody>
</table>

TSSB gate, GET_NEXT function

Returns information about the next queue in the browse.

Input parameters
None

Output parameters
QUEUE_NAME
is the name of the queue.

[LAST_REFERENCED_TIME]
is the time at which the queue was last referenced.

[TOTAL_ITEMS]
is the total number of items in the queue.

[TOTAL_LENGTH]
is the sum of the lengths of all the items in the queue.

[MAXIMUM_ITEM_LENGTH]
is the length of the longest item in the queue.

[MINIMUM_ITEM_LENGTH]
is the length of the shortest item in the queue.

[TRANSID]
is the id of the transaction which created the queue.

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>BROWSE_END, SERVER_ERROR, IO_ERROR</td>
</tr>
</tbody>
</table>

TSSB gate, END_BROWSE function

Ends the browse.
**Temporary storage domain (TS)**

**Input parameters**

RESPONSE

is the domain's response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]

is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>BROWSE_END, SERVER_ERROR, IO_ERROR</td>
</tr>
</tbody>
</table>

**TSSB gate, INQUIRE_QUEUE function**

**Input parameters**

[POOL_TOKEN]

is the token for the pool being inquired upon.

QUEUE_NAME

is the name of the queue being inquired upon.

[KEY_COMPARISON]

specifies the constraints on the inquire. The default is KEY_COMPARISON(EQ). It can have any one of these values:

EQ|GT|GTEQ

[TRANSACTION_NUMBER]

is the 4-byte transaction number (in packed-decimal format).

**Output parameters**

[OUTPUT_QUEUE_NAME]

is the name of the queue whose information is returned. Note that this may differ from queue_name unless key_comparison(eq) is specified.

[LAST_REFERENCED_TIME]

is the time at which the queue was last referenced.

[TOTAL_ITEMS]

is the total number of items in the queue.

[TOTAL_LENGTH]

is the sum of the lengths of all the items in the queue.

[MAXIMUM_ITEM_LENGTH]

is the length of the longest item in the queue.

[MINIMUM_ITEM_LENGTH]

is the length of the shortest item in the queue.

[TRANSID]

is the id of the transaction which created the queue.

RESPONSE

is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]

is returned when RESPONSE is EXCEPTION. Possible values are:
## TSSR gate, SET_START_TYPE function

**Input parameters**

START

indicates the type of startup requested.

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

- OK
- DISASTER
- INVALID
- KERNERROR
- PURGED

## TSSR gate, SET_BUFFERS function

Sets the number of TS buffers to be used.

**Input parameters**

BUFFERS

the number of buffers required.

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

- OK
- DISASTER
- INVALID
- KERNERROR
- PURGED

## TSSR gate, SET_STRINGS function

This function sets the number of strings to be used.

**Input parameters**

STRINGS

the number of strings to be used.

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

- OK
- DISASTER
- INVALID
- KERNERROR
- PURGED

## TSBR gate, INQUIRE_QUEUE function

**Input parameters**

QUEUE_NAME

is the name of the queue being inquired upon.

**Output parameters**

[CREATION_TIME]

is the time at which the queue was created.

[LAST_REFERENCED_TIME]

is the time at which the queue was last referenced.

[TRANSID]

is the id of the transaction which created the queue.

[TOTAL_ITEMS]

is the total number of items in the queue.
Temporary storage domain (TS)

[TOTAL_LENGTH] is the sum of the lengths of all the items in the queue.

[MAXIMUM_ITEM_LENGTH] is the length of the longest item in the queue.

[MINIMUM_ITEM_LENGTH] is the length of the shortest item in the queue.

[STORAGE_TYPE] indicates whether the queue is held in main or auxiliary storage. It can have either of these values:

MAIN|AUXILIARY

RESPONSE is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>QUEUE_NOT_FOUND</td>
</tr>
</tbody>
</table>

TSBR gate, START_BROWSE function

Output parameters

RESPONSE is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>QUEUE_NOT_FOUND</td>
</tr>
</tbody>
</table>

TSBR gate, GET_NEXT function

Returns information about the next queue in the browse.

Input parameters

None

Output parameters

QUEUE_NAME is the name of the queue.

[CREATION_TIME] is the time at which the queue was created.

[LAST_REFERENCED_TIME] is the time at which the queue was last referenced.

[TRANSID] is the id of the transaction which created the queue.
is the total number of items in the queue.

[TOTAL_LENGTH]
is the sum of the lengths of all the items in the queue.

[MAXIMUM_ITEM_LENGTH]
is the length of the longest item in the queue.

[MINIMUM_ITEM_LENGTH]
is the length of the shortest item in the queue.

[STORAGE_TYPE]
indicates whether the queue is held in main or auxiliary storage. It can have either of these values:

MAIN|AUXILIARY

RESPONSE
is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>BROWSE_END</td>
</tr>
</tbody>
</table>

**TSBR gate, END_BROWSE function**

Ends the browse.

**Input parameters**

RESPONSE
is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>BROWSE_END</td>
</tr>
</tbody>
</table>

**TSBR gate, CHECK_PREFIX function**

Checks whether there are any queues with the prefix provided.

**Input parameters**

RESPONSE
is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>DUPLICATE, NOT_FOUND</td>
</tr>
</tbody>
</table>
Temporary storage domain (TS)

**TSIC gate, DELIVER_IC_RECOVERY_DATA function**

**Output parameters**

RESPONSE

is the domain’s response to the call. It can have any of these values:

OK|DISASTER|INVALID|KERNERROR|PURGED

**TSIC gate, INQUIRE_QUEUE function**

**Output parameters**

RESPONSE

is the domain’s response to the call. It can have any of these values:

OK|DISASTER|INVALID|KERNERROR|PURGED

**TSIC gate, SOLICIT_INQUIRES function**

This call is made from TS to IC to initiate inquire_queue requests from IC to TS.

**Output parameters**

RESPONSE

is the domain’s response to the call. It can have any of these values:

OK|DISASTER|INVALID|KERNERROR|PURGED

---

**Temporary storage domain’s generic gates**

Table 105 summarizes the storage manager domain’s generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMDM</td>
<td>TS 0101</td>
<td>INITIALIZE_DOMAIN</td>
<td>DMDM</td>
</tr>
<tr>
<td></td>
<td>TS 0102</td>
<td>QUIESCE_DOMAIN</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TS 0102</td>
<td>TERMINATE_DOMAIN</td>
<td></td>
</tr>
<tr>
<td>STST</td>
<td>TS 0501</td>
<td>COLLECT_STATISTICS</td>
<td>STST</td>
</tr>
<tr>
<td></td>
<td>TS 0502</td>
<td>COLLECT_RESOURCE_STATS</td>
<td></td>
</tr>
<tr>
<td>APUE</td>
<td>TS 0601</td>
<td>SET_EXIT_STATUS</td>
<td>APUE</td>
</tr>
<tr>
<td></td>
<td>TS 0602</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RMRO</td>
<td>TS 0401</td>
<td>PERFORM_PREPARE</td>
<td>RMRO</td>
</tr>
<tr>
<td></td>
<td>TS 0402</td>
<td>PERFORM_COMMIT</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PERFORM_SHUNT</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PERFORM_UNSHUNT</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>START_BACKOUT</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>END_BACKOUT</td>
<td></td>
</tr>
<tr>
<td>RMDE</td>
<td>TS 0401</td>
<td>START_DELIVERY</td>
<td>RMDE</td>
</tr>
<tr>
<td></td>
<td>TS 0402</td>
<td>DELIVER_RECOVERY</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>END_DELIVERY</td>
<td></td>
</tr>
<tr>
<td>RMKP</td>
<td>TS 0401</td>
<td>TAKE_KEYPOINT</td>
<td>RMKP</td>
</tr>
<tr>
<td></td>
<td>TS 0402</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For descriptions of these functions and their input and output parameters, refer to the sections dealing with the corresponding generic formats:
## Modules

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHTSDM</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>INITIALIZE_DOMAIN</td>
</tr>
<tr>
<td></td>
<td>QUIESCE_DOMAIN</td>
</tr>
<tr>
<td></td>
<td>TERMINATE_DOMAIN</td>
</tr>
<tr>
<td>DFHTSQR</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>WRITE</td>
</tr>
<tr>
<td></td>
<td>REWRITE</td>
</tr>
<tr>
<td></td>
<td>READ_INTO</td>
</tr>
<tr>
<td></td>
<td>READ_SET</td>
</tr>
<tr>
<td></td>
<td>READ_NEXT_INTO</td>
</tr>
<tr>
<td></td>
<td>READ_NEXT_SET</td>
</tr>
<tr>
<td></td>
<td>DELETE</td>
</tr>
<tr>
<td>DFHTSPT</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>PUT</td>
</tr>
<tr>
<td></td>
<td>PUT_REPLACE</td>
</tr>
<tr>
<td></td>
<td>GET</td>
</tr>
<tr>
<td></td>
<td>GET_SET</td>
</tr>
<tr>
<td></td>
<td>GET_RELEASE</td>
</tr>
<tr>
<td></td>
<td>GET_RELEASE_SET</td>
</tr>
<tr>
<td></td>
<td>RELEASE</td>
</tr>
<tr>
<td>DFHTSSH</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>INITIALIZE</td>
</tr>
<tr>
<td></td>
<td>INQUIRE_POOL_TOKEN</td>
</tr>
<tr>
<td></td>
<td>INQUIRE_SYSID_TABLE_TOKEN</td>
</tr>
<tr>
<td></td>
<td>WRITE</td>
</tr>
<tr>
<td></td>
<td>REWRITE</td>
</tr>
<tr>
<td></td>
<td>READ_INTO</td>
</tr>
<tr>
<td></td>
<td>READ_NEXT_INTO</td>
</tr>
<tr>
<td></td>
<td>READ_SET</td>
</tr>
<tr>
<td></td>
<td>READ_NEXT_SET</td>
</tr>
<tr>
<td></td>
<td>DELETE</td>
</tr>
<tr>
<td></td>
<td>START_BROWSE</td>
</tr>
<tr>
<td></td>
<td>GET_NEXT</td>
</tr>
<tr>
<td></td>
<td>END_BROWSE</td>
</tr>
<tr>
<td></td>
<td>INQUIRE_QUEUE</td>
</tr>
</tbody>
</table>
## Temporary storage domain (TS)

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHTSSR</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>SET_START_TYPE</td>
</tr>
<tr>
<td></td>
<td>SET_BUFFERS</td>
</tr>
<tr>
<td></td>
<td>SET_STRINGS</td>
</tr>
<tr>
<td></td>
<td>SET_EXIT_STATUS</td>
</tr>
<tr>
<td>DFHTSRM</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>PERFORM_PREPARE</td>
</tr>
<tr>
<td></td>
<td>PERFORM_COMMIT</td>
</tr>
<tr>
<td></td>
<td>PERFORM_SHUNT</td>
</tr>
<tr>
<td></td>
<td>PERFORM_UNSHUNT</td>
</tr>
<tr>
<td></td>
<td>START_BACKOUT</td>
</tr>
<tr>
<td></td>
<td>END_BACKOUT</td>
</tr>
<tr>
<td></td>
<td>START_DELIVERY</td>
</tr>
<tr>
<td></td>
<td>DELIVER_RECOVERY</td>
</tr>
<tr>
<td></td>
<td>END_DELIVERY</td>
</tr>
<tr>
<td></td>
<td>TAKE_KEYPOINT</td>
</tr>
<tr>
<td>DFHTSBR</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>INQUIRE_QUEUE</td>
</tr>
<tr>
<td></td>
<td>START_BROWSE</td>
</tr>
<tr>
<td></td>
<td>GET_NEXT</td>
</tr>
<tr>
<td></td>
<td>END_BROWSE</td>
</tr>
<tr>
<td></td>
<td>CHECK_PREFIX</td>
</tr>
<tr>
<td>DFHTSST</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>COLLECT_STATISTICS</td>
</tr>
<tr>
<td></td>
<td>COLLECT_RESOURCE_STATISTICS</td>
</tr>
<tr>
<td>DFHTSDUF</td>
<td>TS domain offline dump formatting routines</td>
</tr>
<tr>
<td>DFHTSDUC</td>
<td></td>
</tr>
<tr>
<td>DFHTSDUS</td>
<td></td>
</tr>
<tr>
<td>DFHTSITR</td>
<td>Interprets TS domain trace entries</td>
</tr>
</tbody>
</table>

### Exits

The temporary storage domain has four global user exit points: XTSQRIN, XTSQROUT, XTSPTIN and XTSPTOUT. For further information about these, see the [CICS Customization Guide](#).

### Trace

The point IDs for the temporary storage domain are of the form TS xxxx; the corresponding trace levels are TS 1, TS 2 and Exc.

For more information about the trace points, see the [CICS Trace Entries](#). For more information about using traces in problem determination, see the [CICS Problem Determination Guide](#).
Chapter 92. Terminal abnormal condition program

Terminal error processing for TCAM-supported terminals normally routes any error to the terminal abnormal condition program (DFHTACP). Depending on the type of error, DFHTACP issues messages, sets error flags, and places the terminal or line out of service.

Before default actions are taken, CICS passes control to the terminal error program (DFHTEP) for application-dependent action if necessary. On return from the terminal error program, DFHTACP performs the indicated action as previously set by DFHTACP or as altered by the TEP, a sample version of which is supplied by CICS (DFHXTEP in source code form). See "Chapter 94. Terminal error program" on page 1223 for further information about the TEP.

Design overview

The terminal abnormal condition program (DFHTACP) is used by terminal control to analyze any abnormal conditions. Appropriate action is taken with regard to terminal statistics, line statistics, terminal status, and line status; the task (transaction) can be terminated. Messages are logged to the transient data master terminal destination (CSMT) or the terminal log destination (CSTL). DFHTACP links to the user-supplied (or sample) terminal error program, passing a parameter list via a COMMAREA that is mapped by the DFHTEPCA DSECT. This allows the user to attempt recovery from transmission errors and to take appropriate action for the task.

Table 106 lists the various TACP message processing routines, which assemble the text of the messages and write them to one of three destinations depending on the type of error.

The matrix shown in Figure 93 on page 1193 defines the selection of message routines based on error code. The sequence in which the routines are executed is indicated by the number in the column corresponding to the error code. For example, for error code X'88', the processing routines are executed in the following order: ME, F, W, X, N, BA, and finally R.

Figure 94 on page 1194 gives a generalization of TACP’s default error handling upon completion of the message processing. For each error code, it shows the first routine to be called.

Table 106. TACP message routines

<table>
<thead>
<tr>
<th>Routine</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Establish DFHTC message number 2501 (Msg too long, please resubmit)</td>
</tr>
<tr>
<td>D</td>
<td>Establish DFHTC message number 2502 (TCT search error)</td>
</tr>
<tr>
<td>F</td>
<td>Establish DFHTC message number 2507 (Input event rejected)</td>
</tr>
<tr>
<td>H</td>
<td>Establish DFHTC message number 2506 (Output event rejected)</td>
</tr>
<tr>
<td>I</td>
<td>Establish DFHTC message number 2513 (Output length zero)</td>
</tr>
<tr>
<td>J</td>
<td>Establish DFHTC message number 2514 (No output area provided)</td>
</tr>
<tr>
<td>K</td>
<td>Establish DFHTC message number 2515 (Output area exceeded)</td>
</tr>
</tbody>
</table>
### Terminal abnormal condition program

Table 106. TACP message routines (continued)

<table>
<thead>
<tr>
<th>Routine</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>Establish DFHTC message number 2517 (Unit check SNS=ss, S.N.O.)</td>
</tr>
<tr>
<td>M</td>
<td>Establish DFHTC message number 2519 (Unit exception, S.N.O.)</td>
</tr>
<tr>
<td>N</td>
<td>Generate standard message inserts, for example, ‘at term tttt’</td>
</tr>
<tr>
<td>O</td>
<td>Generate special inserts for message DFHTC2500</td>
</tr>
<tr>
<td>Q</td>
<td>Write to terminal causing the error, after retrieving the message text from ME domain using an MEME RETRIEVE_MESSAGE call</td>
</tr>
<tr>
<td>R</td>
<td>Write to destination (CSMT or CSTL) using an MEME SEND_MESSAGE call to ME domain</td>
</tr>
<tr>
<td>T</td>
<td>Obtain terminal main storage area (message build area)</td>
</tr>
<tr>
<td>V</td>
<td>Establish DFHTC message number 2511 (Incorrect write request)</td>
</tr>
<tr>
<td>W</td>
<td>Establish ‘return code xx’ message insert</td>
</tr>
<tr>
<td>X</td>
<td>Convert hexadecimal byte into 2 printable characters</td>
</tr>
<tr>
<td>AB</td>
<td>Establish DFHTC message number 2534 (Incorrect destination)</td>
</tr>
<tr>
<td>AE</td>
<td>Establish DFHTC message number 2500 (Line</td>
</tr>
<tr>
<td>AF</td>
<td>Obtain terminal statistics</td>
</tr>
<tr>
<td>BA</td>
<td>Obtain line statistics</td>
</tr>
<tr>
<td>BB</td>
<td>Establish DFHTC message number 2516 (Unit check SNS=ss)</td>
</tr>
<tr>
<td>BC</td>
<td>Establish DFHTC message number 2518 (Unit exception)</td>
</tr>
<tr>
<td>BF</td>
<td>Establish DFHTC message number 2521 (Undetermined unit error)</td>
</tr>
<tr>
<td>CA</td>
<td>Establish DFHTC message number 2522 (Intercept required)</td>
</tr>
<tr>
<td>DB</td>
<td>Establish DFHTC message number 2529 (Unsolicited input)</td>
</tr>
<tr>
<td>ME</td>
<td>Initialize parameter list for calling ME domain</td>
</tr>
</tbody>
</table>
Figure 93. TACP message construction matrix
Terminal abnormal condition program

ROUTINE DESCRIPTION

1. Abend transaction
2. Put line in/out of service, as required
3. Put line (or terminal) out of service
4. I/O error test
5. Unsolicited input message
6. Test line for next operation.

Figure 94. TACP default error handling

Modules

DFHTACP

Exits

No global user exit points are provided for this function.

Trace

The following point ID is provided for the terminal abnormal condition program:
- AP 00E6, for which the trace level is TC 1.

DFHTACP provides trace entries immediately before and after calling DFHTEP.

For more information about the trace points, see the CICS Trace Entries. For more information about using traces in problem determination, see the CICS Problem Determination Guide.
Chapter 93. Terminal control

Terminal control allows communication between terminals and application programs. VTAM/NCP and TCAM are used for most terminal data control and line control services.

Terminal control supports automatic task initiation to process transactions that use a terminal but which are not directly initiated by the terminal operator (for example, printers).

Terminal control can also provide a simulation of terminals, using sequential devices, in order to help test new applications.

Design overview

The user can specify that concurrent terminal support is to be provided by any combination of the following access methods:

- VTAM
- TCAM
- Basic sequential access method (BSAM)
- Interregion communication (IRC)
- Console.

The primary function of terminal control is to take an input/output (I/O) request for a terminal and convert it to a format acceptable to the access method (VTAM, TCAM, or BSAM).

Terminal control uses data that describes the communication lines and terminals, kept in the terminal control table (TCT). The TCT is generated by the user as part of CICS system definition, or dynamically as needed. The TCT entries contain terminal request indicators, status, statistics, identification, and addresses of I/O and related areas.

When CICS terminal control is used with VTAM, VTAM itself resides in a separate address space, having a higher priority than CICS. VTAM-related control blocks and support programming comprise the CICS terminal control component. The application programs that run under CICS control communicate with terminals through the CICS terminal control interface with VTAM.

VTAM network functions allow terminals to be connected to any compatible control subsystem that is online. This enables a terminal operator to switch from one CICS system to another, or to another subsystem.

VTAM manages the flow of data between devices in the network and VTAM application programs such as CICS. VTAM is responsible for:

- Connecting, controlling, and terminating communication between the VTAM applications and terminal logical units
- Transferring data between VTAM applications and logical units
- Allowing VTAM applications to share communication lines, communication controllers, and terminals
- Controlling locally attached devices, that is, those not connected through a communication controller
Terminal control

- Providing tools to monitor network operations and make dynamic changes to the network configuration.

In a VTAM environment, the functions of CICS terminal control include:
- Establishing communication with terminal logical units (LUs) by issuing logon requests, communicated through the access method
- Handling terminal input and passing user program requests for communication to VTAM
- Returning terminal LUs to the access method by accepting logoff requests
- Taking measures to ensure the integrity of messages flowing to and from VTAM
- Performing logical error recovery processing for VTAM devices.

Terminal control issues VTAM macros to receive incoming messages, and routes them to the appropriate CICS application program for processing. Likewise, it sends messages destined for various devices in the network to VTAM, which then routes them to the appropriate location.

When TCAM controls communication lines, those lines are not dedicated to the CICS region. Thus a single terminal can access programs in separate regions supported by TCAM. TCAM facilities available within the region supported by TCAM include message switching, broadcasting, disk queuing, checkpoint and restart of the communication network, and TCAM terminal support.

Terminal control services

The following services are performed by, or in conjunction with, terminal control:
- Service request facilities
- System control services
- Transmission facilities.

Service request facilities

Write request
Sets up and issues or queues access method macros; performs journaling and journal synchronization.

Read request
Sets up and issues access method macros; performs journaling if required.

Wait request
Causes a dispatcher to suspend.

Dispatch analysis
Determines the type of access method and terminal used, and executes the appropriate area of terminal control.

System control services

Automatic task initiation
Services requests for automatic task (transaction) initiation caused by events internal to the processing of CICS.

Task initiation
Requests the initiation of a task to process a transaction from a terminal. When an initial input message is accepted, a task is created to do the processing.

Terminal storage
Performs allocation and deallocation of terminal storage.
Transmission facilities—VTAM

Connection services
Accepts logon requests, requests connection of terminals for automatic task initiation, and returns terminals to VTAM, as specified by the user. If the terminal has not been defined, CICS uses the VTAM logon information to autoinstall the terminal.

Transmission facilities—VTAM/non-VTAM

Access method selection
Passes control to the appropriate access method routine based on the access method specified in the terminal control table.

Wait
Synchronizes the terminal control task with all other tasks in the system. When all possible read and write operations have been initiated, terminal control processing is complete and control is returned to the transaction manager to allow dispatching of other tasks.

Transmission facilities—TCAM

TCAM message control program facilities
Performs the following functions:

- Invites and selects terminals to transmit or receive data
- Manages dynamic buffers
- Handles messages and directs the flow of data through the system on a priority basis
- Maintains queues in main storage and on direct-access devices for terminals and application programs
- Handles error checking, operator control, and checkpoint and restart.

Terminal error recovery

The resolution of certain conditions (for example, permanent transmission errors) involves both CICS and additional user coding. CICS cannot arbitrarily take all action with regard to these errors. User application logic is sometimes necessary to resolve the problem.

For the VTAM part of the network, terminal error handling is carried out by the node abnormal condition program (NACP) and a sample node error program (NEP), provided by CICS, or a user-written node error program. For further information about these, see “Chapter 59. Node abnormal condition program” on page 849 and “Chapter 60. Node error program” on page 855.

For the portion of the telecommunication network connected to TCAM or BSAM, these error-handling services are provided by the terminal abnormal condition program (TACP) and by the user-written or sample terminal error program (TEP). For further information about these, see “Chapter 92. Terminal abnormal condition program” on page 1191 and “Chapter 94. Terminal error program” on page 1223.

The following sequence of events takes place when a permanent error occurs for a terminal:

1. The terminal is “locked” against use.
2. The node or terminal abnormal condition program is attached to the system to run as a separate CICS task.
3. The node or terminal abnormal condition program writes the error data to a destination in transient data control if the user has defined one. This destination is defined by the user and can be intrapartition or extrapartition.

4. The node or terminal abnormal condition program then links to the appropriate node/terminal error program to allow terminal- or transaction-oriented analysis of the error. In the node or terminal error program, the user may decide, for example, to have the terminal placed out of service, have the line placed in or out of service, or have the transaction in process on the terminal abnormally terminated.

5. The terminal is "unlocked" for use.

6. The node or terminal abnormal condition program is detached from the system if no other terminals are to be processed.

**Testing facility—BSAM**

To allow the user to test programs, BSAM can be used to control sequential devices, such as card readers, printers, magnetic tape, and direct-access storage devices. These sequential devices can then be used to supply input/output to CICS before actual terminals are available or during testing of new applications.

**Terminal control modules (DFHZCP, DFHTC)**

Terminal control consists of two CICS resource managers:

**ZCP** DFHZCP, DFHZCX, and DFHZCXR provide both the common (VTAM and non-VTAM) interface, and DFHZCA, DFHZCB, DFHZCC, DFHZCW, DFHZCY, and DFHZCZ provide the VTAM-only support.

**TCP** DFHTCP provides the non-VTAM support (not MVS console support).

Terminal control communicates with application programs, CICS system control functions (transaction manager, storage control), CICS application services (basic mapping support and data interchange program), system reliability functions (abnormal condition handling), and operating system access methods (VTAM, TCAM, or BSAM).

Requests for terminal control functions made by application programs, BMS, or the transaction manager, are processed through the common interface of DFHZCP. Generally, terminal control requests for other CICS or operating system functions are issued by either ZCP or TCP, depending upon the terminal being serviced.

The ZCP and TCP suites of programs are loaded at CICS system initialization according to specified system initialization parameters, as follows:

- DFHTCP is loaded only if TCP=YES is specified.
- DFHZCP, DFHZCX, and DFHZCXR are always loaded.
- DFHZCA, DFHZCB, DFHZCY, and DFHZCZ are loaded only if VTAM=YES is specified.
- DFHZCC and DFHZCW are loaded only if ISC=YES is specified.

Figure 95 on page 1199 gives an overview of the terminal control resource managers.
Figure 96 on page 1200 shows the relationships between the components of terminal control.

Figure 95. Terminal control resource managers

Modules:
DFHTCP  TCAM, BSAM only  DFHZCW  VTAM only (ISC)
DFHZCA  VTAM only  DFHZCX  All
DFHZCB  VTAM only  DFHZCXR  All
DFHZCC  VTAM only (ISC)  DFHZCY  VTAM only
DFHZCP  All  DFHZCZ  VTAM only
Terminal control

Figure 96. Terminal control interfaces

Notes for Figure 96
Common interface

1. When a terminal control request is issued by an application program, or internally by the basic mapping support (BMS) routines using the DFHTC macro, request bits are set in the user’s task control area (TCA) and control is passed to the common interface (VTAM, non-VTAM) routines of DFHZCP.

2. If the request includes WAIT and the IMMED option is not in effect, control is passed to the transaction manager to place the requesting program (task) in a suspended state. If WAIT is not included, control is returned to the requesting task.

3. The task’s TCA contains the TCTTE address either in a field named TCAFCAAAA (facility control area associated address) or in a field named TCATPTA when passing TCATPTA to terminal control.

4. The dispatcher dispatches terminal control through the common interface (DFHZDSP in DFHZCP) for one of the following reasons:
   - The system address space exit time interval (specified by the ICV system initialization parameter) has elapsed since the last terminal control dispatch.
   - The specified terminal scan delay (specified by the ICVTSD system initialization parameter) has elapsed.
   - There is high-performance option (HPO) work to process.
   - The terminal control event has been posted complete (for example, an exit scheduled in the case of VTAM, or an event control block (ECB) posted in the case of non-VTAM), and CICS is about to go into a wait condition.

5. Terminal control, through its common interface (DFHZDSP) requests the dispatcher to perform a CICS WAIT when the terminal control task has processed through the terminal network and has no further work that it can do.

Access method dependent interface

6. Terminal control communicates with storage manager to obtain and release storage as follows:

   **VTAM**
   - ZCP modules issue domain calls for terminal storage (TIOAs), receive-any input area (RAIA) storage, and request parameter list (RPL) storage.

   **Non-VTAM**
   - DFHTCP issues DFHSC macros to obtain and release terminal and line storage.

7. Terminal control communicates with the transaction manager by means of the DFHKC macro. The macro can be issued by certain CICS control modules, depending upon the terminal being serviced. Terminal control may request the transaction manager to perform one of the following:
   - Attach a task upon receipt of a transaction identifier from a terminal.
   - Respond to a DFHKC TYPE=AVAIL request (a task control macro documented only for system programming) when a terminal is required by or for a task and that facility is available.

8. Terminal control communicates with operating system access methods in either of the following ways, depending upon the terminal being serviced:

   **VTAM**
   - ZCP (referring here to the resource manager) builds VTAM request information in the RPL which is then passed to VTAM for servicing. VTAM notifies terminal control of completion by placing completion information in the RPL. ZCP analyzes the contents of the RPL upon
Terminal control
completion to determine the type of completion and the presence of
error information. Communication with VTAM also occurs by VTAM
scheduling exits, for example, LOGON or LOSTERM. VTAM passes
parameter lists and does not always use an RPL.

When authorized-path VTAM has been requested (HPO),
communication with VTAM also occurs in service request block (SRB)
mode (using DFHZHPRX); ZCP uses the RPL with an extension to
communicate with its SRB mode code. When an SRB mode RPL request
is complete, ZCP calls the relevant exit or posts the ECB, as indicated
by the RPL extension.

Non-VTAM
DFHTCP builds access method requests in the data event control block
(DECB), which is part of the terminal control table line entry (TCTLE).
The DECB portion of the TCTLE is passed to the access method by
terminal control to request a service of that access method. The access
method notifies terminal control of the completion of the service
through the DECB. Terminal control analyzes the contents of the DECB
upon completion to determine the type of completion and to check for
error information.

9. Terminal control communicates with the CICS abnormal condition functions in
   either of the following ways, depending upon the terminal being serviced:

   VTAM
   The activate scan routine (DFHZACT, in the DFHZCA load module)
   attaches the CSNE transaction to run the node abnormal condition
   program (DFHZNAC); this is done during CICS initialization.
   DFHZNAC does some preliminary processing and then passes control
to the node error program (DFHZNEP). (The node error program can
be either your own version or the default CICS-supplied version.) Upon
the completion of the user’s error processing, control is returned to
DFHZNAC. (For further information about DFHZNAC, see
"Chapter 59. Node abnormal condition program" on page 849.)

   Non-VTAM
   DFHTCP attaches the CSTE transaction to run the terminal abnormal
condition program (TACP) and passes a terminal abnormal condition
line entry (TACLE) when an error occurs. The TACLE is a copy of the
DECB portion of the TCTLE and contains all information necessary for
proper evaluation of the error, together with special action indicators
that can be manipulated to alter the error correction procedure. After
analyzing the DECB, DFHTACP calls the terminal error program
(DFHTEP) with a COMMAREA containing the TACLE address. (The
terminal error program can be either your own version or the default
CICS-supplied version.) For further information about DFHTACP, see
"Chapter 92. Terminal abnormal condition program" on page 1191.

10. Terminal control is executed under either the user’s TCA or its own TCA as
    follows:

    User’s TCA
    a. During the application program interface
    b. During the interface with basic mapping support
    c. While performing direct VTAM terminal SEND requests.

    Terminal control’s TCA
    a. When the dispatcher dispatches terminal control
b. When terminal control issues a request to the transaction manager to attach a task

c. When terminal control issues a request to storage control

d. While performing non-VTAM terminal I/O or queued VTAM terminal I/O

e. For session-control functions when no task is attached.

Because many devices are supported by CICS terminal control, a large number of modules are required to provide this support.

Figure 97 gives an overview of the relationships between the functions within terminal control and the rest of CICS and Figure 98 on page 1204 through Figure 100 on page 1206 show some of the flows through the terminal control modules.
Terminal control

Figure 98. Terminal control ZCP and TCP common control routines
Terminal control

Chapter 93. Terminal control 1205
Terminal control

Enter from 'Select terminal routine'
DFHTCSTD

Enter EP DFHxxxxx

Event analysis

Completion code analysis

Output event completion analysis

Activity control

Input event completion analysis

Input event task initiation

Output event preparation

Input event preparation

Event initiation

Exit to
TCCCTMR
(advance to next line and wait)

Figure 100. Terminal control general flow through device-dependent modules (TCP only)
High-performance option
When running CICS under MVS, the high-performance option (HPO) can be used. HPO uses VTAM with CICS as an authorized program so that the VTAM path length is reduced. This is achieved by dispatching SRBs to issue the send and receive requests for data to and from the terminals. The SRB code is executed in the DFHZHPRX module.

System console support
One or more MVS system consoles can be used as CICS terminals. This includes any MVS extended console introduced from MVS/ESA SP 4.1 onward; for example, a TSO user issuing the TSO CONSOLE command.

Each console has a unique number (releases prior to MVS/ESA SP 4.1) or a unique name (MVS/ESA SP 4.1 onwards). This matches the console number or name defined in the MVS system generation. Consoles are defined to CICS using CEDA DEFINE TERMINAL (see "Chapter 72. Resource definition online (RDO)" on page 997). The console number or name is specified using the CONSOLE or CONSNAME keyword respectively, depending on the level of MVS.

The console operator communicates with CICS using the MVS MODIFY command to start transactions. CICS communicates with the console using either the WTO macro or the WTOR macro.

A system console is modeled by CICS as a TCTTE that has an associated control block, the console control element (CCE). The CCE holds the event control block (ECB) for the console, and both the console ID and the console name.

The interface between a system console and CICS is the command input buffer (CIB), which is created in MVS-protected storage for each MODIFY command. A CIB contains the data for a MODIFY command. CICS addresses the first CIB using the EXTRACT macro and the CIBs are chained together.

The MVS communication ECB is in MVS-protected storage; it is posted complete for each MODIFY command and reset when there are no CIBs to be processed. The CICS system wait list holds pointers to the MVS communication ECB and the ECB for each system console.

When CICS is initialized, an EXTRACT macro is executed to obtain the job name and point to the MVS communication ECB and the first CIB; all these are stored in the TCT prefix.

DFHZCP contains two modules, DFHZCNA and DFHZCNR, which perform system console support.

DFHZCNA is used to:
• Resume a task on completion of a terminal event for the task
• Attach a task to satisfy a request for transaction initiation by a MODIFY command
• Attach a task (AVAIL) requested by automatic transaction initiation (ATI)
• Detach a terminal from a task when the task has completed
• Shut down console support when CICS is quiescing.

DFHZCNR is used to:
Terminal control

- Issue WTO macros for application program WRITE requests
- Issue WTO and WTOR macros for application program CONVERSE or (WRITE,READ) requests
- Issue a WTOR macro with message DFH4200 for application program READ requests.

Console support control modules
DFHZDSP calls DFHZCNA to scan the consoles for any activity.

DFHZCNA checks whether any task is suspended because it is waiting for a terminal event, for example, a READ, and, if the event is completed, resumes that task before starting any new task. This is done by scanning the CCE chain for ECBs that have been posted by MVS.

When a MODIFY command is executed, the communication ECB is posted complete and a CIB for the command is added to the end of the CIB chain. DFHZCNA processes the CIB chain in first-in, first-out order. For each CIB, DFHZCNA searches the CCE chain for the console. With MVS/ESA SP 4.1 (or later), the search is on console name; otherwise, the search is on console ID.

The task is then attached if the ‘task pending’ flag in the CCE is not set by a preceding CIB in the chain. In the course of scanning the CIB chain, DFHZCNA may find a MODIFY command that requires a task to be attached, but cannot attach the task immediately because there is already a task active, or there is an outstanding error condition to clear. DFHZCNA therefore sets the ‘task pending’ flag in the CCE to remember the existence of the CIB. During the CIB chain scan, the condition preventing the task attach might clear, and a subsequent CIB might be selected for attach. However, the ‘task pending’ flag prevents this, and ensures that CIBs are processed in order. All ‘task pending’ flags are reset before each CIB chain scan.

If the task is to be attached, DFHZCNA obtains a TIOA and moves the data from the CIB to the TIOA. DFHZAITT is then called to attach the task. If the attach fails, the TIOA is freed. A QEDIT macro frees the CIB if the attach is successful, and the scan continues.

When a transaction is automatically initiated and DFHKCP schedules the transaction for a terminal which is a console, a flag is set in the CCE by DFHZLOC. After DFHZCNA has completed scanning the CIB chain, it checks that the console does not have a task already attached and there is not a CIB on the chain for the console; if both these conditions are satisfied, the task is attached.

DFHZCNA issues a QEDIT macro to prevent any more MODIFY commands being accepted when CICS is shutting down. Any MODIFY commands on the CIB chain after shutdown has been started are processed. When other access methods have been quiesced, and there are no tasks attached for a console, console support is shut down.

If a console not defined to CICS is used to enter a MODIFY command, DFHZCNA sets up an error code and links to DFHACP to issue the error message. This is done using the TCTTE for the error console, CERR.

DFHZCNR sends terminal control requests from an application program to a specific system console by issuing WTO and WTOR macros. It is called by DFHZARQ.
For a WRITE request, DFHZNCR executes either a single WTO macro, or one or more multiline WTO macros, depending on the amount of data specified for the request.

For a READ request, DFHZNCR acquires a TIOA for the reply area and executes a WTOR macro with a CICS-supplied message, DFH4200. This message requests the operator to reply, and the transaction waits for this reply.

For a CONVERSE or (WRITE,READ) request, DFHZNCR acquires a TIOA for the reply area and executes a WTOR macro with the data specified for the WRITE. If there is any data remaining, DFHZNCR then executes either a single WTO macro, or one or more multiline WTO macros, depending on the amount of data. The transaction then waits until the operator replies to this request.

Defining terminals to CICS

Terminal definitions are created as CSD records or DFHTCT macros (non-VTAM only) and then installed in (added to) the terminal control table (TCT) as TCT terminal entries (TCTTEs).

When a cold start is performed, CICS obtains its TCT entries from DFHTCT macros or from groups of resource definitions in the CSD file, which are named in the GRPLIST system initialization parameter. These are recorded in the CICS catalog.

When a warm start is performed, CICS obtains the definitions from the DFHTCT macros and from the CICS catalog; the GRPLIST is ignored.

On emergency restart, CICS obtains the definitions from the DFHTCT macros and from the CICS catalog; the GRPLIST is ignored. Then CICS re-applies any in-flight TCT updates using information from the system log.

During CICS execution, TCT entries can be added as follows:

- By using the CEDA INSTALL command
- By the autoinstall process when an unknown terminal logs on
- By the transaction routing component when a TCT entry is shipped from a terminal-owning to an application-owning region.
- By using the EXEC CICS CREATE command

During CICS execution, TCT entries can be deleted as follows:

- By using the EXEC CICS DISCARD command
- By the autoinstall process when an autoinstalled terminal logs off or has been logged for a period.
- By the transaction routing component when a TCT entry has been unused for a period.
- Using the CEDA INSTALL, EXEC CICS CREATE, transaction routing, or autoinstall processes to replace the old entry.

Figure 101 on page 1210 shows the terminal control table (TCT).
DFHZCQ installs, deletes, catalogs, uncatalogs, recovers, and inquires on terminals. Entries are installed in and deleted from the terminal control table by DFHZCQ. DFHZCQ is called by the following modules:

DFHAMTP
For the CEDA transaction and EXEC CICS CREATE, to install TCT entries
Terminal control

DFHEIQSC
For EXEC CICS DISCARD CONNECTION, to discard a connection.

DFHEIQST
For EXEC CICS DISCARD TERMINAL, to discard a terminal.

DFHTBSS
During CICS initialization, to restore terminal definitions at warm or emergency restart

DFHZATA
The autoinstall program

DFHZATD
The autoinstall delete program

DFHZATS
When a TCT entry is shipped, installed, or deleted for transaction routing

DFHZTSP
When a transaction route request is received to recatalog the connection if certain characteristics have changed.

DFHQRY
When the QUERY function is used to discover the actual characteristics of a device, complete the TCT entry, and recatalog the resulting TCTTE

DFHWKP
The warm keypoint program, to record information for RDO-eligible terminals in the CICS catalog, and to uncatalog autoinstalled entries.

DFHZCQ calls the table builder services (TBS) modules which in turn, call the appropriate DFHBSxxx modules to build the TCTTE for the input parameters. DFHZCQ is heavily dependent on the module that calls it to supply the complete set of parameters to be used to create the TCTTE; DFHZCQ itself is not responsible for determining parameters for the TCTTE.

DFHBS* builder programs
DFHZCQ calls the builder programs, whose names all begin DFHBS. These builders are responsible for creating TCTTEs. The parameters given to DFHZCQ are passed on to the builders, which extract the parameters and set the relevant fields in the TCTTE.

For further information about builders, see “Chapter 11. Builders” on page 167.

Contents of the TCT
The TCT describes the logical units (LUs) known to CICS. Each active LU is represented by a terminal control table terminal entry (TCTTE). The TCT does not describe the network configuration; it describes the CICS logical viewpoint of the network.

The TCT contains pointers to these VTAM-related control blocks:
- Access method control block (ACB)— Link an application program, such as CICS, to VTAM
- Receive-any control blocks (RA-RPL, RA-ECB, RACE)— Process initial transaction input
- Node initialization block (NIB) descriptors and bind-area models— Used during logon processing
- TCTTEs— Describe the logical units known to CICS
Terminal control

- ACB and RPL exit lists— Point to the VTAM exit routines.

TCT indexing (DFHZGTI and DFHZLOC)
There are two types of requests that can be used in CICS to locate terminal entries:
1. DFHZGTI calls
2. and DFHTC CTYPE=LOCATE calls
Both these modules use DFHTM calls to a variety of indexes and chains to locate terminal entries in the TCT with efficiency.

The DFHZGTI module has the following call types:

Locate  Find a TCT entry in the given ‘domain’ which matches the name

GetStart  Obtain a browse token for Getnexts.

GetFirst  Find the first entry that matches the name in the given domain.

GetNext  Find the next entry that matches the name in the given domain.

GetEnd  Release the browse token

Release  Unlock an entry

Callers can decide to have an entry returned as locked or unlocked.

In DFHZGTI the total TCT is carved up into ‘domains’. A TCT entry can reside in several domains depending on its type. Callers to DFHZGTI specify one domain on a call and are returned one entry that fits the name (or partial name) that is supplied. DFHZGTI calls can be for the following domains:

**Terminal by termid**
All terminals (local, remote, non-vtam) by the terminal id (4-char).

**Session by termid**
All sessions (VTAM, MRO, remote) by the terminal id (4-char).

**Global by termid**
All terminal and all sessions by the terminal id (4-char).

**System by sysid**
All connections (local, remote) by the sysid (4-char)

**MRO system by sysid**
MRO connections by sysid (4-char).

**LU61 system by sysid**
LU61 connections by sysid (4-char).

**REMDEL system by sysid**
Systems that need REMDEL sent to them (because they do not support timeout) when a local entry is deleted by sysid (4-char).

**Terminal by netname**
VTAM local terminals by the netname (8-char).

**System by netname**
All connections (local, remote) by the netname (8-char).
Remote terminal by netname
Remote terminals by the netname (8-char).

Global by netname
Terminals, remote terminals and sessions by the netname (8-char).

Remote by Unique
All remote terminals and remote connections by the unique name that is Terminal-Owning-Region (TOR) netname, followed by a period, followed by the termid or sysid in the TOR. (13-char).

Remote terminal by Rsyisd
Remote terminals by the value of REMOTESYSTEM (4-char).

Remote system by Rsyisd
Remote connections by the value of REMOTESYSTEM (4-char).

Indirect system by Rsyisd
Indirect connections by the value of REMOTESYSTEM (4-char).

Generic system by mbrname
Generic connections by the member-name of the connection in the generic VTAM resource (8-char).

DFHTC CTYPE=LOCATE calls are processed by DFHZLOC. DFHZLOC does not have access to as wide a range of domains as DFHZGTI, but it provides extra facilities such as finding particular types of sessions for a connection. Both DFHZGTI and DFHZLOC can lock TCT entries.

Locks
The table manager program (DFHTMP) is used to locate TCT entries by both DFHZGTI and DFHZLOC. When DFHTMP gives the address of an entry, it notes the address of the calling task, and this has the effect of a shared lock unless the caller asked for the entry not to be locked. All locks are released implicitly at the end of the task.

When a TCT entry is deleted, it must not be in use by another task. This is achieved by issuing the DFHTM QUIESCE macro. Other tasks that issue DFHTM LOCATE for that entry are suspended when they acquire a shared lock. These tasks are resumed when the original task issues a delete (if the commit option is used), or at syncpoint if not.

In addition to TMP read locks, DFHZLOC and DFHZGTI, use update locks which are obtained and released by DFHZGTA. DFHZGTA's involvement in TCT updates is discussed in "Chapter 11. Builders" on page 167. For efficiency, two flags in each TCT entry (one for delete and one for update) are examined before a TCT entry is returned. If either is set, and the request does not ask to see all updates, DFHZGTA is called to determine if the inquiring task holds the lock on the termid or sysid name. If it does, the entry is returned, otherwise the entry is ignored. This hides entries that are being installed or replaced from other parts of CICS until they are ready to be used, without requiring a lock search for each inquiry. The Builders, see "Chapter 11. Builders" on page 167, are responsible for setting and resetting the flags in the TCT entry.

The following sections describe some of the callers of DFHZCQ.
System initialization (DFHTCRP, DFHAPRDR and DFHTBSS)
The DFHTCRP program is responsible for reestablishing TCTTEs that were in existence in the previous CICS run. There are three stages of processing in DFHTCRP:
1. Initialize DFHZCQ and DFHAPRDR, then exit if START=COLD
2. Reestablish TCTTEs recorded in the CICS catalog calling DFHZCQ for each one.
3. Call DFHAPRDR to allow it to proceed and forward-recover in-flight updates to TCTTEs recorded in the system log at emergency restart or XRF takeover.

The DFHAPRDR program is called by DFHTCRP in two phases:
1. To initialize its control blocks.
2. To wait until Recovery Manager has delivered any inflight log records and DFHAPRDR (running on another task) has called DFHTBSS to recover them.

DFHAPRDR is called by Recovery Manager (RM) for each log record that are for UOWs that did not write a Forget record to the system log when CICS failed. It is then called again to denote the end of any such records. On this call DFHAPRDR waits until DFHTCRP has rebuilt the TCT from the catalog, and then calls DFHTBSS to recover each log record (which will update the TCT and catalog). Then it posts DFHTCRP to show that the TCT has recovered and returns to Recovery Manager.

The DFHTBSS program is called by DFHAPRDR with log records for TCT updates that were being written to the catalog when CICS failed. It then calls DFHZCQ to re-install or re-delete the entries that the log records represent.

CEDA INSTALL and EXEC CICS CREATE (DFHAMTP)
When the CEDA INSTALL command is used to install a group of TERMINAL definitions, the flow of control is as follows:
1. DFHAM processes CEDA and EXEC CICS CREATE commands.
2. DFHAM calls DFHTOR and then DFHZCQ.
3. DFHTOR receives as input a partial definition (TERMINAL, TYPETERM, CONNECTION, or SESSIONS), calling one of the DFHTOAxx modules, depending on the type of resource definition:
   • DFHTOAxx adds a partial definition to a BPS. For a terminal device, a complete BPS is built from information from one TYPETERM and one TERMINAL definition; for an ISC or MRO link, a complete BPSes are built from information from one CONNECTION and one (or more) SESSIONS definition(s).
   • DFHTOBPS builds the BPS, calling one of the DFHTRZxP modules to translate the parameter list into BPS format.
4. When DFHTOR has built a complete BPS, it returns it to DFHAM, ready to be passed to DFHZCQ.

For additional information about this process, see “Chapter 72. Resource definition online (RDO)” on page 997.

Autoinstall
For information about this process, see “Chapter 8. Autoinstall for terminals, consoles and APPC connections” on page 113.
QUERY function (DFHQRY)

The QUERY function (DFHQRY) is used to determine the characteristics of IBM 3270 Information Display System devices, and complete the information about a device in the TCTTE. DFHQRY sends a read partition query structured field to the device, and analyzes the response. The TCTTE fields mainly affected are those used by basic mapping support (BMS), such as extended attributes. If QUERY(ALL) or QUERY(COLD) is specified in the terminal definition, DFHQRY is executed before any other transaction is initiated at a terminal. If QUERY(ALL) is specified, this is done after each logon. If QUERY(COLD) is specified, it is only done following the first logon after a cold start. After completing the TCTTE fields, DFHQRY calls DFHZCQ to recatalog the TCTTE.

Control blocks

Figure 102 on page 1216 shows the control blocks associated with terminal control.
**Notes:**

1. TACLE is created only when line or terminal error has occurred.
Figure 102. Control blocks associated with terminal control (Part 2 of 2)
Figure 103 shows the TCTLE and Figure 104 shows the TACLE.

### DFHTCTLE

<table>
<thead>
<tr>
<th>Address</th>
<th>Description</th>
</tr>
</thead>
</table>
| X'15'       | TCTLETLA  
Address of terminal list         |
| X'40'       | TCTLEPLA  
Address of polling list            |
| X'44'       | TCTLETEA  
Address of active term table entry  |
| X'4C'       | TCTLEECA  
Address of line error chain         |
| X'54'       | TCTLEPA   
Address of first terminal on line    |

**Figure 103. Terminal control table line entry (TCTLE)**

Terminal input/output areas (TIOAs) are set up by storage control and chained to the terminal control table terminal entry (TCTTE) as needed for terminal input/output operations. The TCTTE contains the address of the first terminal-type storage area obtained for a task (the beginning of the chain), and the address of the active TIOA.

See the [CICS Data Areas](#) manual for a detailed description of these control blocks.

### Modules

The DFHZCx modules contain CSECTs that issue VTAM macros to perform specific communication functions, and exit routines that are driven by VTAM when network events occur that are related to CICS.

The following is a list of the DFHZCx load modules concerned with terminal control and VTAM management in CICS, together with brief descriptions of their component object modules (CSECTs):
# Module CSECT Description

<table>
<thead>
<tr>
<th>DFHZCA</th>
<th>DFHZACT</th>
<th>Activate scan</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHZFRE</td>
<td>Freemain</td>
<td></td>
</tr>
<tr>
<td>DFHZGET</td>
<td>Getmain</td>
<td></td>
</tr>
<tr>
<td>DFHZQUE</td>
<td>Queue manager</td>
<td></td>
</tr>
<tr>
<td>DFHZRST</td>
<td>RESETSR request</td>
<td></td>
</tr>
<tr>
<td>DFHZCB</td>
<td>DFHZATI</td>
<td>Automatic task initiation</td>
</tr>
<tr>
<td>DFHZDET</td>
<td>Task detach</td>
<td></td>
</tr>
<tr>
<td>DFHZHPSR</td>
<td>Authorized path SRB requests</td>
<td></td>
</tr>
<tr>
<td>DFHZLRP</td>
<td>Logical record presentation</td>
<td></td>
</tr>
<tr>
<td>DFHZRAC</td>
<td>Receive-any completion</td>
<td></td>
</tr>
<tr>
<td>DFHZRAS</td>
<td>Receive-any slowdown processing</td>
<td></td>
</tr>
<tr>
<td>DFHZRVS</td>
<td>Receive specific</td>
<td></td>
</tr>
<tr>
<td>DFHZRVX</td>
<td>Receive specific exit</td>
<td></td>
</tr>
<tr>
<td>DFHZSDR</td>
<td>Send response</td>
<td></td>
</tr>
<tr>
<td>DFHZSDS</td>
<td>Send DFSYN</td>
<td></td>
</tr>
<tr>
<td>DFHZSDX</td>
<td>Send synchronous data exit</td>
<td></td>
</tr>
<tr>
<td>DFHZSSX</td>
<td>Send DFSYN command exit</td>
<td></td>
</tr>
<tr>
<td>DFHZUIX</td>
<td>User input exit</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DFHZCC</th>
<th>DFHZARER</th>
<th>Protocol error and exception handler</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHZARL</td>
<td>APPC application request logic</td>
<td></td>
</tr>
<tr>
<td>DFHZARM</td>
<td>APPC migration logic</td>
<td></td>
</tr>
<tr>
<td>DFHZARR</td>
<td>Application receive request logic</td>
<td></td>
</tr>
<tr>
<td>DFHZARRA</td>
<td>Application receive buffer support</td>
<td></td>
</tr>
<tr>
<td>DFHZARRC</td>
<td>Classify what next to receive</td>
<td></td>
</tr>
<tr>
<td>DFHZARRF</td>
<td>Receive FMI7 and ER1</td>
<td></td>
</tr>
<tr>
<td>DFHZBKT</td>
<td>Bracket state machine</td>
<td></td>
</tr>
<tr>
<td>DFHZCHS</td>
<td>Chain state machine</td>
<td></td>
</tr>
<tr>
<td>DFHZCNT</td>
<td>Contention state machine</td>
<td></td>
</tr>
<tr>
<td>DFHZCRT</td>
<td>RPL_B state machine</td>
<td></td>
</tr>
<tr>
<td>DFHZRLP</td>
<td>GDS post-VTAM receive logic</td>
<td></td>
</tr>
<tr>
<td>DFHZRLX</td>
<td>GDS receive exit logic</td>
<td></td>
</tr>
<tr>
<td>DFHZRNL</td>
<td>GDS pre-VTAM receive logic</td>
<td></td>
</tr>
<tr>
<td>DFHZSDL</td>
<td>GDS send logic</td>
<td></td>
</tr>
<tr>
<td>DFHZSLX</td>
<td>GDS send exit logic</td>
<td></td>
</tr>
<tr>
<td>DFHZSTAP</td>
<td>Conversation state determination</td>
<td></td>
</tr>
<tr>
<td>DFHZUSR</td>
<td>Conversation state machine</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DFHZCP</th>
<th>DFHZARQ</th>
<th>Application request handler</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHZATT</td>
<td>Attach routine</td>
<td></td>
</tr>
<tr>
<td>DFHZCNA</td>
<td>MVS console</td>
<td></td>
</tr>
<tr>
<td>DFHZDSP</td>
<td>Dispatcher</td>
<td></td>
</tr>
<tr>
<td>DFHZISP</td>
<td>Allocate/free/point</td>
<td></td>
</tr>
<tr>
<td>DFHZSUP</td>
<td>Startup task</td>
<td></td>
</tr>
<tr>
<td>DFHZUCT</td>
<td>3270 uppercase translate</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DFHZCW</th>
<th>DFHZERH</th>
<th>APPC ERP logic</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHZEV1</td>
<td>APPC bind security (part 1)</td>
<td></td>
</tr>
<tr>
<td>DFHZEV2</td>
<td>APPC bind security (part 2)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DFHZCX</th>
<th>DFHSNAS</th>
<th>Create signon/sign-off ATI sessions</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHSNPU</td>
<td>Preset userid signon/sign-off</td>
<td></td>
</tr>
<tr>
<td>DFHSNSU</td>
<td>Session userid signon/sign-off</td>
<td></td>
</tr>
<tr>
<td>DFHSNTU</td>
<td>Terminal userid signon/sign-off</td>
<td></td>
</tr>
<tr>
<td>DFHSNUS</td>
<td>US domain - local and remote signon</td>
<td></td>
</tr>
<tr>
<td>DFHSNXR</td>
<td>XRF reflecting signon state</td>
<td></td>
</tr>
<tr>
<td>DFHZABD</td>
<td>Abend routine for incorrect requests</td>
<td></td>
</tr>
<tr>
<td>DFHZAND</td>
<td>Build TCB before issuing PC abends</td>
<td></td>
</tr>
<tr>
<td>DFHZCNR</td>
<td>MVS console request</td>
<td></td>
</tr>
<tr>
<td>DFHZIS1</td>
<td>ISC/IRC syncpoint</td>
<td></td>
</tr>
<tr>
<td>DFHZIS2</td>
<td>IRC internal requests</td>
<td></td>
</tr>
<tr>
<td>DFHZLOC</td>
<td>Locate TCTTE and ATI requests</td>
<td></td>
</tr>
<tr>
<td>DFHZSTU</td>
<td>Status changing TCTTEs/LCDs and TCTSEs</td>
<td></td>
</tr>
</tbody>
</table>
Terminals control

Module CSECT Description

<table>
<thead>
<tr>
<th>CSECT</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHZCXR</td>
<td>DFHBSXGS APPC session name generation</td>
</tr>
<tr>
<td></td>
<td>DFHZTSP  Terminal sharing functions</td>
</tr>
<tr>
<td></td>
<td>DFHZXRL  APPC command routing</td>
</tr>
<tr>
<td></td>
<td>DFHZXRT  Routed APPC command handling</td>
</tr>
<tr>
<td>DFHZCY</td>
<td>DFHZAX   DFASY exit</td>
</tr>
<tr>
<td></td>
<td>DFHZDST  SNA-ASCII translation</td>
</tr>
<tr>
<td></td>
<td>DFHZLIX   LERAD exit</td>
</tr>
<tr>
<td></td>
<td>DFHZLIX   LOGON exit</td>
</tr>
<tr>
<td></td>
<td>DFHZLTX   LOSTERM exit</td>
</tr>
<tr>
<td></td>
<td>DFHZNSP  Network services exit</td>
</tr>
<tr>
<td></td>
<td>DFHZOPA  Open VTAM ACB</td>
</tr>
<tr>
<td></td>
<td>DFHZRXX  Release request exit</td>
</tr>
<tr>
<td></td>
<td>DFHZRSY1 Resynchronization part 1</td>
</tr>
<tr>
<td></td>
<td>DFHZRSY2 Resynchronization part 2</td>
</tr>
<tr>
<td></td>
<td>DFHZRSY3 Resynchronization part 3</td>
</tr>
<tr>
<td></td>
<td>DFHZRSY4 Resynchronization part 4</td>
</tr>
<tr>
<td></td>
<td>DFHZRSY5 Resynchronization part 5</td>
</tr>
<tr>
<td></td>
<td>DFHZRSY6 Resynchronization part 6</td>
</tr>
<tr>
<td></td>
<td>DFHZXSCX SESSION control input exit</td>
</tr>
<tr>
<td></td>
<td>DFHZSDA  Send command</td>
</tr>
<tr>
<td></td>
<td>DFHZSES  SESSIONC</td>
</tr>
<tr>
<td></td>
<td>DFHZSEX  SESSIONC exit</td>
</tr>
<tr>
<td></td>
<td>DFHZSHU  Shutdown VTAM</td>
</tr>
<tr>
<td></td>
<td>DFHZSIM  SIMLOGON</td>
</tr>
<tr>
<td></td>
<td>DFHZSIX  SIMLOGON exit</td>
</tr>
<tr>
<td></td>
<td>DFHZSRK  Send response to command</td>
</tr>
<tr>
<td></td>
<td>DFHZSLB  SETLOGON start</td>
</tr>
<tr>
<td></td>
<td>DFHZSYN  Handle CTYPE=syncpoint/recover request</td>
</tr>
<tr>
<td></td>
<td>DFHZXYX  SYNAD exit</td>
</tr>
<tr>
<td></td>
<td>DFHZTPX  TPEND exit</td>
</tr>
<tr>
<td></td>
<td>DFHZTRA  Create ZCP/VIO trace requests</td>
</tr>
<tr>
<td></td>
<td>DFHZXPS  APPC persistent session recovery</td>
</tr>
<tr>
<td></td>
<td>DFHZXRC  XRF and persistent sessions state data analysis</td>
</tr>
<tr>
<td>DFHZCZ</td>
<td>DFHZCLS  CLSDST</td>
</tr>
<tr>
<td></td>
<td>DFHZCLX  CLSDST exit</td>
</tr>
<tr>
<td></td>
<td>DFHZCQ   CTYPE command request</td>
</tr>
<tr>
<td></td>
<td>DFHZENW  Error message writer</td>
</tr>
<tr>
<td></td>
<td>DFHZOPN  OPNDST</td>
</tr>
<tr>
<td></td>
<td>DFHZOPX  OPNDST exit</td>
</tr>
<tr>
<td></td>
<td>DFHZRAQ  Read ahead queuing</td>
</tr>
<tr>
<td></td>
<td>DFHZRAR  Read ahead retrieval</td>
</tr>
<tr>
<td></td>
<td>DFHZTAX  Turnaround exit</td>
</tr>
</tbody>
</table>

**Exits**

DFHZCB has three global user exit points: XZCIN, XZCOUT, and XZCOUT1.

DFHZCP has one global user exit point: XZCATT

DFHTCP has the following global user exit points: XTCIN, XTCOUT, XTCATT, XTCIN, and XTCOUT1

For further information about these, see the CICS Customization Guide.
The following point IDs are provided for terminal control:

- AP 00E6 (DFHTCP), for which the trace level is TC 2
- AP 00FC (DFHZCP), for which the trace level is TC 1
- AP FBxx, for which the trace levels are TC 1, TC 2 and Exc
- AP FCxx, for which the trace levels are TC 1, TC 2, and Exc
- AP FDxx, for which the trace level is TC 1
- AP FExx (APPC application receive requests), for which the trace levels are TC 2 and Exc.

For more information about the trace points, see the CICS Trace Entries. For more information about using traces in problem determination, see the CICS Problem Determination Guide.
Chapter 94. Terminal error program

The terminal error program (DFHTEP) is invoked by the terminal abnormal condition program (DFHTACP) when an abnormal condition associated with a terminal or line occurs. The terminal error program (TEP) can be either of the following:

- The CICS-supplied sample TEP (DFHXTEP in source code form)
- A user-supplied TEP.

Design overview

The TEP analyzes the cause of the terminal or line error that has been detected by the terminal control program. The CICS-supplied version is designed to attempt basic and generalized recovery actions.

A user-supplied TEP can be used to enable processing to be performed whenever a communication system error is reported to CICS; for example, to analyze the error and accept or override the default actions set by DFHTACP.

When TEP processing is complete, control goes back to DFHTACP.

**Note:** Communication system errors (non-VTAM) are passed only to DFHTEP—not to the application programs.

Guidance information about TEP coding is given in the CICS Recovery and Restart Guide. Reference information about TEP coding is given in the CICS Customization Guide.

Modules

DFHTEP

Exits

No global user exit points are provided for this function.

Trace

No trace points are provided specifically for this function; however, DFHTACP provides trace entries immediately before and after calling the terminal error program (see “Chapter 92. Terminal abnormal condition program” on page 1191 for further details).
The timer domain provides interval timing and alarm clock services for CICS domains. These are processes that cause an action to occur at some predetermined future time. This service (called “notifying”) can be performed after a specific interval, at periodic intervals, at a specified time of day, or at a specific time of day every day.

The timer domain also provides date and time provision and conversion functions. This includes the facility to synchronize the CICS local time with the operating clock when the system operator has adjusted the time zone.

Timer domain’s specific gate

Table 107 summarizes the timer domain’s specific gate. It shows the level-1 trace point IDs of the modules providing the functions for the gate, the functions provided by the gate, and whether or not the functions are available through the exit programming interface (XPI).

Table 107. Timer domain’s specific gate

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>XPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>TISR</td>
<td>TI 0100</td>
<td>REQUEST_NOTIFY_INTERVAL</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>TI 0101</td>
<td>REQUEST_NOTIFY_TIME_OF_DAY</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CANCEL</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INQUIRE_EXPIRATION_TOKEN</td>
<td>NO</td>
</tr>
</tbody>
</table>

TISR gate, REQUEST_NOTIFY_INTERVAL function

The REQUEST_NOTIFY_INTERVAL function of the TISR gate is used to request the timer domain to notify the calling domain after a specified real interval of time. The calling domain can request a NOTIFY on a one-off basis or periodically, and can specify the type of NOTIFY to be expected.

Input parameters

**DOMAIN_TOKEN** is a token that is to be passed as a parameter on the NOTIFY call.

**STCK_INTERVAL** specifies an interval as a doubleword binary interval in stored clock (STCK) format, where bit 51 of the doubleword represents 1 microsecond.

**PERIODIC_NOTIFY** specifies whether the requested NOTIFY is to be repeated at the specified interval until canceled (YES), or is to be just a one-off NOTIFY (NO). It can have either of these values:

**YES** | **NO**

**NOTIFY_TYPE** specifies whether the attached task or the timer task is to be used to notify the calling domain after the specified interval of time. It can have either of these values:

**ATTACHED_TASK** | **TIMER_TASK**
Timer domain (TI)

[ATTACH_PRIORITY] defines the priority, in the range 0 through 255, at which the requested NOTIFY task is to be attached.

[ATTACH_TASK_TIMEOUT] defines the value, in seconds, of a wait in the attached task after which the dispatcher causes a time-out.

[ATTACH_MODE] is the optional TCB mode in which the attached NOTIFY task is to run.

[ORIGIN_DATE] defines the date from which the timer domain is to start the interval timing for this request. This parameter is mandatory if ORIGIN_TIME has been specified. It holds the origin date as MMDDYYYY.

[ORIGIN_TIME] defines the local time of day from which the timer domain is to start the interval timing for this request. The value in decimal digits is specified in the form HHMMSS:

- **HH**: Hours in the range 00 through 23
- **MM**: Minutes in the range 00 through 59
- **SS**: Seconds in the range 00 through 59.

ORIGIN_TIME defaults to the current time.

Output parameters

TIMER_TOKEN is the token that is returned by the timer domain. The timer token may be used to cancel the NOTIFY request.

RESPONSE is the domain’s response to the call. It can have any of these values:

- OK
- INVALID
- EXCEPTION
- DISASTER
- KERNERROR
- PURGED

[REASON] is returned when RESPONSE is INVALID. It has this value:

- INVALID_INTERVAL

TISR gate, REQUEST_NOTIFY_TIME_OF_DAY function

The REQUEST_NOTIFY_TIME_OF_DAY function of the TISR gate is used to inform the timer domain that an alarm call is required from the timer domain (that is, a NOTIFY) at the specified time of day. The calling domain can request a NOTIFY on a one-off basis or daily, and the type of NOTIFY to be expected.

Input parameters

DOMAIN_TOKEN is the token that is to be passed as a parameter on the NOTIFY call.

REQUESTED_TIME is the time of day at which the NOTIFY function is to be invoked. The value is specified in the form HHMMSS.

PERIODIC_NOTIFY specifies whether the requested NOTIFY is to be repeated every day at the requested time (YES), or is to be just a one-off NOTIFY (NO). It can have either of these values:

- YES
- NO
NOTIFY_TYPE
specifies whether the attached task or the timer task is to be used to notify
the calling domain after the specified interval of time. It can have either of
these values:
ATTACHED_TASK|TIMER_TASK

[ATTACH_PRIORITY]
defines the priority, in the range 0 through 255, at which the requested
NOTIFY task is to be attached.

[ATTACH_TASK_TIMEOUT]
defines the value, in seconds, of a wait in the attached task after which the
dispatcher causes a time-out.

[ATTACH_MODE]
is the optional TCB mode in which the attached NOTIFY task is to run.

Output parameters

TIMER_TOKEN
is the token that is returned by the timer domain.

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|INVALID|EXCEPTION|DISASTER|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is INVALID. It has this value:
TOO_LATE

TISR gate, CANCEL function
The CANCEL function of the TISR gate is used to cancel a timer request that has
already been initiated by one of these functions:
REQUEST_NOTIFY_INTERVAL
REQUEST_NOTIFY_TIME_OF_DAY

Input parameters

TIMER_TOKEN
is the token that was returned when the timer request was made.

Output parameters

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|INVALID|EXCEPTION|DISASTER|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>REQUEST_NOT_FOUND, TOO_LATE</td>
</tr>
</tbody>
</table>

TISR gate, INQUIRE_EXPIRATION_TOKEN function
The INQUIRE_EXPIRATION_TOKEN function of the TISR gate is used by the
dispatcher domain during its initialization.

Input parameters
None.
Timer domain (TI)

**Output parameters**

**EXPIRATION_TOKEN**

is a token used during initialization of the dispatcher domain.

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK|INVALID|EXCEPTION|DISASTER|KERNERROR|PURGED

---

**Timer domain’s generic gate**

Table 108 summarizes the timer domain’s generic gate. It shows the level-1 trace point IDs of the modules providing the functions for the gate, the functions provided by the gate, and the generic format for calls to the gate.

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMDM</td>
<td>TI 0001</td>
<td>INITIALIZE_DOMAIN</td>
<td>DMDM</td>
</tr>
<tr>
<td></td>
<td>TI 0002</td>
<td>QUIESCE_DOMAIN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TERMINATE_DOMAIN</td>
<td></td>
</tr>
</tbody>
</table>

You can find descriptions of these functions and their input and output parameters in the section dealing with the corresponding generic format, in format DMDM under "Domain manager domain’s generic formats" on page 377.

In initialization and quiesce processing, the timer domain performs only internal routines.

The timer domain does no termination processing.

---

**Timer domain’s generic format**

Table 109 describes the generic format owned by the timer domain and shows the function performed on the calls.

<table>
<thead>
<tr>
<th>Format</th>
<th>Calling module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>TISR</td>
<td>DFHTISR</td>
<td>NOTIFY</td>
</tr>
</tbody>
</table>

In the descriptions of the formats that follow, the “input” parameters are input not to timer domain, but to the domain being called by the timer. Similarly, the “output” parameters are output by the domain that was called by timer domain, in response to the call.

**TISR format, NOTIFY function**

The NOTIFY function of the TISR format is used by the timer domain itself to notify a domain after its requested interval or time has expired.

**Input parameters**

**DOMAIN_TOKEN**

is a token that is to be passed as a parameter on the NOTIFY call.
Output parameters

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK | INVALID | EXCEPTION | DISASTER | KERNEERROR | PURGED

### Modules

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHTIDM</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>INITIALIZE_DOMAIN</td>
</tr>
<tr>
<td></td>
<td>QUIESCE_DOMAIN</td>
</tr>
<tr>
<td></td>
<td>TERMINATE_DOMAIN</td>
</tr>
<tr>
<td>DFHTIDUF</td>
<td>Formats the timer domain’s control blocks</td>
</tr>
<tr>
<td>DFHTISR</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>REQUEST_NOTIFY_INTERVAL</td>
</tr>
<tr>
<td></td>
<td>REQUEST_NOTIFY_TIME_OF_DAY</td>
</tr>
<tr>
<td></td>
<td>CANCEL</td>
</tr>
<tr>
<td></td>
<td>INQUIRE_EXPIRATION_TOKEN</td>
</tr>
<tr>
<td>DFHTITRI</td>
<td>Interprets timer domain trace entries</td>
</tr>
</tbody>
</table>

### Exits

No global user exit points are provided in this domain.

### Trace

The point IDs for the timer domain are of the form TI xxxx; the corresponding trace levels are TI 1 and Exc.

For more information about the trace points, see the [CICS Trace Entries](#). For more information about using traces in problem determination, see the [CICS Problem Determination Guide](#)
Chapter 96. Trace control macro-compatibility interface

DFHTRP is responsible for handling all requests for trace services that are made by using the routine addressed by CSATRNAC in the CICS common system area (CSA).

Some parts of the CICS AP domain invoke DFHTRP to record trace information. This is achieved by use of the DFHTR, DFHTRACE, or DFHLFM macro.

DFHTRP converts all requests for recording trace entries into TRACE_PUT calls to the trace domain. All requests for changing the various trace flags that control tracing are converted into KEDD format calls to the kernel domain.

Design overview

The input to DFHTRP, set up by the macro used for the invocation or by the calling program directly, consists of the following TCA fields:

TCATRTR
The trace request byte. The bottom half byte has one of the following values:
2 User trace entry
3 An entry requested via DFHLFM on entry to a LIFO module
4 A system entry requested via DFHTR or DFHTRACE
5 An entry requested via DFHLFM on exit from a LIFO module.

TCATRID
The trace ID of the entry to be made. This is one byte X’n’n’. The resulting trace point ID is AP 00nn.

TCATRF1/TCATRF2
Two 4-byte fields to appear as FIELD A and FIELD B in the trace entry.

TCATRRSN
An 8-character field used by some entries to specify a resource name.

The following flags in the TCA and CSA are tested by DFHTRP before making the call to the trace domain (TRACE_PUT function):

CSATRMAS (X’80’ bit in CSATRMF1)
The trace master flag. This is off unless at least one of internal, auxiliary, or GTF trace is active.

TCANOTRC (X’40’ bit in TCAFLAGS)
This is set according to the TRACE (YES|NO) specification on the TRANSACTION definition for the transaction ID used to start this task. It allows suppression of all trace activity for specified transaction IDs.

X’80’ bit in TCATRMF
This is the user entry ‘single’ flag. It allows suppression of user trace entries for the associated task.

The process flow is as follows:
1. Test appropriate flags and exit if trace not required.
Trace control macro-compatibility

2. Execute data collection routine specific to trace ID in TCATRID to set up fields in trace entry.
3. Call TR domain with TRACE_PUT call to write the entry to the active destinations.
4. Invoke the storage violation trap (if this has been activated) by using the CSFE DEBUG transaction, or by using the CHKSTSK or CHKSTRM startup override. See the CICS Problem Determination Guide for information about the detection of storage violations.

Modules

DFHTRP

Exits

No global user exit points are provided for this function.

Trace

The following point IDs are provided for trace entries recording “trace on” and “trace off” calls to DFHTRP:
- AP 00FE, for trace turned on
- AP 00FF, for trace turned off.

There are no corresponding trace levels for these point IDs; that is, the trace entries are always produced.

For more information about the trace points, see the CICS Trace Entries. For more information about using traces in problem determination, see the CICS Problem Determination Guide.
Chapter 97. Trace domain (TR)

The trace domain is used by CICS system code and user application programs to record details of the sequence of events occurring in the system. The basic unit of information created for this purpose is called a trace entry. The trace domain can put trace entries to any combination of three possible destinations:

**INTERNAL trace**
a wraparound table in main storage in the CICS address space

**AUXILIARY trace**
a pair of CICS-controlled BSAM data sets used alternately

**Generalized trace facility (GTF) trace**
the user-defined destination for MVS GTF records.

Design overview

The trace domain consists of a set of modules that are used to record and manage trace information about internal, auxiliary, and GTF trace. The services of the trace domain are requested by making domain calls, described in "Domain calls" on page 1234. The modules that handle these domain calls are DFHTRDM, DFHTRPT, and DFHTRSR.

Certain sub-functions of the trace domain are required by more than one of these modules. These sub-functions are packaged together in the DFHTRSU module, and are invoked by domain subroutine calls, described in "Subroutine calls" on page 1238.

All processing directly related to the auxiliary trace data sets is carried out by the DFHTRAO module. DFHTRAO is loaded below the 16MB line so that it can run in 24-bit mode when calling BSAM and referencing the auxiliary trace data set data control block (DCB). The DFHTRAO functions are described in "DFHTRAO functions" on page 1239.

TRACE_PUT handling

For performance reasons, it is important to minimize the path length of a request to write a trace entry. This is achieved for most TRACE_PUT requests by handling them in module DFHTRPX, which runs as a subroutine of the domain that is requesting the trace.

DFHTRPX runs in a very restricted environment. It has no working storage and can make no calls out. Nevertheless, it can still handle the majority of TRACE_PUT requests. When DFHTRPX cannot handle a request, it passes control to the TRPT gate of the trace domain for module DFHTRPT to process the request.

DFHTRPX passes control to DFHTRPT in the following situations:
- CICS tracing to GTF is active.
- Transaction dump processing currently holds the trace lock while copying parts of the trace table to a local buffer.
- DFHTRSR currently holds the trace lock while processing the SET_INTERNAL_TABLE_SIZE function.
Trace domain (TR)

- CICS auxiliary trace is active and the requested entry does not fit in the current block, that is, a block write is required.
- The amount of data passed for tracing is larger than the trace domain limit (overlength entry).
- DFHTRPX’s recovery routine has been driven, probably because of a program check while moving data into the internal trace table.
- The FE global trap/trace exit (DFHTRAP) is active.

Locking

The trace domain handles TRACE_PUT requests from many MVS task control blocks (TCBs), and so requires a locking mechanism to prevent overlapping or simultaneous access to its control blocks. This is an MVS TCB lock and is provided by the LOCK and UNLOCK functions of the DFHKERN macro.

DFHTRPX does not acquire the trace lock. It uses “compare double and swap” (CDS) logic to serialize the allocation of space for trace entries in the internal trace table.

Selectivity

The overall trace master flag is logically a combination of the flags controlling internal, auxiliary, and GTF trace. It is owned by the trace domain, and both the kernel and the common system area (CSA) have their own copies that are kept up-to-date by calls from the trace domain.

The user trace master flag is owned by the AP domain. The system trace master flag and the standard and special component trace flags are owned by the kernel. None of these flags is referenced by the trace domain.

Domain calls

This section lists the process flows for the domain calls used for the trace domain services.

**DMDDM gate, PRE_INITIALIZE function**

1. Issue an MVS GETMAIN for the trace domain anchor block (TRA) and initialize it.
2. Acquire startup information from the parameter manager (PA) domain and set it in the TRA. The relevant startup parameters are INTTR, TRTABSZ, AUXTR, AUXTRSW, and GTFTR.
3. Call TRSU SET_UP_INTERNAL_TABLE to get and initialize the internal trace table.
4. Call TRSU GET_GTF_BUFFER to initialize CICS tracing to GTF.
5. Issue the KEDD ADD_GATE call for the DFHTRPT gate to inform the kernel that the trace domain is available.
6. If internal trace or GTF trace is started, turn on the trace master flags in the TRA, the kernel, and the CSA.

**DMDDM gate, INITIALIZE_DOMAIN function**

1. If required, call TRSR ACTIVATE_TRAP to active the FE global trap/trace exit, DFHTRAP.
2. If required, call TRSR START_AUXILIARY_TRACE to start auxiliary trace on DFHAUXT.
DMDM gate, QUIESCE_DOMAIN function
Do nothing.

DMDM gate, TERMINATE_DOMAIN function
If auxiliary trace is active, call TRSR STOP_AUXILIARY_TRACE.

KETI gate, NOTIFY_RESET function
Call KETI CONVERT_TO_STCK_FORMAT to get the new STCK value for the last local midnight, and store this in the TRA.

TRPT gate, TRACE_PUT function
1. Acquire the trace lock.
2. Calculate the length of the required entry.
3. If the entry does not fit in the current trace block (TRBL) and auxiliary trace is active, call TRSU WRITE_AUX_BUFFER.
4. Use “compare double and swap” (CDS) to update pointer and available length for next entry in the TRA.
5. Build the entry in allocated space.
6. If GTF trace is required, issue the GTRACE macro to write an entry to GTF, and if the entry is more than 256 bytes, split it into multiple entries.
7. If the FE global trap/trace exit, DFHTRAP, has been activated as a result of using the CSFE DEBUG transaction, or specifying the TRAP=ON system initialization parameter, invoke the exit. See the CICS Problem Determination Guide for details of DFHTRAP.
8. Release the trace lock.

TRSR gate, SET_INTERNAL_TABLE_SIZE function
1. If the call is from the parameter manager (during initialization), set the required size in the TRA and return.
2. Acquire the trace lock.
3. If auxiliary trace is active, call TRSU WRITE_AUX_BUFFER to write the current TRBL.
4. If the new table size is smaller, free part of the old table and reset chaining and pointers.
5. If a larger table is required, free all but 16KB (KB equals 1024 bytes) of the old table. Call TRSU SET_UP_INTERNAL_TABLE. If this completes correctly, free the 16KB that was kept back. If it does not work, make the 16KB piece the new table.
6. Release the trace lock.

TRSR gate, START_INTERNAL_TRACE function
1. Set the required status in the TRA.
2. If the call is from the parameter manager (during initialization), return.
3. If required, change the kernel and CSA copies of the trace master flag.

TRSR gate, STOP_INTERNAL_TRACE function
1. Set the required status in the TRA.
2. If the call is from the parameter manager (during initialization), return.
3. If required, change the kernel and CSA copies of the trace master flag.

TRSR gate, INQUIRE_INTERNAL_TRACE function
Get the internal status and internal table size from the TRA.
**Trace domain (TR)**

**TRSR gate, START_AUXILIARY_TRACE function**
1. If the call is from the parameter manager (during initialization), set the status in the TRA and return.
2. If already started, return immediately.
3. If auxiliary trace is currently stopped (rather than paused):
   a. Issue an MVS GETMAIN for an auxiliary trace buffer, DCB, and DECB storage.
   b. Issue LDLD ACQUIRE_PROGRAM for DFHTRAO.
   c. Call DFHTRAO to OPEN the auxiliary trace data set.
4. Acquire the trace lock.
5. Skip the current TRBL pointer in the TRA to the next TRBL to avoid entries from before start appearing in the auxiliary trace.
6. Release the trace lock.
7. Set the auxiliary trace status in the TRA to started.
8. If required, change the kernel and CSA copies of the trace master flag.

**TRSR gate, STOP_AUXILIARY_TRACE function**
1. If the call is from the parameter manager (during initialization), set the status in the TRA and return.
2. If already stopped, return immediately.
3. Acquire the trace lock.
4. If auxiliary trace is started (rather than paused), call TRSU WRITE_AUX_BUFFER to output the current TRBL to the auxiliary trace data set, and move the current TRBL pointer in the TRA to the next TRBL.
5. Call TRSU WRITE_AUX_BUFFER to write an end-of-file indication on the auxiliary trace data set.
6. Call DFHTRAO to ensure (CHECK) that output is complete.
7. Call DFHTRAO to CLOSE the auxiliary trace data set.
8. Call TRSU TERMINATE_AUXILIARY_TRACE.
9. Release the trace lock.
10. Issue LDLD RELEASE_PROGRAM for DFHTRAO.

**TRSR gate, PAUSE_AUXILIARY_TRACE function**
1. If auxiliary trace is stopped, return with error.
2. If auxiliary trace is paused, return ‘OK’.
3. Acquire the trace lock.
4. Call TRSU WRITE_AUX_BUFFER to output the current TRBL to the auxiliary trace data set, and move the current TRBL pointer in the TRA to the next TRBL.
5. Release the trace lock.
6. Change the kernel and CSA copies of the trace master flag if required.

**TRSR gate, SET_AUX_TRACE_AUTOSWITCH function**
Set the new autoswitch status in the TRA.

**TRSR gate, SWITCH_AUXILIARY_EXTENTS function**
1. If auxiliary trace is started or paused:
   a. Acquire the trace lock.
   b. Call TRSU WRITE_AUX_BUFFER to write an end-of-file indication on the auxiliary trace data set.
c. Call DFHTRAO to ensure (CHECK) that output is complete.
d. Call DFHTRAO to close the auxiliary trace data set.

2. Change the name of the current extent in the TRA from DFHAUXT to DFHBUXT or from DFHBUXT to DFHAUXT.

3. If auxiliary trace is started or paused:
   a. Call DFHTRAO to OPEN the auxiliary trace data set.
   b. Release the trace lock.

TRSR gate, INQUIRE_AUXILIARY_TRACE function
Get the auxiliary trace status, current extent name, and autoswitch status from the TRA.

TRSR gate, START_GTF_TRACE function
1. If the call is from the parameter manager (during initialization), set the required status in the TRA and return.
2. If already started, return immediately.
3. Call TRSU GET_GTF_BUFFER.
4. Set the status in the TRA to started.
5. If required, change the kernel and CSA copies of the trace master flag.

TRSR gate, STOP_GTF_TRACE function
1. Set the status in the TRA to stopped.
2. If the call is from the parameter manager (during initialization), return.
3. If required, change the kernel and CSA copies of the trace master flag.
4. If the GTF buffer is present:
   a. Acquire the trace lock.
   b. Issue an MVS FREEMAIN for the GTF buffer.
   c. Release the trace lock.

TRSR gate, INQUIRE_GTF_TRACE function
Get the GTF status from the TRA.

TRSR gate, ACTIVATE_TRAP function
1. If the call is from the parameter manager (during initialization), set the required status in the TRA and return.
2. If the trap is already active, check whether it is marked unusable because a program check occurred while the trap was in control:
   a. If the trap is unusable, return with error.
   b. If the trap is usable, set the required status in the TRA and return.
3. Issue LDLD ACQUIRE_PROGRAM for DFHTRAP.
4. Issue an MVS GETMAIN for the DFHTRAP work area (TRGTW).
5. Acquire the trace lock.
6. Check whether another task has activated the trap:
   a. If the trap is not active, update the trap status in the TRA, release the trace lock, and return.
   b. If the trap has been activated by another task, release the trace lock, issue LDLD RELEASE_PROGRAM for DFHTRAP, issue an MVS FREEMAIN for the DFHTRAP work area, and return.
Trace domain (TR)

TRSR gate, DEACTIVATE_TRAP function
1. If the call is from the parameter manager (during initialization), set the required status in the TRA and return.
2. If the trap is not active, return.
3. Acquire the trace lock.
4. Update the trap status in the TRA.
5. Release the trace lock.
6. Issue LDLD RELEASE_PROGRAM for DFHTRAP.
7. Issue an MVS FREEMAIN for the DFHTRAP work area.

Subroutine calls
This section lists the process flows for the domain subroutine calls used for the trace domain sub-functions.

TRSU format, WRITE_AUX_BUFFER function
1. If output to the auxiliary trace data set is pending, call DFHTRAO with a CHECK request to allow output to complete.
2. If there was no output pending or the output completed successfully, call DFHTRAO to write the current TRBL, and return.

If an 'end of extent' was encountered on the BSAM CHECK:
3. Call DFHTRAO to close the auxiliary trace data set.
4. If autoswitch is not required:
   a. Issue an MVS FREEMAIN for the auxiliary trace buffer, DCB, and DECB.
   b. Set auxiliary trace status in the TRA to stopped.
   c. Change the kernel and CSA copies of the trace master flag if required.
   d. Return.
5. If autoswitch next is specified, change to autoswitch off.
6. Change the name of the current extent in the TRA from DFHAUXT to DFHBUXT or from DFHBUXT to DFHAUXT.
7. Call DFHTRAO to OPEN the auxiliary trace data set.
8. Call DFHTRAO with a WRITE request to rewrite the block that caused the end-of-extent.
9. Go back to the top of this function’s processing to issue the write that was originally requested in this call.

TRSU format, TERMINATE_AUXILIARY_TRACE function
1. Issue an MVS FREEMAIN for the auxiliary trace buffer, DCB, and DECB.
2. Set the auxiliary trace status in the TRA to stopped.
3. If required, change the kernel and CSA copies of the trace master flag.

TRSU format, GET_GTF_BUFFER function
1. Issue an MVS GETMAIN for the GTF buffer.
2. Save the address in the TRA.

TRSU format, SET_UP_INTERNAL_TABLE function
1. Issue an MVS V-type GETMAIN for the required size.
2. Initialize all TRBL headers within the acquired area.
DFHTRAO functions

This section lists the process flows for the DFHTRAO functions for auxiliary trace data sets.

DFHTRAO, OPEN function
1. If the DCB indicates already open, return ‘OK’.
2. Issue the BSAM OPEN macro.

DFHTRAO, CLOSE function
1. If the DCB indicates already closed, return ‘OK’.
2. Issue the BSAM CLOSE macro.

DFHTRAO, CHECK function
1. Issue the BSAM CHECK macro.
2. If an end-of-extent is caused by the write for which this CHECK is issued, the DCB ABEND exit is driven and causes DFHTRAO to return an end-of-extent indication to the caller.
3. Clear output pending status in TRA.

DFHTRAO, WRITE function
1. Move the specified TRBL to the auxiliary trace buffer.
2. Issue the BSAM WRITE macro.
3. Set output pending status in the TRA.

Trace domain’s specific gates

Table 110 summarizes the trace domain’s specific gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and whether or not the functions are available through the exit programming interface (XPI).

Table 110. Trace domain’s specific gates

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>XPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRPT</td>
<td>None</td>
<td>TRACE_PUT</td>
<td>YES</td>
</tr>
<tr>
<td>TRSR</td>
<td>TR 0201</td>
<td>SET_INTERNAL_TABLE_SIZE</td>
<td>NO</td>
</tr>
<tr>
<td>TRSR</td>
<td>TR 0202</td>
<td>START_INTERNAL_TRACE</td>
<td>NO</td>
</tr>
<tr>
<td>TRSR</td>
<td></td>
<td>STOP_INTERNAL_TRACE</td>
<td>NO</td>
</tr>
<tr>
<td>TRSR</td>
<td></td>
<td>INQUIRE_INTERNAL_TRACE</td>
<td>NO</td>
</tr>
<tr>
<td>TRSR</td>
<td></td>
<td>START_AUXILIARY_TRACE</td>
<td>NO</td>
</tr>
<tr>
<td>TRSR</td>
<td></td>
<td>STOP_AUXILIARY_TRACE</td>
<td>NO</td>
</tr>
<tr>
<td>TRSR</td>
<td></td>
<td>PAUSE_AUXILIARY_TRACE</td>
<td>NO</td>
</tr>
<tr>
<td>TRSR</td>
<td></td>
<td>SET_AUX_TRACE_AUTOSWITCH</td>
<td>NO</td>
</tr>
<tr>
<td>TRSR</td>
<td></td>
<td>SWITCH_AUXILIARY_EXTENTS</td>
<td>NO</td>
</tr>
<tr>
<td>TRSR</td>
<td></td>
<td>INQUIRE_AUXILIARY_TRACE</td>
<td>NO</td>
</tr>
<tr>
<td>TRSR</td>
<td></td>
<td>START_GTF_TRACE</td>
<td>NO</td>
</tr>
<tr>
<td>TRSR</td>
<td></td>
<td>STOP_GTF_TRACE</td>
<td>NO</td>
</tr>
<tr>
<td>TRSR</td>
<td></td>
<td>INQUIRE_GTF_TRACE</td>
<td>NO</td>
</tr>
<tr>
<td>TRSR</td>
<td></td>
<td>ACTIVATE_TRAP</td>
<td>NO</td>
</tr>
<tr>
<td>TRSR</td>
<td></td>
<td>DEACTIVATE_TRAP</td>
<td>NO</td>
</tr>
</tbody>
</table>

TRPT gate, TRACE_PUT function

This function is invoked to write a trace entry to the active trace destinations.
Trace domain (TR)

Input parameters

POINT_ID

is a number, unique within the calling domain, that identifies the trace entries made from this call.

[DATA1] through [DATA7]

are BLOCK descriptions of up to seven areas to be included in the data section of the trace entry. They appear in numerical order in the entry, each preceded by a 2-byte length field.

The maximum total length of data that can be traced in one call is as described below:

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length of trace table block</td>
<td>4096</td>
</tr>
<tr>
<td>less length of trace table block header</td>
<td>- 24</td>
</tr>
<tr>
<td>less length of trace entry header</td>
<td>- 32</td>
</tr>
<tr>
<td>------------------------------------------------------</td>
<td>--------</td>
</tr>
<tr>
<td>Maximum space for data + length fields</td>
<td>4040</td>
</tr>
</tbody>
</table>

For each DATA field specified, 2 bytes must be subtracted to allow for the length field.

Maximum space for actual data = 4040 - (2 * n)

where 'n' is the number of DATA fields specified.

[RETURN_ADDR]

is used by DFHTRP to give a return address in the trace entry from the calling module rather than in DFHTRP.

Output parameters

RESPONSE

is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

Note: No response is returned when the TRACE_PUT request is handled by module DFHTRPX without involving the trace domain.

TRSR gate, SET_INTERNAL_TABLE_SIZE function

The SET_INTERNAL_TABLE_SIZE function of the TRSR gate is used to change the size of the internal trace table during a CICS run.

Input parameters

TABLE_SIZE

is the required table size, specified as a number of KB (KB equals 1024 bytes). This is rounded up to the nearest multiple of 4KB. The lower limit is 16KB. The upper limit is set only by the amount of storage available. If the table is being made larger, the existing table is freed and a variable MVS GETMAIN issued for the required size. The actual length of the new table can be determined by issuing an INQUIRE_INTERNAL_TRACE command. If the table is being made smaller, part of the existing table is freed.

Output parameters

RESPONSE

is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]

is returned when RESPONSE is INVALID. Possible values are:
**TRSR gate, START_INTERNAL_TRACE function**

The START_INTERNAL_TRACE function of the TRSR gate is used to activate tracing to the internal trace table.

**Input parameters**
None.

**Output parameters**

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

**TRSR gate, STOP_INTERNAL_TRACE function**

The STOP_INTERNAL_TRACE function of the TRSR gate is used to deactivate tracing to the internal trace table.

**Input parameters**
None.

**Output parameters**

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

**TRSR gate, INQUIRE_INTERNAL_TRACE function**

The INQUIRE_INTERNAL_TRACE function of the TRSR gate is used to return the status of the internal trace and the current size of the internal trace table.

**Input parameters**
None.

**Output parameters**

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

**TABLE_SIZE**

is the size of the current internal trace table in KB (KB equals 1024 bytes).

**INTERNAL_STATUS**

indicates whether internal trace is active (STARTED) or inactive (STOPPED).
Trace domain (TR)

**TRSR gate, START_AUXILIARY_TRACE function**

The START_AUXILIARY_TRACE function of the TRSR gate is used to open the current auxiliary trace extent (if it is closed) and start tracing to it.

**Input parameters**
None.

**Output parameters**

RESPONSE is the domain’s response to the call. It can have any of these values:

- OK
- EXCEPTION
- DISASTER
- INVALID
- KERNERROR
- PURGED

**[REASON]**

is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>CANT_GET_AUX_BUFFER, DFHTRAO_NOT_AVAILABLE, OPEN_FAILED</td>
</tr>
</tbody>
</table>

**Note:**

- CANT_GET_AUX_BUFFER indicates that MVS had insufficient free storage to satisfy the request for a buffer below the 16MB line.
- DFHTRAO_NOT_AVAILABLE indicates that the request to the CICS loader to acquire the auxiliary trace output program, DFHTRAO, has failed.
- OPEN_FAILED indicates that the MVS open of the auxiliary trace data set has failed.

**TRSR gate, STOP_AUXILIARY_TRACE function**

The STOP_AUXILIARY_TRACE function of the TRSR gate is used to stop auxiliary tracing and close the currently active auxiliary trace extent.

**Input parameters**
None.

**Output parameters**

RESPONSE is the domain’s response to the call. It can have any of these values:

- OK
- EXCEPTION
- DISASTER
- INVALID
- KERNERROR
- PURGED

**TRSR gate, PAUSE_AUXILIARY_TRACE function**

The PAUSE_AUXILIARY_TRACE function of the TRSR gate is used to stop auxiliary tracing without closing the currently active extent.

**Input parameters**
None.

**Output parameters**

RESPONSE is the domain’s response to the call. It can have any of these values:

- OK
- EXCEPTION
- DISASTER
- INVALID
- KERNERROR
- PURGED

**[REASON]**

is returned when RESPONSE is EXCEPTION. It has this value:

- AUX_TRACE_STOPPED

meaning that the pause is allowed only if auxiliary trace is not stopped.
**TRSR gate, SET_AUX_TRACE_AUTOSWITCH function**

The SET_AUX_TRACE_AUTOSWITCH function of the TRSR gate is used to allow the autoswitch facility for the CICS auxiliary trace data set to be enabled or disabled.

**Input parameters**

AUTOSWITCH_STATUS

Indicates whether or not an automatic switch to the inactive CICS auxiliary extent is to occur once only when the current extent fills up, or that such automatic switching should occur “continuously” whenever the current extent fills up. It can have any one of these values:

OFF|ONCE|CONTINUOUS

**Output parameters**

RESPONSE

is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]

is returned when RESPONSE is EXCEPTION. It has this value:

INVALID_AUTOSWITCH_STATUS

meaning that an incorrect value was passed for AUTOSWITCH_STATUS.

**TRSR gate, SWITCH_AUXILIARY_EXTENTS function**

The SWITCH_AUXILIARY_EXTENTS function of the TRSR gate allows switching from one auxiliary trace extent to the other.

**Input parameters**

None.

**Output parameters**

RESPONSE

is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]

is returned when RESPONSE is EXCEPTION. It has this value:

OPEN_FAILED

meaning that the attempt to open the new auxiliary extent failed.

**TRSR gate, INQUIRE_AUXILIARY_TRACE function**

The INQUIRE_AUXILIARY_TRACE function of the TRSR gate is used to return the current state of the auxiliary trace.

**Input parameters**

None.

**Output parameters**

AUXILIARY_STATUS

Indicates the current status of auxiliary trace. It can have any one of these values:

STARTED|STOPPED|PAUSED
Trace domain (TR)

**EXTENT**
indicates the currently active CICS auxiliary trace extent; that is, the extent that is already in use or is used if CICS auxiliary tracing is started. It can have either of these values:
DFHAUXT|DFHBUXT

**AUTOSWITCH_STATUS**
Indicates whether or not an automatic switch to the inactive CICS auxiliary extent is to occur once only when the current extent fills up, or that such automatic switching should occur “continuously” whenever the current extent fills up. It can have any one of these values:
OFF|ONCE|CONTINUOUS

**RESPONSE**
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

### TRSR gate, START_GTF_TRACE function
The START_GTF_TRACE function of the TRSR gate is used to start the tracing of CICS activity to GTF. It is the responsibility of the user to ensure that GTF has been started in MVS with at least TRACE=USR. If it has not, CICS issues the GTF calls but they are ignored by GTF.

**Input parameters**
None.

**Output parameters**

**RESPONSE**
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**[REASON]**
is returned when RESPONSE is EXCEPTION. It has this value:
CANT_GET_GTF_BUFFER
meaning that there was insufficient storage for a buffer to be used in constructing continuation records when an individual entry is longer than 256 bytes.

### TRSR gate, STOP_GTF_TRACE function
The STOP_GTF_TRACE function of the TRSR gate is used to stop tracing of CICS activity to GTF.

**Input parameters**
None.

**Output parameters**

**RESPONSE**
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

### TRSR gate, INQUIRE_GTF_TRACE function
The INQUIRE_GTF_TRACE function of the TRSR gate is used to return the current state of the GTF trace.
Input parameters
None.

Output parameters
GTF_STATUS indicates whether CICS tracing to GTF is active (STARTED) or inactive (STOPPED).

RESPONSE is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

TRSR gate, ACTIVATE_TRAP function
The ACTIVATE_TRAP function of the TRSR gate is used to activate the FE global trap/trace exit (DFHTRAP).

Input parameters
None.

Output parameters
RESPONSE is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>DFHTRAP_NOT_FOUND, DFHTRAP_UNUSABLE</td>
</tr>
</tbody>
</table>

Note: DFHTRAP_NOT_FOUND indicates that the request to the CICS loader to acquire the FE global trap/trace exit program, DFHTRAP, has failed.

DFHTRAP_UNUSABLE indicates that the trap was already active, but marked as unusable because a program check had previously occurred when DFHTRAP was in control.

TRSR gate, DEACTIVATE_TRAP function
The DEACTIVATE_TRAP function of the TRSR gate is used to deactivate the FE global trap/trace exit (DFHTRAP).

Input parameters
None.

Output parameters
RESPONSE is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED
Trace domain (TR)

Trace domain’s generic gates

Table 111 summarizes the trace domain’s generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 111. Trace domain’s generic gates

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMDM</td>
<td>TR 0001</td>
<td>PRE_INITIALIZE</td>
<td>DMDM</td>
</tr>
<tr>
<td></td>
<td>TR 0002</td>
<td>INITIALIZE_DOMAIN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>QUIESCE_DOMAIN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TERMINATE_DOMAIN</td>
<td></td>
</tr>
<tr>
<td>KETI</td>
<td>TR 0201</td>
<td>NOTIFY_RESET</td>
<td>KETI</td>
</tr>
<tr>
<td></td>
<td>TR 0202</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For descriptions of these functions and their input and output parameters, refer to the sections dealing with the corresponding generic formats:

Functions and parameters

- Format DMDM—“Domain manager domain’s generic formats” on page 377
- Format KETI—“Kernel domain’s generic formats” on page 752

In preinitialization processing, the trace domain establishes the initial tracing status:

- A suitably sized internal trace table is created.
- If internal tracing or GTF tracing is required, set on the trace master flag.
- If required, start internal tracing and CICS GTF tracing.
- As required, set the auxiliary tracing switch status to ‘started’ or ‘stopped’.

The information always comes from the system initialization parameters—trace domain is always cold started.

In initialization processing, the trace domain starts auxiliary tracing if it is required.

The trace domain does no quiesce processing.

In termination processing, the trace domain stops auxiliary tracing if it is active.

Control blocks

Figure 105 on page 1247 shows the control blocks associated with the trace domain.
**TR domain anchor block (TRA).**
There is one TRA in the system. It contains all status information relating to the trace domain and also pointers to the other trace domain control blocks.

**Internal trace table.**
An area of virtual storage above the 16MB line used for recording trace entries.

**TR block (TRBL).**
The internal trace table consists of a number of TRBLs chained in a loop. They are each 4096 bytes long. Each block contains a standard header and a sequence of variable-length trace entries.

**Trace entry (TREN).**
All trace entries consist of a header together with any data specified on the call. The length of each trace entry is in the range 32 through 4072 bytes.

**TR auxiliary trace data set DCB, DECB, and buffer.**
During the auxiliary trace start process, an MVS GETMAIN is issued to acquire storage below the 16MB line for these areas. Their addresses are kept in the TRA. The storage is released when auxiliary trace is stopped.

**GTF buffer.**
During the GTF trace start process, an MVS GETMAIN is issued to acquire storage above the 16MB line for this area. It is 256 bytes long, and its address is kept in the TRA. The storage is released when GTF trace is stopped. The buffer is used when splitting large entries (more than 256 bytes) into 256-byte pieces to be written to GTF. This is done because GTF has a length restriction of 256 bytes.

**Global trap/trace exit work area (TRGTW).**
When the FE global trap/trace exit (DFHTRAP) is activated, an MVS GETMAIN is issued to acquire storage above the 16MB line for the TRGTW. This area contains a register save area and all working storage...
Trace domain (TR)

associated with DFHTRAP, including the parameter list passed to the exit program. Its address is kept in the TRA. The storage is released when the trap is deactivated.

See the CICS Data Areas manual for a detailed description of these control blocks.

Modules

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHTRDM</td>
<td>Processes requests to the DMDM gate of the trace domain. Part of the DFHSIP load module.</td>
</tr>
<tr>
<td>DFHTRPT</td>
<td>Processes requests to the TRPT gate of the trace domain. Part of the DFHSIP load module.</td>
</tr>
<tr>
<td>DFHTRPX</td>
<td>Processes, within the calling domain, all TRACE_PUT requests that do not require special handling. Part of the DFHSIP load module.</td>
</tr>
<tr>
<td>DFHTRSR</td>
<td>Processes requests to the TRSR and KETI gates of the trace domain. Part of the DFHSIP load module.</td>
</tr>
<tr>
<td>DFHTRSU</td>
<td>Processes domain subroutine requests of format TRSU. Part of the DFHSIP load module.</td>
</tr>
<tr>
<td>DFHTRAO</td>
<td>Auxiliary trace output subroutines for interfacing with BSAM. Loaded separately below the 16MB line when auxiliary trace is started.</td>
</tr>
<tr>
<td>DFHTRAP</td>
<td>FE global trap/trace exit program. Loaded separately above the 16MB line when the trap is activated.</td>
</tr>
</tbody>
</table>

Copy books

<table>
<thead>
<tr>
<th>Copy book</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHTRADS</td>
<td>Contains the definition of the parameter list passed to DFHTRAP.</td>
</tr>
<tr>
<td>DFHTRDS</td>
<td>Contains the definitions of the TRA and TRBL.</td>
</tr>
<tr>
<td>DFHTREN</td>
<td>Contains the definition of the trace entry (TREN) format.</td>
</tr>
</tbody>
</table>

Exits

No global user exit points are provided in this domain.

Trace

The point IDs for the trace domain are of the form TR xxxx; the corresponding trace levels are TR 1 and Exc.

For more information about the trace points, see the CICS Trace Entries. For more information about using traces in problem determination, see the CICS Problem Determination Guide.

Dumps

A formatted system dump contains (depending on the options specified on the TR keyword):

- TR anchor block with interpretation
- Auxiliary trace data set DCB if data set open
• Auxiliary trace data set DECB if data set open
• Auxiliary trace buffer if data set open
• Internal trace table in abbreviated format
• Internal trace table in full format.

System dumps requested by the trace domain fall into two categories:

**Dump code TRnnnn**

These dump codes are preceded by a console message, DFHTRnnnn. See the [CICS Messages and Codes](#) manual for details.

**Dump code KERNDUMP**

At many points in its processing, the trace domain cannot issue domain calls because they would lead to further trace calls and possible recursion of the error. In these circumstances, the trace domain uses MVS WTO to write a console message and the kernel dump function to take a system dump. All such dumps have dump code KERNDUMP. The message numbers for which this occurs are DFHTR0105, DFHTR0114, DFHTR0115, and DFHTR0116. See the [CICS Messages and Codes](#) manual for more details.
Trace domain (TR)
Chapter 98. Trace formatting

There are three possible destinations for CICS trace entries:

**Internal**
To main storage in the CICS region

**Auxiliary**
To a BSAM data set managed by CICS

**GTF**
To the MVS-defined destination for generalized trace facility (GTF) records.

This section describes the code used to interpret and format CICS trace entries from all of these destinations when they are processed offline.

For more information about using traces in problem determination, see the CICS Trace Entries.

In this context, “formatting” is used to mean the overall process of producing a report, suitable for viewing or printing, from trace data in a dump or trace data set. “Interpretation” is the process of taking just the point ID and the data fields from a trace entry and producing a character string describing what the entry represents.

There are four environments for trace formatting:

- Internal trace in transaction dump
- Internal trace in system dump
- Printing auxiliary trace data set
- Printing GTF trace data set or processing GTF records in an SDUMP.

### Table 112. CICS trace formatting summary

<table>
<thead>
<tr>
<th>CICS trace type</th>
<th>Transaction dump printout</th>
<th>System dump printout</th>
<th>Auxiliary trace printout</th>
<th>GTF trace printout</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data set</td>
<td>DFHDMPx</td>
<td>SYS1.DUMPnn</td>
<td>DFHxUXT</td>
<td>SYS1.DUMPnn or SYS1.TRACE</td>
</tr>
<tr>
<td>Controlling program</td>
<td>DFHDU530</td>
<td>DFHTRDUF</td>
<td>DFHTRPRA</td>
<td>DFHTRPRG</td>
</tr>
<tr>
<td>Load module name</td>
<td>DFHDU530</td>
<td>DFHPD530</td>
<td>DFHTU530</td>
<td>AMDUSREF (alias DFHTR530)</td>
</tr>
</tbody>
</table>

### Design overview

The controlling program (DFHDU530, DFHTRDUF, DFHTRPRA, or DFHTRPRG) is responsible for acquiring the trace formatting control area (TRFCA), which is used for communication between the different routines.

As far as possible, the necessary code is constructed of routines that can run in all four environments. Subroutines required by the common code that cannot
Trace formatting

themselves be common (such as the line print subroutine) have their addresses placed in the TRFCA by the controlling program.

The controlling routines are:

**DFHDU530**
The dump utility program used to print transaction dumps. Invokes DFHTRFPB for each internal table block.

**DFHTRDUF**
The system dump formatting routine for the trace domain. Invokes DFHTRFPB for each internal table block.

**DFHTRPRA**
The main routine of the trace utility program DFHTU530 used to print an auxiliary trace data set. Invokes DFHTRFPP to encode selective print parameters. Invokes DFHTRFPB for each auxiliary trace block.

**DFHTRPRG**
The main routine of the GTF format appendage for CICS entries (format ID X'EF') AMDUSREF (alias DFHT530). Invokes DFHTRFPP to encode selective print parameters. Invokes DFHTRFFE for each trace entry.

A noncommon subroutine required in all four environments is:

**TRFPRL**
Print a specified character buffer. This is contained in the controlling program.

The common routines required in more than one environment are:

**DFHTRFPP**
Process parameters. Passed a character string, encodes the string as selective print parameters into the TRFCA (for DFHTRPRA and DFHTRPRG only). See the [CICS Operations and Utilities Guide](#) for details of the selective print parameters.

**DFHTRFPB**
Process block. Processes a trace block from a dump or auxiliary trace data set, calling DFHTRFFE for each entry in the block.

**DFHTRFFE**
Format entry. Passed a trace entry, it calls DFH{xx}TRI, TRFPRL, and DFHTRFFD to produce the formatted entry.

**DFHTRFFD**
Format data. To format and print the trace data fields of a particular entry in hex and character form. Calls TRFPRL to print each line.

**DFH{xx}TRI**
The interpretation routine for the xx domain. Builds the interpretation string for a particular entry given the trace point ID and the data fields from the entry. The AP domain routine DFHAPTRI calls one of the interpretation routines DFHAPTRx. Each of these is responsible for a functional component of the AP domain.

**DFHTRIB**
The interpretation build program. Adds printable data to the interpretation buffer in the TRFCA as requested by the interpretation routine.

**DFHCDCON**
The interpretation of some trace entries requires analysis of domain call
parameter lists. Converts a hexadecimal parameter list into a printable list of keywords. If the resulting interpretation string would have been more than 1024 bytes long if all keywords were included, the warning '<<INTERPRETATION OVERFLOWED>>' is printed with the string.

**DFHxxyyT**

The data file for an `xxyy` format parameter list that is used by DFHCDCON to translate the hexadecimal parameter list into a printable list of keywords.

The components of the trace formatting function are shown in *Figure 106*.

---

**Segmented entries on GTF**

GTF entries with the CICS format ID X’EF’ are written from parts of CICS that run asynchronously with the mainline code, as well as from the trace domain itself. The source of the entry is identified by the type byte in TREN_TYPE in the entry header. See DFHTREN in the [CICS Data Areas](https://www.example.com) manual for a full description of the trace entry header.

<table>
<thead>
<tr>
<th>Type</th>
<th>Source of entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>TR domain</td>
</tr>
<tr>
<td>01</td>
<td>not used</td>
</tr>
<tr>
<td>02</td>
<td>DFHMNSVC</td>
</tr>
<tr>
<td>03</td>
<td>'normal' CICS VTAM exit</td>
</tr>
<tr>
<td>04</td>
<td>CICS VTAM LERAD/SYNAD exit</td>
</tr>
<tr>
<td>05</td>
<td>CICS VTAM TPEND exit</td>
</tr>
<tr>
<td>06</td>
<td>CICS VTAM HPO exit</td>
</tr>
<tr>
<td>07</td>
<td>CICS VTAM HPO LERAD/SYNAD exit</td>
</tr>
</tbody>
</table>

For trace formatting, the different types run on different MVS threads. Because CICS entries can be split into several GTF entries due to the 256-byte restriction on
Trace formatting

GTF entry length, it is possible that header and continuation entries of the different types may be interleaved on the GTF data set. DFHTRPRG allows for this by having 4KB buffers for each type in which it can reconstruct segmented entries. This is made all the more relevant when it is recognized that there could be several CICS regions writing to the GTF data set at the same time. Not only may different types become interleaved, but also records of the same type but from different CICS regions. For each type there can be up to five 4KB buffers for reconstructing the segmented entries to ensure that all the entries for any region are formatted completely and correctly. This makes the segmenting of the entries transparent in a formatted GTF trace, although they appear in order of completion and so may be out of time sequence.

Control blocks

The trace formatting control area (TRFCA) is used as a communication area between the routines that go to make up each of the four trace formatting load modules. See the CICS Data Areas manual for details of DFHTRFCA.

Modules

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Controlling programs</td>
<td></td>
</tr>
<tr>
<td>DFHDUS30</td>
<td>Internal trace in transaction dump</td>
</tr>
<tr>
<td>DFHTRDUF</td>
<td>Internal trace in system dump</td>
</tr>
<tr>
<td>DFHTRPRA</td>
<td>Auxiliary trace</td>
</tr>
<tr>
<td>DFHTRPRG</td>
<td>GTF trace</td>
</tr>
<tr>
<td>Common routines</td>
<td></td>
</tr>
<tr>
<td>DFHTRFPB</td>
<td>Process trace block</td>
</tr>
<tr>
<td>DFHTRFPP</td>
<td>Process selective print parameters</td>
</tr>
<tr>
<td>DFHTRFFE</td>
<td>Format trace entry</td>
</tr>
<tr>
<td>DFHTRFFD</td>
<td>Format data from entry</td>
</tr>
<tr>
<td>DFHTRIB</td>
<td>Interpretation build routine</td>
</tr>
<tr>
<td>DFHCDCON</td>
<td>Parameter list decode routine</td>
</tr>
<tr>
<td>Trace interpretation routines</td>
<td></td>
</tr>
<tr>
<td>DFHAPTRA</td>
<td>MRO entries</td>
</tr>
<tr>
<td>DFHAPTRB</td>
<td>XRF entries</td>
</tr>
<tr>
<td>DFHAPTRC</td>
<td>User exit management entries</td>
</tr>
<tr>
<td>DFHAPTRD</td>
<td>DFHAPDM/DFHAPAP entries</td>
</tr>
<tr>
<td>DFHAPTRE</td>
<td>Data tables entries</td>
</tr>
<tr>
<td>DFHAPTRF</td>
<td>SAA communications and resource recovery entries</td>
</tr>
<tr>
<td>DFHAPTRG</td>
<td>ZC exception and VTAM exit entries</td>
</tr>
<tr>
<td>DFHAPTRI</td>
<td>Application domain entries (router)</td>
</tr>
<tr>
<td>DFHAPTRJ</td>
<td>ZC VTAM interface entries</td>
</tr>
<tr>
<td>DFHAPTRL</td>
<td>CICS OS/2 LU2 mirror entries</td>
</tr>
<tr>
<td>DFHAPTRN</td>
<td>Autoinstall terminal model manager entries</td>
</tr>
<tr>
<td>DFHAPTRO</td>
<td>LU6.2 application request logic entries</td>
</tr>
<tr>
<td>DFHAPTRP</td>
<td>Program control entries</td>
</tr>
<tr>
<td>DFHAPTRR</td>
<td>Partner resource manager entries</td>
</tr>
<tr>
<td>DFHAPTRS</td>
<td>DFHEISR trace entries</td>
</tr>
<tr>
<td>DFHAPTRV</td>
<td>DFHSRP trace entries</td>
</tr>
<tr>
<td>DFHAPTRW</td>
<td>Front End Programming Interface feature entries</td>
</tr>
<tr>
<td>DFHAPTR0</td>
<td>Old-style entries</td>
</tr>
<tr>
<td>DFHAPTR2</td>
<td>Statistics entries</td>
</tr>
<tr>
<td>DFHAPTR4</td>
<td>Transaction manager entries</td>
</tr>
<tr>
<td>DFHAPTR5</td>
<td>File control entries</td>
</tr>
</tbody>
</table>
### Module Function

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHAPTR6</td>
<td>DBCTL entries</td>
</tr>
<tr>
<td>DFHAPTR7</td>
<td>Transaction routing entries</td>
</tr>
<tr>
<td>DFHAPTR8</td>
<td>Security entries</td>
</tr>
<tr>
<td>DFHAPTR9</td>
<td>Interval control entries</td>
</tr>
<tr>
<td>DFHCCTRI</td>
<td>Local and global catalog domain entries</td>
</tr>
<tr>
<td>DFHDDTRI</td>
<td>Directory manager entries</td>
</tr>
<tr>
<td>DFHDMTRI</td>
<td>Domain manager domain entries</td>
</tr>
<tr>
<td>DFHDSTRI</td>
<td>Dispatcher domain entries</td>
</tr>
<tr>
<td>DFHDUTRI</td>
<td>Dump domain entries</td>
</tr>
<tr>
<td>DFHKEetri</td>
<td>Kernel domain entries</td>
</tr>
<tr>
<td>DFHLDTRI</td>
<td>Loader domain entries</td>
</tr>
<tr>
<td>DFHLGTRI</td>
<td>Log Manager domain entries</td>
</tr>
<tr>
<td>DFHL2TRI</td>
<td>Log Manager domain entries</td>
</tr>
<tr>
<td>DFHLMTRI</td>
<td>Lock manager domain entries</td>
</tr>
<tr>
<td>DFHMETRI</td>
<td>Message domain entries</td>
</tr>
<tr>
<td>DFHMNTRI</td>
<td>Monitoring domain entries</td>
</tr>
<tr>
<td>DFHNQetri</td>
<td>Enqueue domain entries</td>
</tr>
<tr>
<td>DFHPATRI</td>
<td>Parameter manager domain entries</td>
</tr>
<tr>
<td>DFHPGTRI</td>
<td>Program manager domain entries</td>
</tr>
<tr>
<td>DFHRMTRI</td>
<td>Recovery Manager domain entries</td>
</tr>
<tr>
<td>DFHSNTRI</td>
<td>Signon entries</td>
</tr>
<tr>
<td>DFHSTTRI</td>
<td>Statistics domain entries</td>
</tr>
<tr>
<td>DFHTITRI</td>
<td>Timer domain entries</td>
</tr>
<tr>
<td>DFHTRTRI</td>
<td>Trace domain entries</td>
</tr>
<tr>
<td>DFHTSITR</td>
<td>Temporary Storage domain entries</td>
</tr>
<tr>
<td>DFHUSTRI</td>
<td>User domain entries</td>
</tr>
<tr>
<td>DFHXMTRI</td>
<td>Transaction manager domain entries</td>
</tr>
<tr>
<td>DFHIXTRI</td>
<td>Security domain entries</td>
</tr>
</tbody>
</table>

### Exits

Global user exit points are not applicable to offline utilities.
Chapter 99. Transaction Failure program

The abnormal condition program has been divided into two new programs according to function.

1. **DFHTFP** which is a new program that is invoked after transaction initialization on abnormal termination.

2. **DFHACP** which is invoked by transaction manager whenever an incorrect transaction is detected.

The transaction failure program (DFHTFP) is invoked during transaction abend processing. Its purpose is to reset the status of a terminal attached to the transaction, and to send a message informing the terminal operator that the transaction has abended. It also calls the user-written (or default) program error program (DFHPEP), and writes a message to the CSMT transient data destination.

DFHTFP resolves any abnormal conditions other than those associated with a terminal, or those handled directly by the operating system.

**Design overview**

Errors can be classified as belonging in either of two broad categories:

1. **DFHTFP.** Task abnormal conditions, which are detected by CICS control programs and are often due to an application program destroying system control information. When this happens, the task is terminated, the program error program (DFHPEP) is called, the terminal operator is, if possible, informed of the error, and the error is logged at destination CSMT. If the transaction has entered syncpoint processing, then DFHPEP is **NOT** called.

2. **DFHACP.** Operator errors, such as incorrect transaction identifiers, security key violations, or failure of an operator to sign on to the system before attempting to communicate with CICS. When any of these happens, the program error program is **NOT** called, the terminal operator is notified, and the error is logged at destination CSMT.

Figure 107 on page 1258 and Figure 108 on page 1259 show the interfaces between the abnormal condition programs, DFHTFP and DFHACP, and other components when an error has been detected.
Notes:

1. DFHTFP is invoked by transaction manager whenever a task is abnormally terminated. The operator ID for error messages is in the terminal control table terminal entry (TCTTE) at TCTTEOI. DFHTFP returns to transaction manager after the error message has been issued. When a task is abnormally terminated because of a stall purge condition, the stall purge count is increased by one and the transaction identifier (from the installed resource definition) is included in the error message.

2. DFHTFP communicates with storage control to obtain and release terminal input/output areas (TIOAs).

3. DFHTFP links to the user-supplied (or default) program error program by issuing a DFHPGLU LINK_URM domain call, which passes a parameter list via a COMMAREA (mapped in this case by DFHPCOM TYPE=DSECT). Any abend within a DFHPEP program results in control returning to DFHTFP unless there is an active HANDLE ABEND for this program. See [Chapter 65. Program error program](#) on page 874 for further information about the DFHPEP program.

4. DFHTFP and DFHACP both write error messages to the transient data destination, CSMT, by calling the message domain.

Figure 107. DFHTFP abnormal condition program interfaces
DFHACP is invoked by transaction manager whenever an incorrect transaction code is detected.

2. DFHTFP and DFHACP both write error messages to the transient data destination, CSMT, by calling the message domain.

**Modules**

DFHTFP, DFHACP, DFHAPAC, and DFHAPXME

**Exits**

No global user exit points are provided for this function.

**Trace**

The following point ID is provided for the abnormal condition program:
- AP 00DC, for which the trace level is AP 1.

For more information about the trace points, see the [CICS Trace Entries]. For more information about using traces in problem determination, see the [CICS Problem Determination Guide].
Chapter 100. Transaction manager domain (XM)

The transaction manager domain (also sometimes known simply as “transaction manager”) provides transaction-related services to:

- Create tasks
- Terminates tasks
- Purge tasks
- Inquire on tasks
- Manage transaction definitions
- Manage tranclass definitions.

The transaction manager domain also provides a transaction environment to enable other CICS components to implement transaction-related services.

Transaction manager domain’s specific gates

Table 113 summarizes the transaction manager domain’s specific gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and whether or not the functions are available through the exit programming interface (XPI).

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>XPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>XMAT</td>
<td>XM 1101</td>
<td>ATTACH</td>
<td>NO</td>
</tr>
<tr>
<td>XM 1102</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XMBD</td>
<td>XM 0501</td>
<td>START_BROWSE_TRANDEF</td>
<td>NO</td>
</tr>
<tr>
<td>XM 0502</td>
<td>GET_NEXT_TRANDEF</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>XM 0502</td>
<td>END_BROWSE_TRANDEF</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>XMCL</td>
<td>XM 0A01</td>
<td>ADD_REPLACE_TCLASS</td>
<td>NO</td>
</tr>
<tr>
<td>XM 0A02</td>
<td>ADD_TCLASS</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>XM 0A02</td>
<td>INQUIRE_TCLASS</td>
<td>YES</td>
<td></td>
</tr>
<tr>
<td>XM 0A02</td>
<td>INQUIRE_ALL_TCLASSES</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>XM 0A02</td>
<td>SET_TCLASS</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>XM 0A02</td>
<td>DELETE_TCLASS</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>XM 0A02</td>
<td>START_BROWSE_TCLASS</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>XM 0A02</td>
<td>GET_NEXT_TCLASS</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>XM 0A02</td>
<td>END_BROWSE_TCLASS</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>XM 0A02</td>
<td>REGISTER_TCLASS_USAGE</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>XM 0A02</td>
<td>Deregister_TCLASS_USAGE</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>XM 0A02</td>
<td>LOCATE_AND_LOCK_TCLASS</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>XM 0A02</td>
<td>UNLOCK_TCLASS</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>XMDD</td>
<td>XM 0601</td>
<td>DELETE_TRANDEF</td>
<td>NO</td>
</tr>
<tr>
<td>XM 0602</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XMER</td>
<td>XM 1204</td>
<td>SET_DEFERRED_MESSAGE</td>
<td>NO</td>
</tr>
<tr>
<td>XM 1205</td>
<td>INQUIRE_DEFERRED_MESSAGE</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>XM 1205</td>
<td>SET_DEFERRED_ABEND</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>XM 1205</td>
<td>INQUIRE_DEFERRED_ABEND</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>XM 1205</td>
<td>REPORT_MESSAGE</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>XM 1205</td>
<td>ABEND_TRANSACTION</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>XMFD</td>
<td>XM 0701</td>
<td>FIND_PROFILE</td>
<td>NO</td>
</tr>
<tr>
<td>XM 0702</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 113. Transaction manager domain’s specific gates (continued)

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>XPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>XMIQ</td>
<td>XM 1001</td>
<td>INQUIRE_TRANSACTION</td>
<td>YES</td>
</tr>
<tr>
<td></td>
<td>XM 1002</td>
<td>SET_TRANSACTION</td>
<td>YES</td>
</tr>
<tr>
<td></td>
<td></td>
<td>START_BROWSE_TRANSACTION</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GET_NEXT_TRANSACTION</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>END_BROWSE_TRANSACTION</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>START_BROWSE_TXN_TOKEN</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GET_NEXT_TXN_TOKEN</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>END_BROWSE_TXN_TOKEN</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INQUIRE_TRANSACTION_TOKEN</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SET_TRANSACTION_TOKEN</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PURGE_TRANSACTION</td>
<td>NO</td>
</tr>
<tr>
<td>XMLD</td>
<td>XM 0401</td>
<td>LOCATE_AND_LOCK_TRANDEF</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>XM 0402</td>
<td>UNLOCK_TRANDEF</td>
<td>NO</td>
</tr>
<tr>
<td>XMSR</td>
<td>XM 0801</td>
<td>INQUIRE_MXT</td>
<td>YES</td>
</tr>
<tr>
<td></td>
<td>XM 0802</td>
<td>SET_MXT</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INQUIRE_DTRTRAN</td>
<td>YES</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SET_DTRTRAN</td>
<td>NO</td>
</tr>
<tr>
<td>XMXD</td>
<td>XM 0201</td>
<td>ADD_REPLACE_TRANDEF</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>XM 0202</td>
<td>SET_TRANDEF</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INQUIRE_TRANDEF</td>
<td>YES</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INQUIRE_REMOTE_TRANDEF</td>
<td>NO</td>
</tr>
<tr>
<td>XMXE</td>
<td>XM 1401</td>
<td>GET_TXN_ENVIRONMENT</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>XM 1402</td>
<td>FREE_TXN_ENVIRONMENT</td>
<td>NO</td>
</tr>
</tbody>
</table>

**XMAT gate, ATTACH function**

The ATTACH function of the XMAT gate is used to attach a new transaction.

**Input parameters**

**TRANSACTION_ID**

The transaction identifier to attach.

**TPNAME**

Alternative means of specifying the transaction identifier to attach.

**[ATTACH_PARMS]**

Parameters to be passed to the attached transaction.

**[PRIORITY]**

Combined user and terminal priority to be added to that of the transaction definition to determine the total priority of the attached transaction.

**[TOTAL_PRIORITY]**

The overriding priority to be associated with the attached transaction.

**FACILITY_TYPE**

The type of principal facility to be associated with the attached transaction. It can have any of these values:

- NONE
- TERMINAL
- TD
- START

**START_CODE**

Indicates the reason for the attach. It can have any of these values:

- C
- T
- TT
- QD
- S
- SD
- SZ
- DF

**[TF_TOKEN]**

Token identifying a terminal to be associated with the transaction.
Transaction manager domain (XM)

[IC_TOKEN]
Token identifying a START request to be associated with the transaction.

[TD_TOKEN]
Token identifying a TDQ to be associated with the transaction.

[US_TOKEN]
Token identifying a user to be associated with the transaction.

[SYSTEM_ATTACH]
Indicates whether the transaction should be attached as a system transaction. It can have either of these values:
YES | NO

[SUSPEND]
Indicates whether the attacher is willing to suspend during the attach. It can have either of these values:
YES | NO

RETURN_NOT_FOUND
Indicates whether the attacher wishes to receive the NOT_FOUND exception. Default is to attach CSAC in place of the requested transaction. It can have either of these values:
YES | NO

[RESTART_COUNT]
If the attach is for a restarted transaction then this count indicates the number of this restart attempt.

Output parameters

[TRANSACTION_TOKEN]
Is the token identifying the newly attached transaction.

[TRANNUM]
Is the transaction number assigned to the newly attached transaction.

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>NOT_FOUND, DISABLED, ININSUFIENT_STORAGE, NOT_ENABLED_FOR_SHUTDOWN</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

XMBD gate, START_BROWSE_TRANDEF function
The START_BROWSE_TRANDEF function of the XMBD gate is used to initiate a browse of installed transaction definitions.

Input parameters

[START_AT]
Identifies a transaction identifier that the browse is to start at.
Output parameters

BROWSE_TOKEN
Token identifying this transaction definition browse.

RESPONSE is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>LOGIC_ERROR, ABEND, LOOP</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

XMBD gate, GET_NEXT_TRANDEF function

The GET_NEXT_TRANDEF function of the XMBD gate is used to return information about the next transaction definition in the browse.

Input parameters

BROWSE_TOKEN
Token identifying this browse of the transaction definitions.

Output parameters

[TRANSACTION_ID]
Transaction identifier

[INITIAL_PROGRAM]
Initial program of transaction.

[PROFILE_NAME]
Profile of transaction.

[TWASIZE]
Size of Transaction Work Area.

[TRAN_PRIORITY]
Transaction priority

[STATUS]
The status of the transaction. It can have either of these values:
ENABLED|DISABLED

[PARTITIONSET]
The partitionset defined for the transaction. It can have any of these values:
NONE|NAMED|KEEP|OWN

[PARTITIONSET_NAME]
The name of the user defined partitionset used by the transaction.

[TASKDATAKEY]
The storage key that task-lifetime storage is allocated in. It can have either of these values:
CICS|USER

[TASKDATALOC]
The location of task-lifetime storage. It can have either of these values:
Whether task-lifetime storage is to be cleared before it is freemained. It can have either of these values:
YES|NO

Whether the transaction uses the default system runaway limit. It can have either of these values:
YES|NO

The runaway limit associated with the transaction.

Whether the transaction is defined to be dynamic. It can have either of these values:
YES|NO

Whether the transaction is eligible to queue locally when it is started on the remote system. It can have either of these values:
YES|NO

Whether the transaction is remote. It can have either of these values:
YES|NO

The system that a remote transaction is to be routed to.

The name of a remote transaction on the remote system.

Profile to be used to route a remote transaction to a remote system.

Whether the transaction belongs to a tclass. It can have either of these values:
YES|NO

The name of the tclass that the transaction belongs to.

The action to take if work performed by the transaction becomes indoubt. It can have any of these values:
BACKOUT|COMMIT|WAIT

Whether the transaction is restartable. It can have either of these values:
YES|NO

Whether the transaction is system-purgeable. It can have either of these values:
YES|NO

The deadlock timeout value for the transaction.
Transaction manager domain (XM)

[TPURGE]
  Whether the transaction can be purged after a terminal error. It can have either of these values:
  YES|NO

[DUMP]
  Whether transaction dumps are to be taken. It can have either of these values:
  YES|NO

[TRACE]
  The level of tracing associated with the transaction. It can have any of these values:
  STANDARD|SPECIAL|SUPPRESSED

[SHUTDOWN]
  Whether the transaction can be run during shutdown. It can have either of these values:
  ENABLED|DISABLED

[RESSEC]
  Whether resource security checking is active. It can have either of these values:
  YES|NO

[CMDSEC]
  Whether command security checking is active. It can have either of these values:
  YES|NO

[STORAGE_FREEZE]
  Whether storage freeze is on for the transaction. It can have either of these values:
  YES|NO

[ISOLATE]
  Whether the transaction runs in its own subspace. It can have either of these values:
  YES|NO

RESPONSE
  is the domain’s response to the call. It can have any of these values:
  OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
  is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>LOGIC_ERROR, ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>BROWSE_END_TRANDEF</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_BROWSE_TOKEN, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

XMBD gate, END_BROWSE_TRANDEF function
  The END_BROWSE_TRANDEF function of the XMBD gate is used to terminate a browse of installed transaction definitions.
Input parameters

**BROWSE_TOKEN**
Token identifying this transaction definition browse.

Output parameters

**RESPONSE**
is the domain’s response to the call. It can have any of these values:
OK|DISASTER|INVALID|KERNERROR|PURGED

**[REASON]**
is returned when RESPONSE is DISASTER or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>LOGIC_ERROR, ABEND, LOOP</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_BROWSE_TOKEN, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

**XMCL gate, ADD_REPLACE_TCLASS function**
The ADD_REPLACE_TCLASS function of the XMCL gate is used to install a tclass definition.

Input parameters

**TCLASS_NAME**
The name of the tclass.

**MAX_ACTIVE**
The max-active limit of the tclass.

**[PURGE_THRESHOLD]**
The purge-threshold limit of the tclass.

Output parameters

**[TCLASS_TOKEN]**
Token identifying the tclass.

**RESPONSE**
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|KERNERROR|PURGED

**[REASON]**
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>LOGIC_ERROR, ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>INVALID_TCLASS_NAME, INVALID_MAX_ACTIVE, INVALID_PURGE_THRESHOLD</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

**XMCL gate, ADD_TCLASS function**
The ADD_TCLASS function of the XMCL gate is used to add an internal tclass definition.
Transaction manager domain (XM)

Input parameters

[TCLASS_NAME]
The name of the tclass.

MAX_ACTIVE
The max-active limit of the tclass.

[PURGE_THRESHOLD]
The purge-threshold limit of the tclass.

Output parameters

TCLASS_TOKEN
Token identifying the tclass.

RESPONSE
is the domain’s response to the call. Possible values are:
OK|EXCEPTION|DISASTER|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. It can have any of these values:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>LOGIC_ERROR, ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>DUPLICATE_TCLASS_NAME, INVALID_TCLASS_NAME, INVALID_MAX_ACTIVE, INVALID_PURGE_THRESHOLD</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

XMCL gate, INQUIRE_TCLASS function

The INQUIRE_TCLASS function of the XMCL gate is used to inquire upon a tclass.

Input parameters

INQ_TCLASS_NAME
The name of the tclass being inquired upon.

TCLASS_TOKEN
Token identifying tclass being inquired upon.

Output parameters

[TCLASS_NAME]
The name of the tclass.

[MAX_ACTIVE]
The max-active limit of the tclass.

[PURGE_THRESHOLD]
The purge-threshold limit of the tclass.

[current_ACTIVE]
The number of active transactions in the tclass.

[current_QUEUED]
The number of queuing transactions in the tclass.

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED
[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>LOGIC_ERROR, ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>UNKNOWN_TCLASS</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_TCLASS_TOKEN, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

**XMCL gate, INQUIRE_ALL_TCLASSES function**

The INQUIRE_ALL_TCLASSES function of the XMCL gate is used to inquire about the current state of all the tclasses in the system.

**Input parameters**
None.

**Output parameters**

**[TOTAL_ACTIVE]**
The number of transactions active in a tclass.

**[TOTAL_QUEUED]**
The number of transactions queueing for a tclass.

**RESPONSE** is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>LOGIC_ERROR, ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>UNKNOWN_TCLASS</td>
</tr>
<tr>
<td>LOGIC_ERROR</td>
<td>INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

**XMCL gate, SET_TCLASS function**

The SET_TCLASS function of the XMCL gate is used to modify a tclass definition.

**Input parameters**

**TCLASS_NAME**
The name of the tclass to be changed.

**TCLASS_TOKEN**
Token identifying tclass to be changed.

**[MAX_ACTIVE]**
The max-active limit of the tclass.

**[PURGE_THRESHOLD]**
The purge-threshold limit of the tclass.
Transaction manager domain (XM)

Output parameters

RESPONSE is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID.
Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>LOGIC_ERROR, ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>UNKNOWN_TCLASS, INVALID_MAX_ACTIVE, INVALID_PURGE_THRESHOLD</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_TCLASS_TOKEN, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

XMCL gate, DELETE_TCLASS function

The DELETE_TCLASS function of the XMCL gate is used to discard an installed tclass definition.

Input parameters

TCLASS_NAME
The name of the tclass to be deleted.

Output parameters

RESPONSE is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID.
Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>LOGIC_ERROR, ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>UNKNOWN_TCLASS, TCLASS_BUSY</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

XMCL gate, START_BROWSE_TCLASS function

The START_BROWSE_TCLASS function of the XMCL gate is used to initiate a browse of installed tclass definitions.

Input parameters

[START_AT] Identifies a tclass that the browse is to start at.

Output parameters

BROWSE_TOKEN Token identifying this tclass browse.

RESPONSE is the domain’s response to the call. It can have any of these values:
OK|DISASTER|KERNERROR|PURGED
[REASON] is returned when RESPONSE is DISASTER or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>LOGIC_ERROR, ABEND, LOOP</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

**XMCL gate, GET_NEXT_TCLASS function**

The GET_NEXT_TCLASS function of the XMCL gate is used to return information about the next tclass definition in the browse.

**Input parameters**

**BROWSE_TOKEN**

Token identifying this browse of the tclass definitions.

**Output parameters**

**[TCLASS_NAME]**

The name of the tclass.

**[MAX_ACTIVE]**

The max-active limit of the tclass.

**[PURGE_THRESHOLD]**

The purge-threshold limit of the tclass.

**[CURRENT_ACTIVE]**

The number of active transactions in the tclass.

**[CURRENT_QUEUE]**

The number of queuing transactions in the tclass.

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>LOGIC_ERROR, ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>BROWSE_END_TCLASS</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_BROWSE_TOKEN, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

**XMCL gate, END_BROWSE_TCLASS function**

The END_BROWSE_TCLASS function of the XMCL gate is used to terminate a browse of installed tclass definitions.

**Input parameters**

**BROWSE_TOKEN**

Token identifying this tclass browse.
**Transaction manager domain (XM)**

**Output parameters**

RESPONSE  
is the domain’s response to the call. It can have any of these values:  
OK|DISASTER|INVALID|KERNERROR|PURGED

[REASON]  
is returned when RESPONSE is DISASTER or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>LOGIC_ERROR, ABEND, LOOP</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_BROWSE_TOKEN, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

**XMCL gate, REGISTER_TCLASS_USAGE function**

The REGISTER_TCLASS_USAGE function of the XMCL gate is used to register usage of a tclass by a transaction definition.

**Input parameters**

TCLASS_NAME  
The name of the tclass that is being used.

UNKNOWN_ACTION  
Specifies the action to perform if the tclass hasn’t been installed by the user: It can have either of these values:  
CREATE|ERROR

**Output parameters**

TCLASS_TOKEN  
Token identifying tclass.

RESPONSE  
is the domain’s response to the call. It can have any of these values:  
OK|EXCEPTION|DISASTER|KERNERROR|PURGED

[REASON]  
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>LOGIC_ERROR, ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>UNKNOWN_TCLASS</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

**XMCL gate, DEREGISTER_TCLASS_USAGE function**

The DEREGISTER_TCLASS_USAGE function of the XMCL gate is used to deregister usage of a tclass by a transaction definition.

**Input parameters**

TCLASS_TOKEN  
Token identifying tclass that is no longer being used.
Output parameters
RESPONSE
is the domain’s response to the call. It can have any of these values:
OK | DISASTER | INVALID | KERNERROR | PURGED

[REASON]
is returned when RESPONSE is DISASTER or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID</td>
<td>INVALID_TCLASS_TOKEN, NOT_IN_USE, INVALID_FUNCTION</td>
</tr>
<tr>
<td>DISASTER</td>
<td>LOGIC_ERROR, ABEND, LOOP</td>
</tr>
</tbody>
</table>

XMCL gate, LOCATE_AND_LOCK_TCLASS function
The LOCATE_AND_LOCK_TCLASS function of the XMCL gate is used to locate a named tclass and lock it against delete.

Input parameters
TCLASS_NAME
Name of tclass to be located.

Output parameters
TCLASS_TOKEN
Token identifying tclass.

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK | EXCEPTION | DISASTER | KERNERROR | PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>LOGIC_ERROR, ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>UNKNOWN_TCLASS</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

XMCL gate, UNLOCK_TCLASS function
The UNLOCK_TCLASS function of the XMCL gate is used to unlock a previously locked tclass.

Input parameters
TCLASS_TOKEN
Token identifying tclass to be unlocked.

Output parameters
RESPONSE
is the domain’s response to the call. It can have any of these values:
OK | DISASTER | INVALID | KERNERROR | PURGED
**Transaction manager domain (XM)**

[REASON] is returned when RESPONSE is DISASTER or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>LOGIC_ERROR, ABEND, LOOP</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_TCLASS_TOKEN, NOT_LOCKED, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

**XMDD gate, DELETE_TRANDEF function**

The DELETE_TRANDEF function of the XMDD gate is used to discard an installed transaction definition.

**Input parameters**

**TRANSACTION_ID**

The name of the transaction to be deleted.

**Output parameters**

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK, EXCEPTION, DISASTER, KERNERROR, PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>LOGIC_ERROR, ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>UNKNOWN_TRANSACTION_ID, ICE_PENDING, AID_PENDING, SIT_PARAMETER</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

**XMER gate, SET_DEFERRED_MESSAGE function**

The SET_DEFERRED_MESSAGE function of the XMER gate is used to store a message to be issued if the attach of a transaction fails.

**Input parameters**

**MESSAGE**

The message that is to be issued.

[TRANSACTION_TOKEN]

Optional token to identify the transaction that the message is to be sent to. Defaults to the current transaction.

**Output parameters**

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK, EXCEPTION, DISASTER, INVALID, KERNERROR, PURGED

[REASON] is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:
The INQUIRE_DEFERRED_MESSAGE function of the XMER gate is used to retrieve the message that is to be issued which will indicate the cause of a transaction attach failure.

### Output parameters

**MESSAGE**

The message that is to be issued.

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

- OK
- DISASTER
- INVALID
- KERNERROR
- PURGED

**[REASON]**

is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>MESSAGE_ALREADY_SET, DEFERRED_ABEND_ALREADY_SET, INVALID_TRANSACTION_TOKEN</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

The SET_DEFERRED_ABEND function of the XMER gate is used to schedule an abend to be issued if the attach of a transaction fails.

### Input parameters

**DEFERRED_ABEND_CODE**

The abend code that is to be used.

**[TRANSACTION_DUMP]**

Indicates whether a transaction dump is to be taken for the abend. It can have either of these values:

- YES
- NO

**[TRANSACTION_TOKEN]**

Optional token to identify the transaction that is to be abended. Defaults to the current transaction.

### Output parameters

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

- OK
- EXCEPTION
- DISASTER
- INVALID
- KERNERROR
- PURGED

**[REASON]**

is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID.
Transaction manager domain (XM)

Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>MESSAGE_ALREADY_SET, DEFERRED_ABEND_ALREADY_SET, INVALID_TRANSACTION_TOKEN</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_ABEND_CODE, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

XMER gate, INQUIRE_DEFERRED_ABEND function

The INQUIRE_DEFERRED_ABEND function of the XMER gate is used to retrieve the abend that is to be issued for the transaction whose attach has failed.

Output parameters

DEFERRED_ABEND_CODE
The abend code.

[TRANSACTION_DUMP]
Indicates whether a transaction dump is to be taken for the abend. It can have either of these values:
YES|NO

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>DEFERRED_ABEND_NOT_FOUND</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

XMER gate, REPORT_MESSAGE function

The REPORT_MESSAGE function of the XMER gate is used send a deferred message if the attach of a transaction has failed.

Input parameters

MESSAGE
The message that is to be sent.

Output parameters

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:
**XMER gate, ABEND_TRANSACTION function**

The ABEND_TRANSACTION function of the XMER gate is used to abend a transaction whose attachment has failed.

**Output parameters**

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK | DISASTER | INVALID | KERNERROR | PURGED

**[REASON]**

is returned when RESPONSE is DISASTER or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

**XMFD gate, FIND_PROFILE function**

The FIND_PROFILE function of the XMFD gate is used to check whether the given profile is in use by a transaction definition.

**Input parameters**

**PROFILE_NAME**

The profile that is to be found.

**Output parameters**

**[TRANSACTION_ID]**

The name of a transaction definition that is using the profile.

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

**[REASON]**

is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>LOGIC_ERROR, ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>PROFILE_NOT_FOUND,</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

**XMIQ gate, INQUIRE_TRANSACTION function**

The INQUIRE_TRANSACTION function of the XMIQ gate is used to inquire upon a particular transaction.
Transaction manager domain (XM)

Input parameters

[TRANSACTION_NUMBER]
The number of the transaction being inquired upon.

[TRANSACTION_TOKEN]
Or the token representing the transaction being inquired upon.
If neither TRANSACTION_NUMBER or TRANSACTION_TOKEN are specified the current transaction is assumed.

[ATTACH_PARMS]
Specified if the parameter area passed on the transaction. attach are to be returned.

Output parameters

[ABEND_CODE]
The abend code if the transaction is terminating abnormally.

[ABEND_IN_PROGRESS]
Indicates whether the transaction is in the process of terminating abnormally. It can have either of these values:
YES|NO

[CICS_UOW_ID]
The CICS Unit Of Work Identifier associated with the transaction.

[CMDSEC]
Whether command security checking is active. It can have either of these values:
YES|NO

[DTIMEOUT]
The deadlock timeout value for the transaction.

[DUMP]
Whether transaction dumps are to be taken for the transaction. It can have either of these values:
YES|NO

[DYNAMIC]
Whether the transaction is dynamic. It can have either of these values:
YES|NO

[FACILITY_NAME]
The name of the principal facility associated with the transaction.

[FACILITY_TYPE]
The type of the principal facility associated with the transaction. It can have either of these values:
NONE|TERMINAL|TD|START

[INDOUBT]
The action to take if work performed by the transaction becomes indoubt. It can have any of these values:
BACKOUT|COMMIT|WAIT

[INITIAL_PROGRAM]
The initial program to linked to when the transaction started.
[ISOLATE]
Whether the transaction runs in its own subspace. It can have either of
these values:
YES|NO

[LOCAL_QUEUING]
Whether the transaction is eligible to queue locally if it is started on the
remote system. It can have either of these values:
YES|NO

[NETNAME]
The network name of a terminal principal facility.

[ORIGINAL_TRANSACTION_ID]
The transid that was used to attach the transaction.

[OUT_TRANSACTION_TOKEN]
The token that represents this transaction.

[PROFILE_NAME]
The profile of the transaction.

[REMOTE]
Whether the transaction is remote. It can have either of these values:
YES|NO

[REMOTE_NAME]
The name of a remote transaction on the remote system.

[REMOTE_SYSTEM]
The system that a remote transaction is to be routed to.

[RESOURCE_NAME]
The name of a resource that a suspended transaction is waiting for.

[RESOURCE_TYPE]
The type of resource that a suspended transaction is waiting for.

[RESSEC]
Whether resource security checking is active for the transaction. It can have
either of these values:
YES|NO

[RESTART]
Whether the transaction is restartable. It can have either of these values:
YES|NO

[RESTART_COUNT]
Contains the number of times this transaction instance has been restarted.

[RUNAWAY_LIMIT]
The runaway limit associated with the transaction.

[SPURGE]
Whether the transaction is system-purgeable. It can have either of these
values:
YES|NO

START_CODE
Indicates the reason for the attach of the transaction. It can have any of
these values:
C|T|TT|QD|S|SD|SZ|DF
Transaction manager domain (XM)

[STATUS]
The status of the transaction. It can have either of these values:
ENABLED|DISABLED

[STORAGE_CLEAR]
Whether task-lifetime storage will be cleared before it is freemained. It can have either of these values:
YES|NO

[SUSPEND_TIME]
Contains the length of time that the transaction has currently been suspended for.

[SYSTEM_TRANSACTION]
Whether the transaction has been attached by CICS. It can have either of these values:
YES|NO

[TASK_PRIORITY]
The combined priority of the transaction.

[TASKDATASIZE]
The storage key that task-lifetime storage is allocated in. It can have either of these values:
CICS|USER

[TASKDATALOC]
The location of task-lifetime storage. It can have either of these values:
BELOW|ANY

[TCLASS]
Whether the transaction belongs to a tclass. It can have either of these values:
YES|NO

[TCLASS_NAME]
The name of the tclass that the transaction belongs to.

[TPURGE]
Whether the transaction can be purged after a terminal error. It can have either of these values:
YES|NO

[TRACE]
The level of tracing associated with the transaction. It can have any of these values:
STANDARD|SPECIAL|SUPPRESSED

[TRAN_PRIORITY]
The priority of the transaction definition used to attach the transaction.

[TRAN_ROUTING_PROFILE]
Profile used to route the transaction to a remote system.

[TRANNUM]
The transaction number of the transaction.

[TRANSACTION_ID]
The transaction identifier associated with the transaction.

[TWASIZE]
Size of Transaction Work Area associated with the transaction.
The userid of the user associated with the transaction.

**RESPONSE**
is the domain’s response to the call. It can have any of these values:
OK| EXCEPTION| DISASTER| INVALID| KERNERROR| PURGED

**[REASON]**
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>NO_TRANSACTION_ENVIRONMENT, BUFFER_TOO_SMALL,</td>
</tr>
<tr>
<td></td>
<td>UNKNOWN_TRANSACTION_NUMBER, INVALID_TRANSACTION_TOKEN</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

**XMIQ gate, SET_TRANSACTION function**
The SET_TRANSACTION function of the XMIQ gate is used to change some attributes associated with a particular transaction.

**Input parameters**

**[TRANSACTION_NUMBER]**
The number of the transaction being inquired upon.

**[TRANSACTION_TOKEN]**
Or the token representing the transaction being inquired upon.

If neither TRANSACTION_NUMBER or TRANSACTION_TOKEN are specified the current transaction is assumed.

**[ABEND_CODE]**
The abend code if the transaction is terminating abnormally.

**[ABEND_IN_PROGRESS]**
Whether the transaction is in the process of terminating abnormally. It can have either of these values:
YES| NO

**[FACILITY_TYPE]**
The type of the principal facility associated with the transaction. It can have either of these values:
NONE| TERMINAL| TD| START

**START_CODE**
The reason for the attach of the transaction. It can have any of these values:
C| T| TT| QB| S| SD| SZ| DF

**[STORAGE_VIOLATIONS]**
Set to indicate that the transaction has suffered a storage violation.

**[TASK_PRIORITY]**
The combined priority of the transaction.

**[TCLASS_NAME]**
The name of the tclass that the transaction belongs to.
Reserved name DFHTCL00 is used to change a transaction so that it no longer belongs to a tclass.

**Output parameters**

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**[REASON]**

is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>NO_TRANSACTION_ENVIRONMENT, UNKNOWN_TCLASS,</td>
</tr>
<tr>
<td></td>
<td>UNKNOWN_TRANSACTION_NUMBER, INVALID_TRANSACTION_TOKEN</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

**XMIQ gate, START_BROWSE_TRANSACTION function**

The START_BROWSE_TRANSACTION function of the XMIQ gate is used to initiate a browse of all transactions in the system.

**Output parameters**

**BROWSE_TOKEN**

Token identifying this transaction browse.

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**[REASON]**

is returned when RESPONSE is INVALID or DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

**XMIQ gate, GET_NEXT_TRANSACTION function**

The GET_NEXT_TRANSACTION function of the XMIQ gate is used to inquire upon the next transaction in a transaction browse.

**Input parameters**

**BROWSE_TOKEN**

The token identifying this transaction browse.

**[ATTACH_PARMS]**

Specified if the parameter area passed on the transaction. attach is to be returned.

**Output parameters**

**[ABEND_CODE]**

The abend code if the transaction is terminating abnormally.
[ABEND_IN_PROGRESS]
Indicates whether the transaction is in the process of terminating abnormally. It can have either of these values:
YES|NO

[CICS_UOW_ID]
The CICS Unit Of Work Identifier associated with the transaction.

[CMDSEC]
Whether command security checking is active. It can have either of these values:
YES|NO

[DTIMEOUT]
The deadlock timeout value for the transaction.

[DUMP]
Whether transaction dumps are to be taken for the transaction. It can have either of these values:
YES|NO

[DYNAMIC]
Whether the transaction is dynamic. It can have either of these values:
YES|NO

[FACILITY_NAME]
The name of the principal facility associated with the transaction.

[FACILITY_TYPE]
The type of the principal facility associated with the transaction. It can have either of these values:
NONE|TERMINAL|TD|START

[INDOUBT]
The action to take if work performed by the transaction becomes indoubt. It can have any of these values:
BACKOUT|COMMIT|WAIT

[INITIAL_PROGRAM]
The initial program to linked to when the transaction started.

[ISOLATE]
Whether the transaction runs in its own subspace. It can have either of these values:
YES|NO

[LOCAL_QUEUING]
Whether the transaction is eligible to queue locally if it is started on the remote system. It can have either of these values:
YES|NO

[NETNAME]
The network name of a terminal principal facility.

[ORIGINAL_TRANSACTION_ID]
The transid that was used to attach the transaction.

[OUT_TRANSACTION_TOKEN]
The token that represents this transaction.

[PROFILE_NAME]
The profile of the transaction.
Transaction manager domain (XM)

[REMOTE]
Whether the transaction is remote. It can have either of these values:
YES|NO

[REMOTE_NAME]
The name of a remote transaction on the remote system.

[REMOTE_SYSTEM]
The system that a remote transaction is to be routed to.

[RESOURCE_NAME]
The name of a resource that a suspended transaction is waiting for.

[RESOURCE_TYPE]
The type of resource that a suspended transaction is waiting for.

[RESSEC]
Whether resource security checking is active for the transaction. It can have either of these values:
YES|NO

[RESTART]
Whether the transaction is restartable. It can have either of these values:
YES|NO

[RESTART_COUNT]
Contains the number of times this transaction instance has been restarted.

[RUNAWAY_LIMIT]
The runaway limit associated with the transaction.

[SPURGE]
Whether the transaction is system-purgeable. It can have either of these values:
YES|NO

START_CODE
Indicates the reason for the attach of the transaction. It can have any of these values:
C|T|TT|QD|S|SD|SZ|DF

[STATUS]
The status of the transaction. It can have either of these values:
ENABLED|DISABLED

[STORAGE_CLEAR]
Whether task-lifetime storage will be cleared before it is freemained. It can have either of these values:
YES|NO

[SUSPEND_TIME]
Contains the length of time that the transaction has currently been suspended for.

[SYSTEM_TRANSACTION]
Whether the transaction has been attached by CICS. It can have either of these values:
YES|NO

[TASK_PRIORITY]
The combined priority of the transaction.
The storage key that task-lifetime storage is allocated in. It can have either of these values:
CICS USER

The location of task-lifetime storage. It can have either of these values:
BELOW ANY

Whether the transaction belongs to a tclass. It can have either of these values:
YES NO

The name of the tclass that the transaction belongs to.

Whether the transaction can be purged after a terminal error. It can have either of these values:
YES NO

The level of tracing associated with the transaction. It can have any of these values:
STANDARD SPECIAL SUPPRESSED

The priority of the transaction definition used to attach the transaction.

Profile used to route the transaction to a remote system.

The transaction number of the transaction.

The transaction identifier associated with the transaction.

Size of Transaction Work Area associated with the transaction.

The userid of the user associated with the transaction.

is the domain’s response to the call. It can have any of these values:
OK EXCEPTION DISASTER INVALID KERNERROR PURGED

is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>BROWSE_END</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_BROWSE_TOKEN, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>
Transaction manager domain (XM)

**XMIQ gate, END_BROWSE_TRANSACTION function**
The END_BROWSE_TRANSACTION function of the XMIQ gate is used to terminate a browse of all transactions in the system.

**Input parameters**
BROWSE_TOKEN
Token identifying the transaction browse to be terminated.

**Output parameters**
RESPONSE
is the domain's response to the call. It can have any of these values:
OK|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_BROWSE_TOKEN, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

**XMIQ gate, START_BROWSE_TXN_TOKEN function**
The START_BROWSE_TXN_TOKEN function of the XMIQ gate is used to initiate a browse of a particular components transaction token in all transactions in the system.

**Input parameters**
TOKEN_OWNER
Identifies the particular transaction token that is to be browsed in the transactions. It can have any of these values:
AP|SM|TD|MN|PG|IC|XS|US|RF

**Output parameters**
BROWSE_TOKEN
Token identifying this transaction token browse.

RESPONSE
is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

**XMIQ gate, GET_NEXT_TXN_TOKEN function**
The GET_NEXT_TXN_TOKEN function of the XMIQ gate is used to return the transaction token associated with the next transaction in the system.
Input parameters

BROWSE_TOKEN
Identifies this browse of the transaction tokens.

Output parameters

OWNERS_TOKEN
The transaction token associated with the current transaction.

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>BROWSE_END</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_BROWSE_TOKEN, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

XMIQ gate, END_BROWSE_TXN_TOKEN function

The END_BROWSE_TXN_TOKEN function of the XMIQ gate is used to terminate a browse of transaction tokens.

Input parameters

BROWSE_TOKEN
Token identifying the transaction token browse to be terminated.

Output parameters

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_BROWSE_TOKEN, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

XMIQ gate, INQUIRE_TRANSACTION_TOKEN function

The INQUIRE_TRANSACTION_TOKEN function of the XMIQ gate is used to return a particular transaction token associated with a particular transaction.

Input parameters

[TRANSACTION_TOKEN]
Token identifying the transaction being inquired upon.

If omitted defaults to the current transaction.
Transaction manager domain (XM)

TOKEN_OWNER
Identifies the particular transaction token that is to be returned. It can have any of these values:
AP|SM|TD|MN|PG|IC|XS|US|RM|TF

Output parameters
OWNERS_TOKEN
The transaction token associated with the transaction.

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NO_TRANSACTION_ENVIRONMENT</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_BROWSE_TOKEN, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

XMIQ gate, SET_TRANSACTION_TOKEN function
The SET_TRANSACTION_TOKEN function of the XMIQ gate is used to modify a particular transaction token associated with a particular transaction.

Input parameters

[TRANSACTION_TOKEN]
Token identifying the transaction in which the token is to be modified.
If omitted defaults to the current transaction.

TOKEN_OWNER
Identifies the particular transaction token that is to be changed. It can have any of these values:
AP|SM|TD|MN|PG|IC|XS|US|RM|TF

OWNERS_TOKEN
The new value for the transaction token.

Output parameters

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NO_TRANSACTION_ENVIRONMENT</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_BROWSE_TOKEN, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>
XMIQ gate, PURGE_TRANSACTION function

The PURGE_TRANSACTION function of the XMIQ gate is used to purge a particular transaction in the system.

**Input parameters**

**TRANSACTION_NUMBER**

The number of the transaction to be purged.

**TRANSACTION_TOKEN**

Or the token representing the transaction to be purged.

**PURGE_TYPE**

The type of purge that is to be attempted. It can have either of these values:

- NORMAL
- FORCE

**Output parameters**

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

- OK
- EXCEPTION
- DISASTER
- INVALID
- KERNERROR
- PURGED

**[REASON]**

is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID.

Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>UNKNOWN_TRANSACTION_NUMBER, INVALID_TRANSACTION_TOKEN, PURGE_DEFERRED, TRANSACTIION_INITIALIZER, TRANSACTION_TERMINATING, PURGE_SYSTEM_TRANSACTION, PURGE_ABENDING_TRANSACTION, SPURGE_PROTECTED, PURGE_INHIBITED, INVALID_STATE</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

XMLD gate, LOCATE_AND_LOCK_TRANDEF function

The LOCATE_AND_LOCK_TRANDEF function of the XMLD gate is used to locate a particular transaction definition instance.

**Input parameters**

**TRANSACTION_ID**

Transaction identifier to locate.

**TPNAME**

Or alternatively a tpname alias of the transaction definition to locate.

**[USE_DTRTRAN]**

If the named transaction-id or tpname cannot be found then indicates whether the DTRTRAN, if installed, should be used instead. It can have either of these values:

- YES
- NO

**Output parameters**

**TRANDEF_TOKEN**

The token representing the returned transaction definition.
The primary transaction identifier of the returned transaction.

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

**[REASON]**

is returned when RESPONSE is DISASTER, EXCEPTION or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NOT_FOUND</td>
</tr>
<tr>
<td>DISASTER</td>
<td>LOGIC_ERROR, ABEND, LOOP</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_TPNAME, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

**XMLD gate, UNLOCK_TRANDEF function**

The UNLOCK_TRANDEF function of the XMLD gate is used to unlock a previously located transaction definition instance.

**Input parameters**

**TRANDEF_TOKEN**

Transaction definition instance to unlock.

**Output parameters**

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK|DISASTER|INVALID|KERNERROR|PURGED

**[REASON]**

is returned when RESPONSE is DISASTER or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>LOGIC_ERROR, ABEND, LOOP</td>
</tr>
<tr>
<td>INVALID</td>
<td>NOT_LOCKED, INVALID_TOKEN, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

**XMSR gate, INQUIRE_MXT function**

The INQUIRE_MXT function of the XMSR gate is used to inquire upon the state of MXT in the system.

**Output parameters**

**[MXT_QUEUED]**

The number of user transactions queued for MXT.

**[TCLASS_QUEUED]**

The number of transactions queued for tclass membership.

**[CURRENT_ACTIVE]**

The number of active user transactions.

**[CURRENT_ACTIVE]**

The number of user transactions queued on MXT.
**Transaction manager domain (XM)**

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK | DISASTER | INVALID | KERNERROR | PURGED

**[REASON]**

is returned when RESPONSE is DISASTER or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>LOGIC_ERROR, ABEND, LOOP</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

**XMSR gate, SET_MXT function**

The SET_MXT function of the XMSR gate is used to change MXT in the system.

**Input parameters**

**MXT_LIMIT**

The requested setting for MXT.

**Output parameters**

**MXT_LIMIT_SET**

The MXT limit that could be set.

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK | DISASTER | INVALID | KERNERROR | PURGED

**[REASON]**

is returned when RESPONSE is DISASTER, EXCEPTION or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>INVALID_MXT_LIMIT, LIMIT_TOO_HIGH</td>
</tr>
<tr>
<td>DISASTER</td>
<td>LOGIC_ERROR, ABEND, LOOP</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

**XMSR gate, INQUIRE_DTRTRAN function**

The INQUIRE_DTRTRAN function of the XMSR gate returns the name of the dynamic transaction routing transaction.

**Output parameters**

**DTRTRAN**

The name of the dynamic transaction routing transaction definition.

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK | DISASTER | INVALID | KERNERROR | PURGED

**[REASON]**

is returned when RESPONSE is DISASTER or INVALID. Possible values are:
### XMSR gate, SET_DTRTRAN function

The SET_DTRTRAN function of the XMSR gate changes the dynamic transaction routing transaction definition.

**Input parameters**

**DTRTRAN**
The name of the dynamic transaction routing transaction definition.

**Output parameters**

**RESPONSE**
is the domain’s response to the call. It can have any of these values:
OK|DISASTER|INVALID|KERNERROR|PURGED

**[REASON]**
is returned when RESPONSE is DISASTER or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>LOGIC_ERROR, ABEND, LOOP</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

### XMXD gate, ADD_REPLACE_TRANDEF function

The ADD_REPLACE_TRANDEF function of the XMXD gate is used to install a transaction definition.

**Input parameters**

**TRANSACTION_ID**
Name of transaction definition to install.

**PROFILE_NAME**
Profile of transaction.

**TRAN_PRIORITY**
Transaction priority

**[INITIAL_PROGRAM]**
Initial program of transaction.

**[TWASIZE]**
Size of Transaction Work Area.

**[STATUS]**
The status of the transaction. It can have either of these values:
ENABLED|DISABLED

**[PARTITIONSET]**
The partitionset defined for the transaction. It can have any of these values:
NONE|NAMED|KEEP|OWN

**[PARTITIONSET_NAME]**
The name of the user defined partitionset used by the transaction.
[TASKDATAKEY]
The storage key that task-lifetime storage is allocated in. It can have either of these values:
CICS|USER

[TASKDATALOC]
The location of task-lifetime storage. It can have either of these values:
BELOW|ANY

[STORAGE_CLEAR]
Whether task-lifetime storage is to be cleared before it is freemained. It can have either of these values:
YES|NO

[SYSTEM_RUNAWAY]
Whether the transaction uses the default system runaway limit. It can have either of these values:
YES|NO

[RUNAWAY_LIMIT]
The runaway limit associated with the transaction.

[DYNAMIC]
Whether the transaction is defined to be dynamic. It can have either of these values:
YES|NO

[LOCAL_QUEUING]
Whether the transaction is eligible to queue locally when it is started on the remote system. It can have either of these values:
YES|NO

[REMOTE_SYSTEM]
The system that a remote transaction is to be routed to.

[REMOTE_NAME]
The name of a remote transaction on the remote system.

[TRAN_ROUTING_PROFILE]
Profile to be used to route a remote transaction to a remote system.

[TCLASS]
Whether the transaction belongs to a tclass. It can have either of these values:
YES|NO

[TCLASS_NAME]
The name of the tclass that the transaction belongs to.

[INDOUBT]
The action to take if work performed by the transaction becomes indoubt. It can have any of these values:
BACKOUT|COMMIT|WAIT

[RESTART]
Whether the transaction is restartable. It can have either of these values:
YES|NO

[SPURGE]
Whether the transaction is system-purgeable. It can have either of these values:
Transaction manager domain (XM)

YES|NO

[DTIMEOUT]
The deadlock timeout value for the transaction.

[TPURGE]
Whether the transaction can be purged after a terminal error. It can have either of these values:
YES|NO

[DUMP]
Whether transaction dumps are to be taken. It can have either of these values:
YES|NO

[TRACE]
The level of tracing associated with the transaction. It can have any of these values:
STANDARD|SPECIAL|SUPPRESSED

[SHUTDOWN]
Whether the transaction can be run during shutdown. It can have either of these values:
ENABLED|DISABLED

[RESSEC]
Whether resource security checking is active. It can have either of these values:
YES|NO

[CMDSEC]
Whether command security checking is active. It can have either of these values:
YES|NO

[STORAGE_FREEZE]
Whether storage freeze is on for the transaction. It can have either of these values:
YES|NO

[ISOLATE]
Whether the transaction runs in its own subspace. It can have either of these values:
YES|NO

[CATALOGUED_EXTERNALS]
Block of data specified as an alternative to the above parameters when a transaction definition is being installed from the catalog.

[ALIAS]
Alternative name for transaction definition.

[TASKREQ]
Alternative name for transaction definition so that it can be invoked by PF/PA key, light pen, etc.

[XTRANID]
Alternative name for transaction definition originally specified in hexadecimal notation.
[TPNAME]
Alternative name of transaction definition in form of a sixty four character transaction program name.

[System_Definition]
Whether the definition is being added on behalf of CICS or not. It can have either of these values:
YES | NO

Output parameters

[TRANDEF_TOKEN]
Token returned to represent the installed transaction definition.

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION or INVALID.
Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>TWASIZE_INVALID, RUNAWAY_LIMIT_INVALID, TRANSACTION_ID_INVALID, ALIAS_INVALID, XTRANID_INVALID, TASKREQ_INVALID, TPNAME_INVALID, RECOVERY_NOT_COMPLETE</td>
</tr>
<tr>
<td>DISASTER</td>
<td>LOGIC_ERROR, ABEND, LOOP</td>
</tr>
<tr>
<td>INVALID</td>
<td>INITIAL_PROGRAM_EXPECTED, REMOTE_SYSTEM_EXPECTED, REMOTE_NAME_EXPECTED, RUNAWAY_LIMIT_EXPECTED, TRAN_ROUTING_PROF_EXPECTED, TCLASS_NAME_EXPECTED, PARTITIONSET_NAME_EXPECTED, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

XMXD gate, SET_TRANDEF function

The SET_TRANDEF function of the XMXD gate is used to modify transaction definition creating a new transaction definition instance.

Input parameters

TRANSACTION_ID
Name of transaction definition to change.

[TRAN_PRIORITY]
Transaction priority.

[STATUS]
The status of the transaction. It can have either of these values:
ENABLED | DISABLED

[System_Runaway]
Whether the transaction uses the default system runaway limit. It can have either of these values:
YES | NO

[Runaway_Limit]
The runaway limit associated with the transaction.

[TCLASS]
Whether the transaction belongs to a tclass. It can have either of these values:
YES | NO
Transaction manager domain (XM)

[TCLASS_NAME]
The name of the tclass that the transaction belongs to.

[SPURGE]
Whether the transaction is system-purgeable. It can have either of these values:
YES|NO

[DUMP]
Whether transaction dumps are to be taken. It can have either of these values:
YES|NO

[TRACE]
The level of tracing associated with the transaction. It can have any of these values:
STANDARD|SPECIAL|SUPPRESSED

[SHUTDOWN]
Whether the transaction can be run during shutdown. It can have either of these values:
ENABLED|DISABLED

[STORAGE_FREEZE]
Whether storage freeze is on for the transaction. It can have either of these values:
YES|NO

[SHUTDOWN_DISABLEOVERRIDE]
Whether to override a SHUTDOWN setting of DISABLED for the transaction definition. It can have either of these values:
YES|NO

Output parameters

[TRANDEF_TOKEN]
Token returned to represent the new transaction definition instance.

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>UNKNOWN_TRANSACTION_ID, RUNAWAY_LIMIT_INVALID, UNKNOWN_TCLASS</td>
</tr>
<tr>
<td>DISASTER</td>
<td>LOGIC_ERROR, ABEND, LOOP</td>
</tr>
<tr>
<td>INVALID</td>
<td>RUNAWAY_LIMITEXPECTED, TCLASS_NAME_EXPECTED, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

XMXD gate, INQUIRE_TRANDEF function
The INQUIRE_TRANDEF function of the XMXD gate is used to inquire upon a named transaction definition.
Input parameters

INQ_TRANSACTION_ID
Transaction-id to inquire upon.

TRANDEF_TOKEN
Or alternatively token representing transaction definition to inquire upon.

[USE_DTRTRAN]
If the INQ_TRANSACTION_ID cannot be found then indicates whether the DTRTRAN, if installed, should be used for the inquire instead. It can have either of these values:
YES|NO

Output parameters

[TRANSACTION_ID]
Transaction identifier.

[INITIAL_PROGRAM]
Initial program of transaction.

[PROFILE_NAME]
Profile of transaction.

[TWASIZE]
Size of Transaction Work Area.

[TRAN_PRIORITY]
Transaction priority.

[STATUS]
The status of the transaction. It can have either of these values:
ENABLED|DISABLED

[PARTITIONSET]
The partitionset defined for the transaction. It can have any of these values:
NONE|NAMED|KEEP|OWN

[PARTITIONSET_NAME]
The name of the user defined partitionset used by the transaction.

[TASKDATAKEY]
The storage key that task-lifetime storage is allocated in. It can have either of these values:
CICS|USER

[TASKDATALOC]
The location of task-lifetime storage. It can have either of these values:
BELOW|ANY

[STORAGE_CLEAR]
Whether task-lifetime storage is to be cleared before it is freemained. It can have either of these values:
YES|NO

[System_RUNAWAY]
Whether the transaction uses the default system runaway limit. It can have either of these values:
YES|NO

[RUNAWAY_LIMIT]
The runaway limit associated with the transaction.
Transaction manager domain (XM)

[DYNAMIC]
Whether the transaction is defined to be dynamic. It can have either of these values:
YES|NO

[LOCAL_QUEUING]
Whether the transaction is eligible to queue locally when it is started on the remote system. It can have either of these values:
YES|NO

[REMOTE]
Whether the transaction is remote. It can have either of these values:
YES|NO

[REMOTE_SYSTEM]
The system that a remote transaction is to be routed to.

[REMOTE_NAME]
The name of a remote transaction on the remote system.

[TRAN_ROUTING_PROFILE]
Profile to be used to route a remote transaction to a remote system.

[TCLASS]
Whether the transaction belongs to a tclass. It can have either of these values:
YES|NO

[TCLASS_NAME]
The name of the tclass that the transaction belongs to.

[INDOUBT]
The action to take if work performed by the transaction becomes indoubt. It can have any of these values:
BACKOUT|COMMIT|WAIT

[RESTART]
Whether the transaction is restartable. It can have either of these values:
YES|NO

[SPURGE]
Whether the transaction is system-purgeable. It can have either of these values:
YES|NO

[DTIMEOUT]
The deadlock timeout value for the transaction.

[TPURGE]
Whether the transaction can be purged after a terminal error. It can have either of these values:
YES|NO

[DUMP]
Whether transaction dumps are to be taken. It can have either of these values:
YES|NO

[TRACE]
The level of tracing associated with the transaction. It can have any of these values:
Whether the transaction can be run during shutdown. It can have either of these values:
ENABLED | DISABLED

Whether resource security checking is active. It can have either of these values:
YES | NO

Whether command security checking is active. It can have either of these values:
YES | NO

Whether storage freeze is on for the transaction. It can have either of these values:
YES | NO

Whether the transaction runs in its own subspace. It can have either of these values:
YES | NO

Whether a system task will be attached using this transaction definition. It can have either of these values:
YES | NO

Indicates whether the returned transaction definition is the dynamic transaction routing transaction definition or not. It can have either of these values:
YES | NO

returns historical data indicating the frequency of usage of each subspace-inheriting open TCB mode by tasks with the caller’s transaction id.

is the domain’s response to the call. It can have any of these values:
OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>UNKNOWN_TRANSACTION_ID</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_TOKEN, INVALID_FUNCTION</td>
</tr>
<tr>
<td>DISASTER</td>
<td>LOGIC_ERROR, ABEND, LOOP</td>
</tr>
</tbody>
</table>
Transaction manager domain (XM)

**XMXD gate, INQUIRE_REMOTE_TRANDEF function**

The INQUIRE_REMOTE_TRANDEF function of the XMXD gate is used to inquire upon a remote transaction definition.

**Input parameters**

**REMTESYSTEM_KEY**
Remote system of remote transaction definition to be found.

**REMTENENAME_KEY**
Remote name of remote transaction definition to be found.

**Output parameters**

**[TRANSACTION_ID]**
Transaction identifier.

**[INITIAL_PROGRAM]**
Initial program of transaction.

**[PROFILE_NAME]**
Profile of transaction.

**[TWASIZE]**
Size of Transaction Work Area.

**[TRAN_PRIORITY]**
Transaction priority.

**[STATUS]**
The status of the transaction. It can have either of these values:

- ENABLED
- DISABLED

**[PARTITIONSET]**
The partitionset defined for the transaction. It can have any of these values:

- NONE
- NAMED
- KEEP
- OWN

**[PARTITIONSET_NAME]**
The name of the user defined partitionset used by the transaction.

**[TASKDATAKEY]**
The storage key that task-lifetime storage is allocated in. It can have either of these values:

- CICS
- USER

**[TASKDATALOC]**
The location of task-lifetime storage. It can have either of these values:

- BELOW
- ANY

**[STORAGE_CLEAR]**
Whether task-lifetime storage is to be cleared before it is freemained. It can have either of these values:

- YES
- NO

**[SYSTEM_RUNAWAY]**
Whether the transaction uses the default system runaway limit. It can have either of these values:

- YES
- NO

**[RUNAWAY_LIMIT]**
The runaway limit associated with the transaction.
Whether the transaction is defined to be dynamic. It can have either of these values:
YES | NO

Whether the transaction is eligible to queue locally when it is started on the remote system. It can have either of these values:
YES | NO

Whether the transaction is remote. It can have either of these values:
YES | NO

The system that a remote transaction is to be routed to.

The name of a remote transaction on the remote system.

Profile to be used to route a remote transaction to a remote system.

Whether the transaction belongs to a tclass. It can have either of these values:
YES | NO

The name of the tclass that the transaction belongs to.

The action to take if work performed by the transaction becomes indoubt. It can have any of these values:
BACKOUT | COMMIT | WAIT

Whether the transaction is restartable. It can have either of these values:
YES | NO

Whether the transaction is system-purgeable. It can have either of these values:
YES | NO

The deadlock timeout value for the transaction.

Whether the transaction can be purged after a terminal error. It can have either of these values:
YES | NO

Whether transaction dumps are to be taken. It can have either of these values:
YES | NO

The level of tracing associated with the transaction. It can have any of these values:
Transaction manager domain (XM)

STANDARD|SPECIAL|SUPPRESSED

[SHUTDOWN]
Whether the transaction can be run during shutdown. It can have either of these values:
ENABLED|DISABLED

[RESSEC]
Whether resource security checking is active. It can have either of these values:
YES|NO

[CMDSEC]
Whether command security checking is active. It can have either of these values:
YES|NO

[STORAGE_FREEZE]
Whether storage freeze is on for the transaction. It can have either of these values:
YES|NO

[ISOLATE]
Whether the transaction runs in its own subspace. It can have either of these values:
YES|NO

[SYSTEM_ATTACH]
Whether a system task will be attached using this transaction definition. It can have either of these values:
YES|NO

[DTRTRAN]
Indicates whether the returned transaction definition is the dynamic transaction routing transaction definition or not. It can have either of these values:
YES|NO

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>REMOTE_NOT_FOUND</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION</td>
</tr>
<tr>
<td>DISASTER</td>
<td>LOGIC_ERROR, ABEND, LOOP</td>
</tr>
</tbody>
</table>

XMXE gate, GET_TXN_ENVIRONMENT function
The GET_TXN_ENVIRONMENT function of the XMXE gate is used to acquire a transaction environment for a task that was DS instead XM attached.
Transaction manager domain (XM)

Output parameters

RESPONSE
is the domain's response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID.
Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>DUPLICATE_ENVIRONMENT, ATTACHED_TRANSACTION</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

XMXE gate, FREE_TXN_ENVIRONMENT function

The FREE_TXN_ENVIRONMENT function of the XMXE gate is used to release a transaction environment for a task that was DS instead XM attached.

Output parameters

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID.
Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NO_ENVIRONMENT, ATTACHED_TRANSACTION</td>
</tr>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

Transaction manager domain’s generic gates

Table 114 summarizes the transaction manager domain’s generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 114. Transaction manager domain's generic gates

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>XMDM</td>
<td>XM 0101</td>
<td>PRE_INITIALIZE</td>
<td>DMDM</td>
</tr>
<tr>
<td></td>
<td>XM 0102</td>
<td>INITIALIZE_DOMAIN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>QUIESCE.DOMAIN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TERMINATE_DOMAIN</td>
<td></td>
</tr>
<tr>
<td>XMST</td>
<td>XM 0C01</td>
<td>COLLECT_STATISTICS</td>
<td>STST</td>
</tr>
<tr>
<td></td>
<td>XM 0C02</td>
<td>COLLECTRESOURCE_STATS</td>
<td></td>
</tr>
</tbody>
</table>

For descriptions of these functions and their input and output parameters, refer to the sections dealing with the corresponding generic formats:
Transaction manager domain (XM)

Functions and parameters

Format DMDM—“Domain manager domain’s generic formats” on page 377
Format STST—“Statistics domain’s generic format” on page 1087

Transaction manager domain’s generic format

Table 115 shows the generic format owned by the transaction manager domain, and shows the function performed on the call.

| Table 115. Generic format owned by the transaction manager domain |
| --- | --- | --- |
| Format | Calling module | Function |
| XMNT | DFHXMSR | MXT_NOTIFY |
| | DFHXMAT | MXT_CHANGE_NOTIFY |
| | DFHXMTA |  |
| | DFHXMCL |  |
| XMDN | DFHXMXD | TRANDEF_NOTIFY |
| | DFHXMQD | TRANDEF_DELETE_QUERY |
| | DFHXMDD |  |
| XMPP | DFHXMIQ | FORCE_PURGE_INHIBIT_QUERY |

In the descriptions of the format that follow, the “input” parameters are input not to the transaction manager domain, but to the domain being called by the transaction manager. Similarly, the “output” parameters are output by the domain that was called by the transaction manager domain, in response to the call.

Format XMNT, MXT_NOTIFY function

The MXT_NOTIFY function of XMNT format is used to notify other domains when CICS is at, or no longer at, the maximum task limit for user tasks.

Input parameters

MXTQUEUING
Indicates whether queuing for MXT has just started or just stopped. It can have either of these values:
STARTED|STOPPED

Output parameters

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

Format XMNT, MXT_CHANGE_NOTIFY function

The MXT_CHANGE_NOTIFY function of XMNT format is used to notify other domains of a change to the MXT limit. The called domains indicate whether they can cope with the new limit.

Input parameters

REQUESTED_MXT
The new limit requested for MXT.
Output parameters

ALLOCATED_MXT  
Indicates the limit that the called domain can cope with when the LIMIT_TOO_HIGH exception is returned.

RESPONSE  
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]  
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>LIMIT_TOO_HIGH</td>
</tr>
<tr>
<td>DISASTER</td>
<td>LOGIC_ERROR, ABEND, LOOP</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

Format XMDN, TRANDEF_NOTIFY function

The TRANDEF_NOTIFY function of the XMDN format is used to notify other domains that a transaction definition has been installed, changed, or deleted. The called domain’s can then modify any transaction definition related data they are keeping for that definition.

Input parameters

EVENT  
Indicates the event that has caused the notify to be sent. It can have any of the following values:
INSTALL|CHANGE|DELETE

TRANDEF_TOKEN  
Token identifying the transaction definition instance subject to the above event.

Output parameters

RESPONSE  
is the domain’s response to the call. It can have any of these values:
OK|DISASTER|INVALID|KERNERROR|PURGED

[REASON]  
is returned when RESPONSE is DISASTER or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>LOGIC_ERROR, ABEND, LOOP</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

Format XMDN, TRANDEF_DELETE_QUERY function

The TRANDEF_DELETE_QUERY function of the XMDN format allows other domains to object to the deletion of the named transaction definition.
Transaction manager domain (XM)

**Input parameters**

**TRANSACTION_ID**

The transaction definition subject to the delete request.

**Output parameters**

**INHIBIT_DELETE**

Indicates whether the called domain wants to inhibit the deletion of the named transaction definition. It can either of the following values:

YES|NO

**INHIBIT_REASON**

Indicates the reason why the called domain wants to inhibit the deletion of the named transaction definition. It can have any of the following values:

AID_PENDING|ICE_PENDING|SIT_PARAMETER

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK|DISASTER|INVALID|KERNERROR|PURGED

[REASON]

is returned when RESPONSE is DISASTER or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>LOGIC_ERROR, ABEND, LOOP</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

**Format XMPP, FORCE_PURGE_INHIBIT_QUERY function**

The FORCE_PURGE_INHIBIT_QUERY function of the XMPP format allows other domains to object to the force purge request for the specified transaction.

**Input parameters**

**TRANSACTION_TOKEN**

Token identifying the transaction that is subject to the force purge request.

**Output parameters**

**INHIBIT_PURGE**

Indicates whether the called domain wants to inhibit the force purge of the transaction. It can have either of the following values:

YES|NO

**RESPONSE**

is the domain’s response to the call. It can have any of these values:

OK|DISASTER|INVALID|KERNERROR|PURGED

[REASON]

is returned when RESPONSE is DISASTER or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FUNCTION</td>
</tr>
</tbody>
</table>
## Modules

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHXMAB</td>
<td>XM domain abend program</td>
</tr>
<tr>
<td>DFHXMAT</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>ATTACH</td>
</tr>
<tr>
<td>DFHXMBD</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>START_BROWSE_TRANDEF</td>
</tr>
<tr>
<td></td>
<td>GET_NEXT_TRANDEF</td>
</tr>
<tr>
<td></td>
<td>END_BROWSE_TRANDEF</td>
</tr>
<tr>
<td>DFHXMCL</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>ADD_REPLACE_TCLASS</td>
</tr>
<tr>
<td></td>
<td>ADD_TCLASS</td>
</tr>
<tr>
<td></td>
<td>INQUIRE_TCLASS</td>
</tr>
<tr>
<td></td>
<td>SET_TCLASS</td>
</tr>
<tr>
<td></td>
<td>DELETE_TCLASS</td>
</tr>
<tr>
<td></td>
<td>START_BROWSE_TCLASS</td>
</tr>
<tr>
<td></td>
<td>GET_NEXT_TCLASS</td>
</tr>
<tr>
<td></td>
<td>END_BROWSE_TCLASS</td>
</tr>
<tr>
<td></td>
<td>REGISTER_TCLASS_USAGE</td>
</tr>
<tr>
<td></td>
<td>DEREGISTER_TCLASS_USAGE</td>
</tr>
<tr>
<td></td>
<td>LOCATE_AND_LOCK_TCLASS</td>
</tr>
<tr>
<td></td>
<td>UNLOCK_TCLASS</td>
</tr>
<tr>
<td>DFHXMDD</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>DELETE_TRANDEF</td>
</tr>
<tr>
<td>DFHXMDM</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>PRE_INITIALIZE</td>
</tr>
<tr>
<td></td>
<td>INITIALIZE_DOMAIN</td>
</tr>
<tr>
<td></td>
<td>QUIESCE_DOMAIN</td>
</tr>
<tr>
<td></td>
<td>TERMINATE_DOMAIN</td>
</tr>
<tr>
<td>DFHXMDFS</td>
<td>XM domain offline dump formatting routine</td>
</tr>
<tr>
<td>DFHXMER</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>SET_DEFERRED_MESSAGE</td>
</tr>
<tr>
<td></td>
<td>INQUIRE_DEFERRED_MESSAGE</td>
</tr>
<tr>
<td></td>
<td>SET_DEFERRED_ABEND</td>
</tr>
<tr>
<td></td>
<td>INQUIRE_DEFERRED_ABEND</td>
</tr>
<tr>
<td></td>
<td>REPORT_MESSAGE</td>
</tr>
<tr>
<td></td>
<td>ABEND_TRANSACTION</td>
</tr>
<tr>
<td>DFHXMFD</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>FIND_PROFILE</td>
</tr>
<tr>
<td>DFHXMIQ</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>INQUIRE_TRANSACTION</td>
</tr>
<tr>
<td></td>
<td>SET_TRANSACTION</td>
</tr>
<tr>
<td></td>
<td>START_BROWSE_TRANSACTION</td>
</tr>
<tr>
<td></td>
<td>GET_NEXT_TRANSACTION</td>
</tr>
<tr>
<td></td>
<td>END_BROWSE_TRANSACTION</td>
</tr>
<tr>
<td></td>
<td>START_BROWSE_TXN_TOKEN</td>
</tr>
<tr>
<td></td>
<td>GET_NEXT_TXN_TOKEN</td>
</tr>
<tr>
<td></td>
<td>END_BROWSE_TXN_TOKEN</td>
</tr>
<tr>
<td></td>
<td>INQUIRE_TRANSACTION_TOKEN</td>
</tr>
<tr>
<td></td>
<td>SET_TRANSACTION_TOKEN</td>
</tr>
<tr>
<td></td>
<td>PURGE_TRANSACTION</td>
</tr>
<tr>
<td>DFHXMLD</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>LOCATE_AND_LOCK_TRANDEF</td>
</tr>
<tr>
<td></td>
<td>UNLOCK_TRANDEF</td>
</tr>
</tbody>
</table>
Transaction manager domain (XM)

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHXMQC</td>
<td>Is an internal module which handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>TCLASS_ACQUIRE</td>
</tr>
<tr>
<td></td>
<td>TCLASS_RELEASE</td>
</tr>
<tr>
<td></td>
<td>TCLASS_LIMIT_CHANGE</td>
</tr>
<tr>
<td></td>
<td>TCLASS_QUEUE_CHANGE</td>
</tr>
<tr>
<td>DFHXMQD</td>
<td>Is an internal module which handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>QUIESCE_TRANDEF</td>
</tr>
<tr>
<td></td>
<td>DELETE_INSTANCE</td>
</tr>
<tr>
<td>DFHXMRP</td>
<td>Is an internal module which handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>DEFINITION_RECOVERY</td>
</tr>
<tr>
<td>DFHXMSR</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>INQUIRE_MXT</td>
</tr>
<tr>
<td></td>
<td>SET_MXT</td>
</tr>
<tr>
<td></td>
<td>INQUIRE_DTRTRAN</td>
</tr>
<tr>
<td></td>
<td>SET_DTRTRAN</td>
</tr>
<tr>
<td>DFHXMST</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>COLLECT_STATISTICS</td>
</tr>
<tr>
<td></td>
<td>COLLECT_RESOURCE_STATS</td>
</tr>
<tr>
<td>DFHXMTRI</td>
<td>Interprets XM domain trace entries</td>
</tr>
<tr>
<td>DFHXMXD</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>ADD_REPLACE_TRANDEF</td>
</tr>
<tr>
<td></td>
<td>SET_TRANDEF</td>
</tr>
<tr>
<td></td>
<td>INQUIRE_TRANDEF</td>
</tr>
<tr>
<td></td>
<td>INQUIRE_REMOTE_TRANDEF</td>
</tr>
<tr>
<td>DFHXMXE</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>GET_TXN_ENVIRONMENT</td>
</tr>
<tr>
<td></td>
<td>FREE_TXN_ENVIRONMENT</td>
</tr>
</tbody>
</table>

Exits

There is one specific global user exit point in the transaction manager, XXMATT, which is called during Attach processing.

Note also that the general resource install/discard exit, XRSINDI, is also called by transaction manager to log installs and discards of transaction and tclass definitions.

For further information about both these exit points see the CICS Customization Guide.

Trace

The point IDs for the storage manager domain are of the form XMxxxx; the corresponding trace levels are XM 1, XM 2 and Exc.

For more information about the trace points, see the CICS Trace Entries. For more information about using traces in problem determination, see the CICS Problem Determination Guide.
Chapter 101. Transaction restart program

The transaction restart program, DFHREST, is a user-replaceable module that helps you to determine whether or not a transaction is restarted. The default DFHREST module requests a transaction restart under certain conditions; for example, if a program isolation deadlock occurs (that is, when two tasks each wait for the other to release a particular DL/I database segment), one of the tasks is backed out and automatically restarted, and the other is allowed to complete its update.

For further information about the transaction restart program, see the CICS Recovery and Restart Guide. For information about how to provide your own code for the DFHREST module, see the CICS Customization Guide.

Design overview

In the creation of the program control table (PCT), the system programmer can designate selected transactions as restartable.

During the execution of any transaction, certain temporary-storage data, intrapartition destinations, and files are protected for dynamic backout. In addition, for a restartable transaction, the following actions take place:

- Any terminal input/output area (TIOA), command-level communication area, or terminal user area existing at task initiation is copied to the dynamic log.
- Interval control automatic initiate descriptors (AIDs) used in the task are preserved by means of deferred work elements (DWEs) until the next syncpoint.
- Data is maintained to show:
  - What terminal traffic has occurred during the task
  - Whether a syncpoint has been passed
  - Whether or not the current activation of the task is the result of a restart.

If a transaction abends, but before backout has been attempted, the DFHREST module may be invoked to decide whether or not the task is to be restarted. Even if the DFHREST module decides that the transaction can be restarted, CICS may overrule the restart, for example because of a transaction backout failure.

DFHREST is invoked by DFHXMTA passing a parameter list via a COMMAREA that is mapped by the DFHXMRSD DSECT. DFHREST should return to DFHXMTA, indicating whether or not the transaction should be restarted. If the DFHREST module requests a restart, and CICS does not overrule this decision, the principal facility is not released and the principal facility owner reattaches a new task to restart the transaction.

Notes:

1. The DFHREST module can invoke CICS facilities such as file control and transient data, via the command-level interface.
2. If an error occurs while linking to, or in, the transaction restart program, the restart is not attempted for this task.
3. The DFHREST module runs before backout.
Transaction restart program

Control blocks

CICS supplies a description of the transaction restart program commarea, in Assembler-language, COBOL, PL/I, and C, which maps the layout of the parameter list passed between DFHXMTA and DFHREST. The parameter list contains information that helps you code the DFHREST module to determine whether a restart should be requested for a task.

For a detailed description of this control block, see the CICS Data Areas.

Modules

DFHREST is a skeleton user-replaceable module that you can modify.

Exits

Global user exit points are not relevant for this function.

Trace

Trace point IDs are not relevant for this function.

Statistics

CICS keeps a count of the number of times that each transaction has been restarted.
Chapter 102. Transaction routing

Transaction routing allows one CICS system to run a transaction in another CICS system. The transaction routing facility enables a terminal operator to enter a CICS transaction code into a terminal attached to one CICS system, and thereby start a transaction on another CICS system in a different address space in the same processing system or in another system.

There are two cases of transaction routing:
- Advanced program-to-program communications (APPC); that is, LU6.2
- Non-APPC (for example, LU2).

APPC transaction routing makes use of much of the non-APPC function, and there is often considerable overlap between the function provided by modules for each of the two cases.

The CICS Intercommunication Guide gives a detailed description of transaction routing.

Design overview

Figure 109 on page 1312 shows the overall design of this component.
CICS executes the CICS relay program DFHAPRT (which invokes the user-replaceable dynamic transaction routing program) as follows:

- When a transaction defined with the value DYNAMIC(YES) is initiated.
- When a transaction definition is not found and CICS uses the special transaction defined on the DTRTRAN system initialization parameter. (For more information about DTRTRAN, see the CICS System Definition Guide.)
- Before routing a remote, terminal-oriented, transaction initiated by ATI.
- If an error occurs in route selection.
- At the end of a routed transaction, if the initial invocation requests re-invocation at termination.

If CICS has been generated with the appropriate options for intercommunication, the initialization of CICS with the ISC=YES system initialization parameter specified causes the following modules to be loaded:

- DFHXT (transaction routing data transformation program)
Transaction routing

- DFHZCXR (which includes the DFHZTSP CSECT, the terminal sharing program).

The entry point addresses of these modules are contained in the optional features list that is addressed by CSAOPFLA in the CSA.

The rest of this section is mainly concerned with APPC transaction routing, which occurs when an APPC device is linked through an LU6.2 session to a transaction that is defined as remote.

Overview of operation in the application-owning region for APPC transaction routing

Figure 110 on page 1314 shows the modules in the application-owning region for transaction routing for APPC devices.
Transaction routing

CICS internals | Application program
---|---
DFHTC FREE ALLOCATE Other | EXEC CICS EXEC CICS GDS
DFHEGL FREE ALLOCATE Other | CMxxxx (see note)

DFHETC | DFHETL | DFHEGL | DFHCPI

DFHZISP | DFHZARQ | DFHZARM | DFHZARL

DFHZISP | DFHZXRL

Subroutine ZXRL_FREE | Subroutine ZXRL_ALLOCATE | Subroutine ZXRL_COMMANDS

DFHZISP

DFHRTSU

DFHTSP Subroutine RDETENT

Subroutine RAPPCRE

Note:
CMxxxx represents the names of program calls that are defined in the SAA communications interface.

Figure 110. Transaction routing for APPC devices. Modules in the application-owning region.
**APPC control blocks**

A remote APPC device is defined in the application-owning region with a remote terminal control table system entry (or remote system entry). There are no TCT mode entries or session TCTTE entries associated with the remote system entry when it is defined.

A session with the remote APPC device is represented by a surrogate session TCTTE (or surrogate session entry). The surrogate is built dynamically when the conversation between the systems is initiated, and is deleted when the conversation terminates.

Figure 111 shows the way in which the TCT entries are related.

---

**Remote system entry**

- TCSESKA
- TCT skeleton entry
  - TCTSKSYS
  - TCTSKMDE
  - TCTSKSRE
  - TCTSE for link (owning connection)
- Surrogate session entry
  - TCTTEIST

**Figure 111. Transaction routing for APPC devices.** TCT control-block structure in the application-owning region.

**Remote system entry**: The remote system entry is similar to a normal system entry and, together with the TCT skeleton entry, also includes the following information:

- SYSIDNT of the terminal-owning region (TCTSKSYS)
- SYSIDNT of remote APPC device (local name) (TCTSKID)
- REMOTENAME of APPC device (SYSIDNT on terminal-owning region) (TCTSKHID)
- NETNAME of remote APPC device (TCSESID).

The remote system entry may be defined explicitly with CEDA DEFINE and INSTALL commands.

Alternatively, it is installed dynamically when the first transaction is routed from the remote APPC device. In this case, all data required to build the system entry is included in the initial ATTACH data stream from the application-owning region. No INQUIRE or INSTALL data is sent.
The remote system entry is recorded on the catalog and recovered after warm start and restart. It is located by TMP in the REMOTE domain and SYSTEM domain.

**Surrogate session entry:** The session between the terminal-owning region and the APPC device is represented in the application-owning region by a surrogate session entry.

The surrogate session entry is used to support the routing of commands to the APPC device, and to record security and status information for the conversation.

A surrogate session entry cannot be defined by the user; instead it is created when the conversation is initiated (by an ATTACH request from the APPC device, or an ALLOCATE request from the application-owning region), and is deleted when the conversation ends.

The surrogate session entry is not recorded on the catalog, is not accessible via TC LOCATE, and does not have an entry in the TMP index. It is not recovered after warm start or restart.

CEMT and EXEC CICS INQUIRE or SET commands cannot be used to modify a remote system entry.

**DFHZXRL**

This module forms a principal part of the transaction routing component for APPC devices. It passes DFHLUC macro requests issued in an application-owning region to the terminal-owning region.

All DFHLUC macro requests cause DFHZARL to be invoked. DFHZARL passes a request to DFHZXRL if the TCTE address passed is for a surrogate session, and the request is one that DFHZXRL is known to handle (apart from ALLOCATE). ALLOCATE requests are always routed from DFHZARL to DFHZISP. DFHZISP is then responsible for calling DFHZXRL if the system from which a session is to be allocated is found to be remote. Table 116 summarizes this and shows which of the three main routines in DFHZXRL is called. ZXRL_ALLOCATE, ZXRL_COMMANDS, and ZXRL_FREE are described in “ALLOCATE processing in the application-owning region” on page 1318, “Other LU6.2 command processing in the application-owning region” on page 1320, and “FREE processing in the application-owning region” on page 1320 respectively.

**Table 116. DFHZXRL's processing of DFHLUC requests**

<table>
<thead>
<tr>
<th>DFHLUC request</th>
<th>DFHZXRL's caller</th>
<th>DFHZXRL routine called</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALLOCATE</td>
<td>DFHZISP</td>
<td>ZXRL_ALLOCATE</td>
</tr>
<tr>
<td>ISSUE-ABEND</td>
<td>DFHZARL</td>
<td>ZXRL_COMMANDS</td>
</tr>
<tr>
<td>ISSUE-ATTACH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ISSUE-CONFIRMATION</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ISSUE-ERROR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ISSUE-SIGNAL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RECEIVE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SEND</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WAIT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXTRACT-PROCESS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FREE</td>
<td>DFHZARL</td>
<td>ZXRL_FREE</td>
</tr>
</tbody>
</table>

The input and output for DFHZXRL is provided by means of the LUC parameter list, that is, the parameter list which is built by the DFHLUC macro. DFHZARL
passes the LUC parameter list to DFHZXRL unaltered. If the LUC parameter list previously contained only the SYSID name, DFHZISP adds the address of the remote system entry to the LUC parameter list before passing it to DFHZXRL.

DFHZXRL calls routine RAPPCRE of DFHZTSP to build the surrogate TCTTE representing the session with the APPC device, and DFHZISP calls routine RDETENT to free it.

**ATTACH processing in the application-owning region**
The following describes how a transaction is attached in the application-owning region when the attach request has been routed from the terminal-owning region.

**DFHZSUP module:**

1. Issues DFHSEC TYPE=CHECK,RESTYPE=TRAN to validate transaction security against the security values associated with the intersystem link at bind time.

2. Processes the incoming attach FMH5.
   For an LU6.2 ISC connection:
   - Sets the TCTTE to indicate a mapped or unmapped conversation.
   - Validates syncllevel requested in FMH5 against the value negotiated at bind time.
   - Moves the TPN from the FMH5 to the TCA extension.
   - Performs attach-time security processing, as defined by the ATTACHSEC parameter in the resource definition for the LUC CONNECTION to the terminal-owning region. This may change the security values associated with the link from the bind-time established values that were checked in step 1) to user-level values, obtained from the SNT for a userid specified in the FMH5.

   For an MRO connection:
   - Issues DFHZIRCT FN=ZSUP to extract the USERID and UOW-ID from the LU6.2 style FMH5.
   - Performs attach-time security processing, as defined by the ATTACHSEC parameter in the resource definition for the LUC CONNECTION to the terminal-owning region. This can change the security values associated with the link from the bind-time established values that were checked in step 1) to user-level values, obtained from the SNT for a userid specified in the FMH5.
   - Deletes the LU6.2-style FMH5 from the front of the data stream.

3. Issues DFHZUSRM TYPE=SET,REQUEST=ATTACH_INBOUND and DFHLUC TYPE=INIT-CALL macros to move input data into a buffer bypassing the FMH5 ATTACH header.

4. PIP processing is bypassed because PIP is never present on an attach from a terminal-owning region when transaction routing.

5. Puts the remaining data into a TIOA with a DFHTC TYPE=(READ,WAIT),NOATNI=YES.

6. Issues a DFHIS TYPE=RATT, to call DFHZTSP to build a surrogate session entry to represent the session TCTTE in the terminal-owning region.

7. Assign the security values established for the link to the surrogate, as preset security values are shipped from the terminal-owning region, and cannot be defined on the application-owning region.

ATTACH security processing in DFHZSUP has established two SNTTEs associated with the link session:
Transaction routing

a. The SNTTE pointed to by TCTELSNT in the LU6.2 extension or TCTEIRSN for MRO represents link-level security values established at bind time.

b. The SNTTE pointed to by TCTTESNT represents user-level security values established during ATTACH security processing.

TCTTESNT is copied to the surrogate TCTTE. No provision is made for preset user security values to override the TCTTESNT value.

Preset security values defined for the terminal session on the terminal-owning region are processed only on that system, during local attach processing. The SNTTE then associated with the local TCTTE is used to build the routed attach FMH5.

At transaction end, no SNTTEs addressed by the surrogate are deleted when the surrogate is deleted. This is done, if necessary, as part of the termination of the LINK SESSION.

Each system in a “daisy chain” imposes its own link security requirements. An intermediate system with a lower level of security would route the ATTACH with lower security (that is, no USERID or verified bit) which could cause it to be rejected by the next system in the chain.

8. Passes control to the requested application program.

DFHZTSP module:
1. Performs initialization housekeeping, checks the link TCTTE and TIOA.
2. Locates remote system entry from the TMP REMOTE domain. If not found, attaches the CITS transaction (DFHZATS) to install it.
3. Builds surrogate session TCTTE.
4. Gets a TIOA and chains it to the surrogate.
5. Issues DFHIS TYPE=XTP, XFNUM=2 to call DFHXTP.
6. Chains surrogate to TCA and Link TCTTE.
7. Copies link operator dispatching priority from the link and establishes dispatching priority for the surrogate.

DETACH processing in the application-owning region
At transaction end, routine RDETENT of DFHZTSP is called to delete the surrogate session entry. The remote system entry is not deleted, and can be used by a subsequent transaction routing request, by an ATI request, or by an ALLOCATE request issued in the application-owning region.

ALLOCATE processing in the application-owning region
A session can be allocated as a result of either of the following macro calls:
- DFHLUC TYPE=ALLOCATE
- DFHTC TYPE=ALLOCATE

The DFHLUC call invokes DFHZARL, which passes control to DFHZISP, the module that handles allocation and freeing of sessions. The DFHTC call invokes DFHZISP directly.

DFHZISP locates the TCTSE for the system identified on the ALLOCATE request.

The request is routed to DFHZXRL if the following conditions hold:
- The system is LU6.2
The system is remote

DFHZISP was called as a result of a DFHTC TYPE=ALLOCATE request (which is the case when DFHZISP is called from DFHZARL).

The address of the remote TCTSE is inserted in the parameter list passed to DFHZXRL.

If a Privileged Allocate request is made, the transaction abends, because the request is not permitted for a remote system.

**DFHZXRL module:** For an ALLOCATE request, control passes to subroutine ZXRL_ALLOCATE which establishes a session between the application-owning region and the alternate facility, and builds a surrogate session TCTTE.

Subroutine ZXRL_ALLOCATE:
1. Checks that the parameter list contains the TCTSE address for the remote LU6.2 system.
2. Obtains the address of the TCTSE of the system to which the LU6.2 commands are to be routed.
3. Allocates a session to the terminal-owning region.
   - The connection between the terminal-owning region and application-owning region which supports remote alternate facilities may be an LU6.2 ISC connection or an MRO connection. Subroutine ZXRL_ALLOCATE allocates the session using a DFHTC TYPE=ALLOCATE macro call that can allocate a session on either type of connection.
   - The default profile DFHCICCSR is used; this may specify the modename for an LU6.2 connection. The modename specified on the EXEC CICS ALLOCATE is not used here, but is shipped to the terminal-owning region where it is used to allocate an LU6.2 session between the terminal-owning region and the APPC device.
   - The queuing option (NOQUEUE!NOSUSPEND) specified on the ALLOCATE request by the caller is used when the DFHTC TYPE=ALLOCATE macro call is issued for the connection. If NOQUEUE is not specified, the request may also be queued when it is issued in the terminal-owning region. If a session failure occurs during this period, the transaction in the application-owning region and the relay transaction in the terminal-owning region abend.
   - If a session between the application-owning region and terminal-owning region cannot be allocated:
     - When the failure is due to CICS logic, corruption of CICS storage, or incorrect resource definition by the user, the transaction abends.
     - When the failure is due to other conditions (such as session failure or ‘SYSBUSY’), an appropriate return code is passed to the caller.

   The return code is handled so as to minimize the differences between local and remote APPC devices as seen by the user of the DFHLUC interface. The actions available are:
   - Where the condition could be encountered with a local terminal, reflect the return code to the caller in LUCRCOD2 and LUCRCOD3 with LUCESYSI (X’01’) in LUCRCOD1.
   - Where the condition would not occur with a local terminal, reflect a different return code to the caller.
4. Issues a DFHIS TYPE=XTP,XFNUM=3 macro call that invokes a stream that is passed to the terminal-owning region.
5. Issues a DFHTC TYPE=(WRITE, WAIT, READ), FMH=YES macro call to send the request to the terminal-owning region and receive the response.

6. Issues a DFHIS TYPE=RALL that invokes DFHZTSP to build a surrogate session TCTTE, then chains the link session TCTTE and the surrogate session TCTTE together.

7. Issues a DFHIS TYPE=XTP, XFNUM=2 macro call that invokes DFHXTP to unwrap the response from the terminal-owning region and update the surrogate session TCTTE and the parameter list created by the DFHLUC macro.

8. Examines the return codes in the response:
   - If the request has been successful, returns the surrogate session TCTTE address to the caller.
   - If the request has not been successful, issues a DFHIS TYPE=RDET macro call to free the surrogate session TCTTE.

FREE processing in the application-owning region
One of the following macro calls is made in the application-owning region to request that a surrogate session TCTTE should be freed:

- DFHLUC TYPE=FREE
- DFHTC TYPE=FREE

The DFHLUC TYPE=FREE call invokes DFHZARL, which passes control to DFHZXRL; and subroutine ZXRL_FREE in DFHZXRL is then called to issue a DFHTC TYPE=FREE request against the surrogate. The DFHTC TYPE=FREE call invokes DFHZISP.

DFHZISP:
1. Bypasses security processing (sign-off) for a surrogate session entry, because the sign-off is performed for the link.
2. Issues the DFHIS TYPE=RDET macro that calls DFHZTSP to free the surrogate and link TCTTEs.

Other LU6.2 command processing in the application-owning region
Most SAA communications calls, EXEC CICS GDS commands, and EXEC CICS commands relating to LU6.2 sessions cause a call to DFHZARL using the DFHLUC macro.

The EXEC CICS SYNCPOINT, EXEC CICS SYNCPOINT ROLLBACK, and EXEC CICS (GDS) ISSUE PREPARE commands are handled under the control of the syncpoint program, which uses DFHLUC macro requests to send syncpoint flows on LU6.2 sessions, and DFHTC macro calls to end any dangling conversations.

DFHTC macro requests: DFHTC macro requests may be issued against surrogate session TCTTEs. Unlike requests for other surrogate TCTTEs, which are passed to DFHZTSP, DFHZARQ handles these requests in the same way as other requests against LU6.2 sessions: they are passed to DFHZARM which in turn calls DFHZARL. Within DFHZARL, requests are handled in a similar way to those initiated by the DFHLUC macro.

DFHLUC requests: DFHLUC requests are passed to DFHZARL: when the session is a surrogate, the request is passed to DFHZXRL (routine ZXRL_COMMANDS).

DFHZXRL module: Input to routine ZXRL_COMMANDS in DFHZXRL is the application command in the form of a DFHLUC macro call parameter list.
1. **ZXRL_COMMANDS** normally wraps up the command to be shipped and relevant TCTTE fields by calling a transformer routine in DFHXTP. However, if the first syncpoint flow has been received, then:
   - Application requests ISSUE-ERROR and ISSUE-ABEND are sent unwrapped on the link session.
   - All other requests are rejected with a state error.

2. **ZXRL_COMMANDS** tests the state of its link with the terminal-owning region (this may not be the same as the state of the application):
   - If it finds that it is in ‘RECEIVE’ state, it issues a DFHTC TYPE=(READ,WAIT) in order to receive the change direction (CD) indicator from the terminal-owning region. Except during syncpoint processing, however, the session is normally in ‘SEND’ state when a command is issued.

3. **ZXRL_COMMANDS** then sends the wrapped-up request to the remote system using the DFHTC macro. To reduce the number of flows when the command may result in the termination of the conversation, the following rules are applied for both MRO and ISC links:
   - If the application command is SEND LAST WAIT and the application program is in ‘SEND’ state, the command is sent using a DFHTC TYPE=(WRITE,LAST) macro.
   - If the application command is WAIT and the application program is in ‘FREE PENDING AFTER SEND LAST’ state, the command is sent using a DFHTC TYPE=(WRITE,LAST) macro.
   - If the end bracket (EB) indicator has been sent to the terminal-owning region all other commands result in a state error return code.

   In other cases and when the link between the terminal-owning region and application-owning region is MRO, **ZXRL_COMMANDS** issues a DFHTC TYPE=(WRITE,WAIT,READ).

   However, when the link is LU6.2, the following additional rules are applied in order to exploit the buffering provided by LU6.2:
   - When the application’s command is a SEND and the application is in ‘SEND’ state **ZXRL_COMMANDS**, issues a DFHTC TYPE=(WRITE,WAIT) macro to send the request without waiting for a response.
   - When the application’s command is a SEND and the application is not in ‘SEND’ state **ZXRL_COMMANDS**, issues a DFHTC TYPE=(WRITE,WAIT,READ) so that it can get the state error back from the remote system immediately.
   - For all other commands, including SEND INVITE and so on, **ZXRL_COMMANDS** issues a DFHTC TYPE=(WRITE,WAIT,READ).

4. **ZXRL_COMMANDS** receives the response to its DFHTC macro call. This may be:
   - An ATNI or ATND abend. **ZXRL_COMMANDS** frees the link session and returns ‘TERMERR’ to the application.
   - ‘SIGNAL’, which is used by the terminal-owning region when it is in ‘RECEIVE’ state to indicate to the application-owning region that there is an abnormal response pending. **ZXRL_COMMANDS** issues a DFHTC TYPE=(WRITE,WAIT,READ) to send the change direction indicator and get the abnormal response from the terminal-owning region.

5. When the DFHTC macro included a READ, and the request was successfully processed, **ZXRL_COMMANDS** checks for a wrapped reply from the
Transaction routing

terminal-owning region, and calls DFHXT to unwrap the reply. When the resultin
dfhluc parameter list indicates syncpoint or syncpoint rollback, and the link is an MRO connection, zxrl commands issues a dfhlc type=read, because there is a syncpoint or rollback flow pending.

When there is no wrapped reply, zxrl commands checks for syncpoint rollback received (the only possibility under these circumstances).

LU6.2 daisy-chaining considerations

There is no special-case code to distinguish between the terminal-owning region and an intermediate system. When DFHZXRT has interpreted a request received from the application-owning region, it issues the LU6.2 service request (DFHLUC) macro call with the parameter list that was created in the application-owning region. The macro generates a call to DFHZARL. If the TCTE is a surrogate, which is the case in an intermediate system, control passes to DFHZXRL as described above.

Overview of operation in the terminal-owning region for APPC transaction routing

Figure 112 shows the modules in the terminal-owning region for transaction routing for APPC devices.

![Figure 112. Transaction routing for APPC devices. Modules in the terminal-owning region.](image)

In the terminal-owning region, operation is under the control of a relay program. When transaction routing is initiated from the APPC device, the relay program is DFHAPRT (which is also used for non-APPC devices). When transaction routing is initiated by an allocate request in the application-owning region, the relay program is DFHCRT. Both relay programs call DFHZTSP, which calls DFHZXRT.

When an APPC device initiates a conversation with an application in the application-owning region, relay program DFHAPRT is started in the terminal-owning region. It calls the ROUTENT routine of DFHZTSP, which
allocates a session to the application-owning region and starts the requested transaction there (see “ATTACH processing in the terminal-owning region”).

When an application running in the application-owning region initiates a conversation with a remote APPC device by issuing an ALLOCATE request, the DFHCRT relay program is started in the terminal-owning region. It calls the ALLOCLUC routine of DFHZTSP which allocates a session to the APPC device (see “Chapter 65. Program error program” on page 879).

After a conversation has been started by either method, the LU6.2 commands passed from the application-owning region are processed by DFHZXRT, which issues the LU6.2 service request (DFHLUC) macro with an appropriate parameter list against the APPC device.

### ATTACH processing in the terminal-owning region

The following flow describes the steps involved in routing a transaction from an APPC device across an LU6.2 intersystem link.

#### DFHZSUP module:

1. Processes the incoming FMH5 from the terminal. This:
   - Sets TCTTE to indicate mapped or unmapped conversation.
   - Validates synclevel requested in FMH5 against the value negotiated at bind time.
   - Moves the TPN from the FMH5 to the TCA extension.
   - Performs attach-time security processing, as defined by the ATTACHSEC parameter in the resource definition for the APPC device (or CONNECTION). This may change the security values associated with the terminal from the default link-level values to user-level values, obtained from the SNT for a user who is signed on.
2. Checks transaction security code against new security levels developed during ATTACH security processing above.
3. Issues DFHSEC TYPE=CHECK,RESTYPE=TRAN to validate transaction security against the security values associated with the terminal (and with the user, if signed on).
4. Issues DFHZUSRM TYPE=SET,REQUEST=ATTACH_INBOUND and DFHLUC TYPE=INIT-CALL macros to move input data into a buffer bypassing the FMH5 ATTACH header.
5. If PIP is present, builds a new TCA extension and moves the PIP data into it by issuing a DFHLUC TYPE=RECEIVE (which also causes the PIP data to be deleted from the buffer).
6. Puts remaining mapped data into a TIOA with a DFHTC TYPE=(READ,WAIT),NOATNI=YES.
7. Issues DFHPC TYPE=XCTL to the relay program DFHAPRT.

#### DFHAPRT module:

1. Drives the dynamic routing exit if the transaction has been defined as dynamic.
2. Sets up the DFHISCRQ parameter list with remote sysid and tranid.
3. Recognizes that the principal facility is an APPC device.
4. Issues DFHIS macro to invoke DFHZTSP.

#### DFHZTSP module:
1. If the transaction has been defined with an associated TRPROF, the profile named is located with a DFHKC CTYPE=PROFLOC; otherwise the default DFHICSS profile is used.

2. Issues DFHTC TYPE=ALLOCATE,REQUID=CSRR to allocate a session to the remote system using the profile identified in step 1.

3. Flags the returned TCTTE as a relay link and puts the remote sysid into TCTESYID in the terminal TCTTE. If the LINK TCTTE status is ‘COLD’, issues DFHTC CTYPE=CATALOG.

4. Sets up the transformer parameter list (DFHXTSTG) to indicate ATTACH FMH5 required, COLD or not COLD, and transaction routing for an APPC device, passing the tranid, user TCTTE, and link TCTTE.

5. Issues DFHIS TYPE=XTP ,XFNUM=1 to call the transformer program, DFHXTP, to build the data. (See "Transformer program (DFHXTP)" on page 1324.)

6. Issues DFHTC TYPE=(WRITE,WAIT,READ) against the link to route the ATTACH request to the application-owning region. This causes DFHZARM (when the link is ISC) or DFHZIS2 (when the link is MRO) to add an LU6.2 FMH5 preceding the LU6.1 FHM5 built by XTP. This contains security data required to validate the request at the application-owning region.

**ALLOCATE processing in the terminal-owning region**

**DFHCRT module:** Transaction CXRT (program DFHCRT) is started in the terminal-owning region when the attach FMH5 is received from the application-owning region

Program DFHCRT:
1. Checks that the principal facility of the task is an ISC or MRO session. If not, and if it is a terminal, a message is written to the facility, and the transaction terminates.
2. Issues DFHIS TYPE=ALLOC macro which calls DFHZTSP.

**DFHZTSP module:** The ALLOCLUC routine of DFHZTSP is invoked when the DFHIS TYPE=ALLOC macro is issued. This routine is called with input from the application-owning region in a TIOA.

Routine ALLOCLUC:
1. Issues DFHIS TYPE=XTP, XFNUM=4 which updates the TCTTE and builds a parameter list of the type created by the DFHLUC macro.
2. Verifies that the parameter list contains an ALLOCATE request (the only valid request at this stage). If it does not, the transaction abends.
3. Issues a DFHLUC MF=E macro with the supplied parameter list.
4. If the request is successful, DFHZTSP:
   a. Issues DFHIS TYPE=XTP, XFNUM=1 which wraps the updated TCTTE and DFHLUC parameter list ready for transmission to the application-owning region.
   b. Issues a DFHTC TYPE=(WRITE,WAIT,READ) against the session with the application-owning region.
   c. Passes control to DFHZXRT. The TIOA received with the preceding DFHTC request should contain data for one of the requests that DFHZXRT handles.
5. If the request is unsuccessful, DFHZTSP:
Transaction routing

- Issues DFHIS TYPE=XTP, XFNUM=1 which wraps the updated TCTTE and DFHLUC parameter list ready for transmission to the application-owning region.
- Issues DFHTC TYPE=(WRITE, LAST) to send the response to the application-owning region.
- Frees the session with the application-owning region.

FREE processing in the terminal-owning region

When an end-bracket has flowed from the application-owning region to the terminal-owning region as a result of an application command (for example, EXEC CICS SEND LAST), and the corresponding command has been issued in the terminal-owning region against the terminal, DFHZXRT issues a DFHLUC TYPE=FREE macro against the terminal, and a DFHTC TYPE=FREE macro against the link to the application-owning region.

Other LU6.2 command processing in the terminal-owning region

DFHZXRT is called by DFHZTSP following a DFHTC TYPE=(WRITE, WAIT, READ) macro. The reply received from the application-owning region is processed as follows:

1. If an application request has been received, DFHZXRT:
   - Calls DFHXTP to unwrap the application program’s request
   - Issues the DFHLUC macro call with the parameter list created in the application-owning region
   - Calls DFHXTP to wrap the response to the DFHLUC macro
   - Sends the response to the application-owning region.

   Normally the wrapped terminal response is sent to the application-owning region with a DFHTC TYPE=(WRITE, WAIT, READ) macro. However, there are exceptions:

   - If the response to the DFHLUC macro call is a request for SYNCPOINT ROLLBACK, DFHZXRT sends the wrapped terminal response with a DFHTC TYPE=WRITE macro and then issues a DFHSP TYPE=ROLLBACK command.

   - If the response to the DFHLUC macro call is a request for SYNCPOINT, DFHZXRT sends the wrapped terminal response with a DFHTC TYPE=WRITE macro and then issues a DFHSP TYPE=PREPARE against the link.

   The response to the macro is processed in the same way as when a SYNCPOINT request is received from the application, and issued to the terminal, except that the roles of the terminal and link are reversed.

   - If the session to the terminal has been freed by an application command, DFHZXRT sends the wrapped terminal response with a DFHTC TYPE=(WRITE, LAST) macro.

   - When the session to the application-owning region is in ‘RECEIVE’ state, normally DFHZXRT issues a DFHTC TYPE=READ to get the next request from the application.

   However, if the link between the terminal-owning and application-owning regions is LU6.2, and the response to the DFHLUC macro issued to the terminal indicates that the terminal has issued one of ISSUE_SIGNAL, ISSUE_ERROR, ISSUE_ABEND, or SYNCPOINT_ROLLBACK, DFHZXRT issues an ISSUE_SIGNAL against the link with the application-owning region to notify the application-owning region that the terminal-owning region wants to send. It then issues a series of DFHTC TYPE=READ macros until it receives the change of direction indicator.
The data is processed in the normal way when ‘SIGNAL’ is received from the terminal. In the other cases, that is, if a negative response is received from the terminal, the data from the application-owning region is purged. After the change direction indicator is received, DFHZXRT sends the response to the application-owning region, ISSUE_SIGNAL and ISSUE_ERROR are sent using a DFHTC TYPE=(WRITE, WAIT, READ) macro, ISSUE_ABEND is sent using a DFHTC TYPE=(WRITE, LAST) macro, and SYNCPOINT_ROLLBACK is sent using a DFHTC TYPE=WRITE macro.

If the response from the terminal was ‘ROLLBACK’, by a DFHSP TYPE=ROLLBACK macro is issued.

2. If a syncpoint request has been received, DFHZXRT:
   - Issues a DFHLUC TYPE=ISSUE-PREPARE macro against the terminal TCTTE.
   - Checks the terminal’s response:
     - If the terminal response indicates that a SYNCPOINT or BACKOUT request was issued, DFHSPP is called.
     - If the terminal response indicates that the terminal issued a SEND_ERROR request, DFHZXRT issues a DFHTC CTYPE=ISSUE_ERROR macro followed by a DFHTC TYPE=(WRITE, WAIT, READ) macro against the link session.
     - If the terminal response indicates that the terminal issued DEALLOCATE(ABEND), DFHZXRT issues a DFHTC CTYPE=ISSUE_ABEND macro against the link session. It then frees the link with the application-owning region and returns.

3. If a syncpoint rollback request has been received, DFHZXRT issues a SYNCPOINT ROLLBACK request.

When DFHZXRT detects that EB has flowed on both the session with the terminal and the session with the application-owning region, it issues DFHTC TYPE=FREE on both and returns.

Transformer program (DFHXTP)

The terminal-sharing data-transformation program, DFHXTP, constructs and interprets the data streams flowing between terminal-owning and application-owning regions, for both APPC and non-APPC transaction routing environments.

It does this by using four transformers. These either wrap this data from the surrogate TCTTE (in the AOR) or the real TCTTE (in the TOR) into the link TCTTE’s TIOA, or they unwrap this data from the link TCTTE’s TIOA into the surrogate or real TCTTE.

The transformers work in matching wrap and unwrap pairs. Transformer 1 wraps any data to be sent from a TOR to an AOR, which is then unwrapped in the AOR by transformer 2. Transformer 3 wraps any data to be sent from an AOR to a TOR, which is then unwrapped in the TOR by transformer 4. Figure 113 on page 1327 shows this process.
The transformer program is capable of shipping data from the TCTTE and the following control blocks that are chained off the TCTTE:

- The TCTTE extension, chained off TCTTETEA in the TCTTE.
- The terminal partition extension, chained off TCTTETPA in the TCTTE BMS extension.
- The TCTTE user extension, chained off TCTTECIA in the TCTTE.
- The SNTTE, chained off TCTTESNT in the TCTTE.
- The DFHLUC parameter list, and fields chained off it.

Note that because this field is not chained off the TCTTE but is in LIFO, its address is passed as a parameter to the transformer program.

- The TCA extension for LU6.2 communication.
- Fields from the terminal control table system entry (TCTSE), chained off TCTTEIST in the TCTTE.
- Fields from the terminal control table mode entry (TCTME), chained off TCTTEMOD in the TCTTE.
- The data interchange block (DIB), chained off TCTEDIBA in the TCTTE.

The fields to be shipped are defined in tables in the transformer program.

There is special-case code to deal with fields that cannot be processed by the table-driven code.

For the transaction routing of LU6.2 commands, DFHXTP must ensure that the data stream built for transmission contains all the information relevant to support the issuing of a DFHLUC macro request on the remote system. This information consists primarily of:

- The DFHLUC parameter list
- Any data addressed by the parameter list
- The conversation state machine (TCTEUSRS in DFHTCTZE) in the TCTTE
- TCTTE fields required to build the surrogate TCTTE, in particular:
  - The synclvel supported by the terminal
  - The information returned to the application by the EXTRACT PROCESS command.

**Data streams for transaction routing**

Figure 114 on page 1328 shows the types of transaction-routing data streams.
The transformer builds four types of data stream for transaction routing:

1. **Attach data stream for principal facility**
   - Built by transformer 1
   - Shipped from TOR to AOR
   - Unwrapped by transformer 2
   - Contains an LU6.1 attach FMH (FMH5)
   - For LU6.2, the routed data does not contain a DFHLUC parameter list.

2. **Attach data stream for alternate facility**
   - Built by transformer 3
   - Shipped from AOR to TOR
   - Unwrapped by transformer 4
   - Contains an LU6.1 attach FMH (FMH5)
   - For LU6.2, the routed data contains a DFHLUC parameter list.

3. **DFHLUC request data stream**
   - Built by transformer 3
   - Shipped from AOR to TOR
   - Unwrapped by transformer 4
   - For LU6.2, the routed data contains a DFHLUC parameter list.

4. **DFHLUC response data stream**
   - Built by transformer 1
   - Shipped from TOR to AOR
   - Unwrapped by transformer 2
   - For LU6.2, the routed data contains a DFHLUC parameter list.
**Transaction routing**

**Note:** The first transformer request for remote alternate facilities is to transformer 3, and not to transformer 1. This is because the same transformers are used whether transaction routing is initiated in the terminal-owning region or in the application-owning region.

An LU6.1 attach FMH5 is used when a transaction is to be started in the system to which the request is sent. CSRR is specified as the return process to indicate the use of transaction routing. In the case of routing to the application-owning region, the transaction is the user transaction; in the case of routing to the terminal-owning region, the transaction is the CXRT relay transaction.

**Transaction-routed data format**

Figure 115 shows the format of the data stream passed between a TOR and an AOR to provide transaction routing from any supported device.

The fields that are shipped depend principally on the type of terminal and on other parameters, as follows:

<table>
<thead>
<tr>
<th>Code</th>
<th>Hex</th>
<th>Type</th>
<th>Field Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>01</td>
<td>V</td>
<td>XTPCDTC1 TC request bytes or attach start code</td>
</tr>
<tr>
<td>2</td>
<td>02</td>
<td>V</td>
<td>XTPCDOPC Operator class</td>
</tr>
<tr>
<td>3</td>
<td>03</td>
<td>V</td>
<td>XTPCDTUA TCTTE user area</td>
</tr>
<tr>
<td>4</td>
<td>04</td>
<td>V</td>
<td>XTPCDTIA Terminal I/O area</td>
</tr>
<tr>
<td>5</td>
<td>05</td>
<td>V</td>
<td>XTPDCDCA COMMAREA</td>
</tr>
<tr>
<td>6</td>
<td>06</td>
<td>V</td>
<td>XTPCDLPS Terminal partition set</td>
</tr>
<tr>
<td>7</td>
<td>07</td>
<td>V</td>
<td>XTPCDPLM Page LDC mnemonic</td>
</tr>
<tr>
<td>8</td>
<td>08</td>
<td>V</td>
<td>XTPCDPGD Page data</td>
</tr>
<tr>
<td>9</td>
<td>09</td>
<td>V</td>
<td>XTPCDRQI Request ID</td>
</tr>
<tr>
<td>10</td>
<td>0A</td>
<td>V</td>
<td>XTPCDET1 Error terminal ID</td>
</tr>
<tr>
<td>11</td>
<td>0B</td>
<td>V</td>
<td>XTPCDETL Error terminal LDC</td>
</tr>
<tr>
<td>12</td>
<td>0C</td>
<td>V</td>
<td>XTPCDMCF Message control flags</td>
</tr>
<tr>
<td>13</td>
<td>0D</td>
<td>V</td>
<td>XTPCDTTL Message title</td>
</tr>
<tr>
<td>14</td>
<td>0E</td>
<td>V</td>
<td>XTPCDRTT Route target ID: netname.termid.ldc.opid</td>
</tr>
<tr>
<td>15</td>
<td>0F</td>
<td>V</td>
<td>XTPCDCPS Application partition set</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>F</td>
<td>DFHTCCTTE TCTTEAID Automatic initiate descriptor</td>
</tr>
<tr>
<td>17</td>
<td>11</td>
<td>F</td>
<td>DFHTCCTTE TCTTECAD Cursor address</td>
</tr>
<tr>
<td>18</td>
<td>12</td>
<td>F</td>
<td>DFHTCCTTE TCTESIDO Outbound signal data</td>
</tr>
<tr>
<td>19</td>
<td>13</td>
<td>F</td>
<td>DFHTCCTTE TCTESIDI Inbound signal data</td>
</tr>
</tbody>
</table>

**Figure 115. Routed data format**

The length field in Figure 115 depends upon whether the field type is described in the table that follows as being V (Variable), F (Fixed), or U (Undefined). A V field is 2 bytes in length, an F field is 1 byte, and U indicates a variable that is no longer wrapped or unwrapped if it is encountered.

Table 50 on page 451 shows the various data fields that may appear in a transaction routing data stream, together with their codes and field types.
### Transaction routing

**Table 117. Transaction routing data stream (continued).** Built by the terminal sharing transformer (DFHXTP).

<table>
<thead>
<tr>
<th>Code</th>
<th>Hex</th>
<th>Type</th>
<th>DSECT</th>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>14</td>
<td>F</td>
<td>DFHTCTTE</td>
<td>TCTE32SF</td>
<td>Screen size attributes</td>
</tr>
<tr>
<td>21</td>
<td>15</td>
<td>F</td>
<td>DFHTCTTE</td>
<td>TCTE32FX</td>
<td>Transparency attributes</td>
</tr>
<tr>
<td>22</td>
<td>16</td>
<td>F</td>
<td>DFHTCTTE</td>
<td>TCTE32MN</td>
<td>Map set name</td>
</tr>
<tr>
<td>23</td>
<td>17</td>
<td>F</td>
<td>DFHTCTTE</td>
<td>TCTECRE</td>
<td>Request completion extension</td>
</tr>
<tr>
<td>24</td>
<td>18</td>
<td>F</td>
<td>DFHTCTTE</td>
<td>TCTETECR</td>
<td>Request completion analysis</td>
</tr>
<tr>
<td>25</td>
<td>19</td>
<td>F</td>
<td>DFHTCTTE</td>
<td>TCTEDES</td>
<td>TCAM destination name</td>
</tr>
<tr>
<td>26</td>
<td>1A</td>
<td>F</td>
<td>DFHTCTTE</td>
<td>TCTETETM</td>
<td>Terminal model number</td>
</tr>
<tr>
<td>27</td>
<td>1B</td>
<td>F</td>
<td>DFHTCTTE</td>
<td>TCTETID</td>
<td>Teller identification for 2980</td>
</tr>
<tr>
<td>28</td>
<td>1C</td>
<td>F</td>
<td>DFHTCTTE</td>
<td>TCTEOI</td>
<td>Operator identification</td>
</tr>
<tr>
<td>29</td>
<td>1D</td>
<td>F</td>
<td>DFHTCTTE</td>
<td>TCTEEDF</td>
<td>EDF mode</td>
</tr>
<tr>
<td>30</td>
<td>1E</td>
<td>F</td>
<td>DFHTCTTE</td>
<td>TCTETC</td>
<td>Nominated transaction</td>
</tr>
<tr>
<td>31</td>
<td>1F</td>
<td>F</td>
<td>DFHTCTTE</td>
<td>TCTETS</td>
<td>Terminal status</td>
</tr>
<tr>
<td>32</td>
<td>20</td>
<td>U</td>
<td>DFHSNTTE</td>
<td>SNTENLS</td>
<td>User's national language</td>
</tr>
<tr>
<td>33</td>
<td>21</td>
<td>F</td>
<td>DFHTCTTE</td>
<td>TCTEASCZ</td>
<td>Alternate screen size attributes</td>
</tr>
<tr>
<td>34</td>
<td>22</td>
<td>F</td>
<td>DFHTCTTE</td>
<td>TCTEASCL</td>
<td>TCTEASCCL</td>
</tr>
<tr>
<td>35</td>
<td>23</td>
<td>F</td>
<td>DFHTCTTE</td>
<td>TCTEASCC</td>
<td>3270 extended feature flags</td>
</tr>
<tr>
<td>36</td>
<td>24</td>
<td>F</td>
<td>TCTTETTE</td>
<td>TCTEAPGL</td>
<td>Alternate page size</td>
</tr>
<tr>
<td>37</td>
<td>25</td>
<td>F</td>
<td>DFHTCTTE</td>
<td>TCTEAPGC</td>
<td>3270 text feature flag</td>
</tr>
<tr>
<td>38</td>
<td>26</td>
<td>F</td>
<td>DFHTCTTE</td>
<td>TCTETXTF</td>
<td>3270 extended feature flags</td>
</tr>
<tr>
<td>39</td>
<td>27</td>
<td>F</td>
<td>TCTTELUC</td>
<td>TCTECVT</td>
<td>LU6.2 conversation state machine</td>
</tr>
<tr>
<td>40</td>
<td>28</td>
<td>F</td>
<td>TCTTELUC</td>
<td>TCTEESPL</td>
<td>LU6.2 conversation type (mapped or unmapped)</td>
</tr>
<tr>
<td>41</td>
<td>29</td>
<td>F</td>
<td>DFHTCTTE</td>
<td>TCTEESPA</td>
<td>Additional syncpoint flags</td>
</tr>
<tr>
<td>42</td>
<td>2A</td>
<td>F</td>
<td>TCTTELUC</td>
<td>TCTEIAHB</td>
<td>Attach FMH indicator</td>
</tr>
<tr>
<td>43</td>
<td>2B</td>
<td>F</td>
<td>DFHTCTSE</td>
<td>TCSESID</td>
<td>NETNAME of APPC device</td>
</tr>
<tr>
<td>44</td>
<td>2C</td>
<td>U</td>
<td>DFHSNTTE</td>
<td>SNTENLS</td>
<td>User's national language</td>
</tr>
<tr>
<td>45</td>
<td>2D</td>
<td>F</td>
<td>DFHTCTTE</td>
<td>TCTENLS</td>
<td>National Language Support Code</td>
</tr>
<tr>
<td>46</td>
<td>2E</td>
<td>F</td>
<td>DFHTCTTE</td>
<td>TCTESCFL</td>
<td>Security flag</td>
</tr>
<tr>
<td>47</td>
<td>2F</td>
<td>F</td>
<td>DFHTCTTE</td>
<td>TCTEITRS</td>
<td>Trace flags</td>
</tr>
<tr>
<td>48</td>
<td>30</td>
<td>F</td>
<td>DFHTCTME</td>
<td>TCMEMODE</td>
<td>Mode group name</td>
</tr>
<tr>
<td>49</td>
<td>31</td>
<td>F</td>
<td>DFHTCTTE</td>
<td>TCTENLI</td>
<td>National language in use</td>
</tr>
<tr>
<td>50</td>
<td>32</td>
<td>F</td>
<td>TCTTELUC</td>
<td>TCTELUC1</td>
<td>LUC flag byt e 1</td>
</tr>
<tr>
<td>51</td>
<td>33</td>
<td>F</td>
<td>DFHTCTTE</td>
<td>TCTESSPL</td>
<td>Synclevel of link</td>
</tr>
<tr>
<td>52</td>
<td>34</td>
<td>F</td>
<td>DFHTCTTE</td>
<td>TCTEVTP</td>
<td>Send mode/receive mode</td>
</tr>
<tr>
<td>53</td>
<td>35</td>
<td>F</td>
<td>DFHTCTTE</td>
<td>TCTEIO</td>
<td>Task to be initiated</td>
</tr>
<tr>
<td>54</td>
<td>36</td>
<td>F</td>
<td>DFHTCTTE</td>
<td>TCTEIO</td>
<td>Task to be initiated</td>
</tr>
<tr>
<td>55</td>
<td>37</td>
<td>F</td>
<td>DFHLFS</td>
<td>PRESETC</td>
<td>Preset userid</td>
</tr>
<tr>
<td>56</td>
<td>38</td>
<td>F</td>
<td>TCTTETTE</td>
<td>TCTEFMB</td>
<td>Outbound formatting status</td>
</tr>
<tr>
<td>57</td>
<td>39</td>
<td>F</td>
<td>DFHTCTTE</td>
<td>TCTEUCTB</td>
<td>UCTRAN = YES</td>
</tr>
<tr>
<td>58</td>
<td>3A</td>
<td>F</td>
<td>DFHTCTTE</td>
<td>TCTETSU3</td>
<td>UCTRAN = TRANID</td>
</tr>
<tr>
<td>60</td>
<td>3F</td>
<td>F</td>
<td>DFHTCTTE</td>
<td>TCTETT</td>
<td>Terminal type code</td>
</tr>
<tr>
<td>64</td>
<td>40</td>
<td>F</td>
<td>DFHLUCDS</td>
<td>LUCOPN0</td>
<td>LUC request codes</td>
</tr>
<tr>
<td>65</td>
<td>41</td>
<td>F</td>
<td>DFHLUCDS</td>
<td>LUCRCODE</td>
<td>LUC request error feedback</td>
</tr>
<tr>
<td>66</td>
<td>42</td>
<td>F</td>
<td>DFHLUCDS</td>
<td>LUCSDBLK</td>
<td>LUC conversation feedback</td>
</tr>
<tr>
<td>67</td>
<td>43</td>
<td>F</td>
<td>DFHLUCDS</td>
<td>LUCNSYS</td>
<td>System name for LUC Allocate</td>
</tr>
<tr>
<td>68</td>
<td>44</td>
<td>F</td>
<td>DFHLUCDS</td>
<td>LUCMODNM</td>
<td>Mode name for LUC Allocate</td>
</tr>
</tbody>
</table>
### Transaction routing data stream (continued)
Built by the terminal sharing transformer (DFHXTP).

<table>
<thead>
<tr>
<th>Code</th>
<th>Hex</th>
<th>Type</th>
<th>DSECT</th>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>69</td>
<td>45</td>
<td>F</td>
<td>DFHLUCDS</td>
<td>LUCMSGNO</td>
<td>Message number for LUC Abend and Error</td>
</tr>
<tr>
<td>70</td>
<td>46</td>
<td>F</td>
<td>DFHLUCDS</td>
<td>LUCSENSE</td>
<td>Sense code for LUC Abend and Error</td>
</tr>
<tr>
<td>71</td>
<td>47</td>
<td>F</td>
<td>DFHLUCDS</td>
<td>LURQSEQN</td>
<td>Conversation type for LUC Issue Attach</td>
</tr>
<tr>
<td>72</td>
<td>48</td>
<td>F</td>
<td>DFHLUCDS</td>
<td>LURQSYN</td>
<td>Syncpoint level for LUC Issue Attach</td>
</tr>
<tr>
<td>73</td>
<td>49</td>
<td>F</td>
<td>DFHLUCDS</td>
<td>LUCFTPNL</td>
<td>TPN for LUC Issue Attach</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>LUCFTPN</td>
<td></td>
</tr>
<tr>
<td>74</td>
<td>4A</td>
<td>F</td>
<td>DFHLUCDS</td>
<td>LUCPIP</td>
<td>PIP indicator for LUC Issue Attach</td>
</tr>
<tr>
<td>75</td>
<td>4B</td>
<td>F</td>
<td>DFHLUCDS</td>
<td>LUCTAREL</td>
<td>Maximum receivable length for LUC Receive</td>
</tr>
<tr>
<td>76</td>
<td>4C</td>
<td>F</td>
<td>DFHLUCDS</td>
<td>LUCMGAL</td>
<td>Mode group name of allocated session</td>
</tr>
<tr>
<td>90</td>
<td>5A</td>
<td>F</td>
<td>DFHDIBDS</td>
<td>DIBSENSE</td>
<td>DIB system/user sense data</td>
</tr>
<tr>
<td>128</td>
<td>80</td>
<td>V</td>
<td>XTPCDZIR</td>
<td></td>
<td>ZC install response</td>
</tr>
<tr>
<td>129</td>
<td>81</td>
<td>V</td>
<td>XTPCDZBP</td>
<td></td>
<td>ZC builder parameter set</td>
</tr>
<tr>
<td>130</td>
<td>82</td>
<td>V</td>
<td>XTPCDZIM</td>
<td></td>
<td>ZC install message set</td>
</tr>
<tr>
<td>131</td>
<td>83</td>
<td>V</td>
<td>XTPCOPCL</td>
<td></td>
<td>Opclass in routed message</td>
</tr>
<tr>
<td>132</td>
<td>84</td>
<td>V</td>
<td>XTPCDPNM</td>
<td></td>
<td>Program name for ISSUE LOAD</td>
</tr>
<tr>
<td>133</td>
<td>85</td>
<td>V</td>
<td>XTPPLUCSD</td>
<td></td>
<td>Message text for LUC Send</td>
</tr>
<tr>
<td>134</td>
<td>86</td>
<td>V</td>
<td>XTPLUERD</td>
<td></td>
<td>Message text for LUC Receive</td>
</tr>
<tr>
<td>135</td>
<td>87</td>
<td>V</td>
<td>XTPLUUTCX</td>
<td></td>
<td>TCA extension for LU6.2</td>
</tr>
<tr>
<td>136</td>
<td>88</td>
<td>V</td>
<td>XTPLUMSG</td>
<td></td>
<td>Message text for LUC Issue Abend or Issue Error</td>
</tr>
<tr>
<td>137</td>
<td>89</td>
<td>V</td>
<td>XTPIPASS</td>
<td></td>
<td>Issue Pass</td>
</tr>
<tr>
<td>138</td>
<td>8A</td>
<td>V</td>
<td>XTPLDATA</td>
<td></td>
<td>Logon Data</td>
</tr>
<tr>
<td>139</td>
<td>8B</td>
<td>V</td>
<td>XTPRETC</td>
<td></td>
<td>Issue Pass Return Code</td>
</tr>
<tr>
<td>140</td>
<td>8C</td>
<td>V</td>
<td>XTPLMOD</td>
<td></td>
<td>Issue Pass Logmode</td>
</tr>
</tbody>
</table>

### Control blocks

**Relay transaction control blocks**

To support transaction routing, the relay transaction owns two TCTTEs; see Figure 116. One TCTTE is for the terminal, the other is for the link to the user transaction. The link TCTTE has bit TCTERTL in field TCTETSU set on, to indicate that it is being used by the relay transaction.
Transaction routing

User transaction control blocks

The user transaction owns two or more TCTTEs; see Figure 117 on page 1333. One TCTTE is always present for the link to the relay transaction, and another TCTTE, called the surrogate TCTTE, represents the terminal TCTTE in the relay transaction address space. Field TCTTERLA in the surrogate TCTTE contains the address of the TCTTE for the link to the relay transaction. Bit TCTESUR (in field TCTETSU) set on indicates that the TCTTE is for a surrogate terminal. The link TCTTE has bit TCTERLX in field TCTETSU set on, to indicate that it is being used as a relay link.

If the user transaction executes CICS functions that are shipped to another address space or processing system, one TCTTE is chained off from the TCA for each different address space or processing system.

Figure 116. Control blocks associated with the relay transaction
See the CICS Data Areas manual for a detailed description of these control blocks.

**Modules**

The principal modules associated with transaction routing are as follows:

**DFHAPRT**

is the relay program for non-APPC devices, and for APPC devices when the device initiates a transaction by sending an attach FMH5 to CICS.

**DFHCRT**

is the relay program for APPC devices when CICS sends an attach FMH5 to the device.
## Transaction routing

**DFHRTSU**
- is the program which maintains the state of a surrogate APPC session during syncpoint

**DFHXTP**
- is the data transformation program for terminal sharing. It constructs and interprets data streams flowing between terminal-owning and application-owning regions, for both APPC and non-APPC transaction routing environments.

**DFHZTSP**
- is the terminal sharing program. It is used by transaction routing for devices of all types, exclusively so for non-APPC devices.

**DFHZXRL**
- runs in the application-owning region to route APPC requests to the terminal-owning region.

**DFHZXRT**
- runs in the terminal-owning region to receive APPC requests from the application-owning region, and issue them to the APPC device.

### Exits

No global user exit points are provided for this function.

### Trace

The following point IDs are provided for this function:

- AP DBxx (DFHXTP), for which the trace level is IS 1
- AP 08xx (DFHCRT, DFHZXRL, and DFHZXRT), for which the trace levels are IS 1, IS 2, and Exc.

For more information about the trace points, see the CICS Trace Entries. For more information about using traces in problem determination, see the CICS Problem Determination Guide.
Chapter 103. Transient data control

Transient data control provides an optional queuing facility for managing data being transmitted between user-defined destinations (I/O devices or CICS tasks). This function facilitates data collection.

Design overview

The transient data program provides a generalized queuing facility enabling data to be queued (stored) for subsequent internal or offline processing. Selected units of information can be routed to or from predefined symbolic queues. The queues are classified as either intrapartition or extrapartition.

Intrapartition queues

Intrapartition queues are queues of data, held in a direct-access data set, for eventual input to one or more CICS transactions. Intrapartition queues are accessible only by CICS transactions within the CICS address space. Data directed to or from these internal queues is called intrapartition data. It can consist of variable-length records only.

An intrapartition queue is mapped onto one or more control intervals in the intrapartition data set. The control intervals are allocated to a queue as records are written and freed automatically as they are read or as the queue is deleted.

Examples of the data queued for intrapartition processing are:

- Transactions that require processes to be performed serially, not concurrently. An example of this type of process is one in which pending order numbers are to be assigned.
- Data to be used in a data set (file) update that could pass through the queue to allow the data to be applied in sequence.

Recovery of intrapartition transient data queues

Following abnormal system termination, intrapartition queues defined as recoverable by the user can be restored. Recovery is accomplished by reconstructing the queues from catalog data and from log records written automatically by CICS during normal execution. Two types of recovery are possible: physical and logical.

Physical recovery of intrapartition transient data queues: Physically recoverable transient data queues are restored to the state they were in when the system terminated abnormally. A physically recoverable transient data queue is not backed out if it has been updated by a unit of work (UOW) that has subsequently failed. Data written to such a queue is always committed and is restored during warm and emergency restarts.

When a UOW reads, writes, or deletes a physically recoverable queue, a log record is written to the system log. When the system is brought up after an abnormal termination, CICS can recreate a queue by retrieving definition information associated with the queue from the catalog, and state data from the log.

Note: There is an exception to the rule that states that a physically recoverable queue is restored to the state it was in when CICS abnormally terminated. If
a UOW reads a physically recoverable queue and CICS then terminates abnormally, the read operation will be backed out when CICS is subsequently brought back up.

**Logical recovery of intrapartition transient data queues:** Logically recoverable transient data queues are restored to the state they were in at the time they were last syncpointed. All inflight UOWs are backed out. If a UOW updates a logically recoverable queue and subsequently fails, all updates to the queue are backed out. Logically recoverable queues are restored during warm and emergency restarts.

Logically recoverable queues are logged as part of the first phase of syncpoint processing. When CICS is brought up after an abnormal termination, it can recreate logically recoverable queues by retrieving definition information associated with the queue from the catalog, and state data from the log.

Logically recoverable transient data queues can suffer from indoubt failures. If a UOW is indoubt and CICS abnormally terminates, the indoubt UOW environment is recreated when CICS is next brought up. When the indoubt failure is resolved, the UOW is committed or backed out.

**Extrapartition queues**

Extrapartition queues are sequential data sets on tape or direct-access devices. Data directed to or from these external queues is called extrapartition data and can consist of sequential records that are fixed- or variable-length, blocked or unblocked.

Data can be placed on an extrapartition data set by CICS for subsequent input to CICS or for offline processing. Sequentially organized data created by other than CICS programs can be entered into CICS as an extrapartition data set. Examples of data that might be placed on extrapartition data sets are:
- System statistics
- Transaction error messages
- Customer data, such as cash payments that can be applied offline.

**Indirect queues**

Intrapartition and extrapartition queues can be referenced through indirect destinations. This provides flexibility in program maintenance. Queue definitions can be changed, using the CEDA transaction, without having to recompile existing programs.

**Automatic transaction initiation**

When data is sent to an intrapartition queue and the number of entries (WRITEQs from one or more programs) in the queue reaches a predefined level (trigger level), the user can optionally specify that a transaction be automatically initiated to process the data in that queue.

The automatic transaction initiation (ATI) facility allows a user transaction to be initiated either immediately, or, if a terminal is required, when that terminal has no task associated with it. The terminal processing status must be such that messages can be sent to it automatically. Through the trigger level and automatic transaction initiation facility, an application program can switch messages to terminals. After a task has been initiated, a command in the application program is executed to retrieve the queued data. All data in the queue is retrieved sequentially for the application program.
Transient data control

Trigger transactions may only execute sequentially against their associated queue. When a trigger transaction has been attached, another transaction will not be attached until the first transaction has completed. If a trigger transaction suffers an indoubt failure, (the transaction must be associated with a logically recoverable queue) another trigger transaction cannot be attached until the indoubt failure has been resolved.

Transient data services

The following services are performed by the transient data program in response to transient data commands issued in application programs:

Intrapartition data disposition
Controls and queues data for serially reusable or re-enterable facilities (programs, terminals) related to this partition or region.

Intrapartition data acquisition
Retrieves data that has been placed in a queue for subsequent internal processing.

Extrapartition data acquisition
Enters a sequentially organized data set into the system.

Extrapartition data disposition
Writes fixed- or variable-length data in a blocked or unblocked format on sequential devices, usually for subsequent offline processing.

Automatic transaction initiation
Initiates a transaction to process previously queued transient data when a predefined trigger level is reached.

Dynamic open/close
Logically opens or closes specified extrapartition data sets (queues) during the real-time execution of CICS.

Dynamic allocation and deallocation of extrapartition queues
Extrapartition transient data queues do not have to be predefined in your JCL. They can be created dynamically.

Transient data

This section describes transient data’s interfaces.

Intrapartition queues

Figure 118 shows transient data’s interfaces for intrapartition queues.
Transient data control

Notes:
1. An application program invokes a Transient Data request (WRITEQ TD, READQ TD, or DELETEQ TD). The EXEC interface module, DFHETD is invoked and calls Transient Data using the TDTD CDURUN parameter list.
2. Transient Data locates the target queue using a Directory Manager locate.
3. Assuming that the required queue has been found, the call is passed to the module that handles intrapartition queue requests, DFHTDQ.
4. If the target queue is logically recoverable, Transient Data must tell Recovery Manager it is interested in this UOW by setting its work token in the Recovery Manager’s table.
5. If the target queue is logically recoverable, Transient Data must obtain an enqueue on the appropriate end of the queue by invoking the Enqueue Manager.
6. Data is read from (or written to) the target queue using the appropriate access method. In the case of physically recoverable queues only, the buffers are always flushed and the data set hardened.
7. After the request has completed, Transient Data must log the state of the queue, if the queue is physically recoverable.
8. If the request was a WRITEQ TD request and the target queue was physically recoverable or non-recoverable, the trigger level may have been exceeded. If the trigger transaction is to be associated with a terminal DFHALP is invoked so that the required AID can be scheduled. If the trigger transaction is to be associated with a file, Transaction Manager is invoked to attach the trigger transaction.

Figure 118. Transient data interfaces for intrapartition queues
9. If a UOW has updated a logically recoverable queue, Recovery Manager invokes Transient Data when the UOW begins syncpoint processing DFHTDRM.

10. Transient Data invokes the appropriate access methods to harden the data set. Finally, Recovery Manager invokes Transient Data once more, detailing whether Transient Data should commit or back out its updates.

11. If the UOW commits the updates, Transient Data attaches a trigger transaction or schedules an AID if the trigger level has been exceeded. DFHALP is invoked if the trigger transaction is associated with a terminal. Transaction Manager is invoked if the trigger transaction is associated with a file.

**Extrapartition queues**

Figure 119 shows the transient data interfaces for extrapartition queues.

---

**Notes:**

1. An application program invokes Transient Data services (WRITEQ TD, READQ TD or DELETEQ TD). The EXEC interface module, DFHETD is invoked. DFHETD invokes Transient Data using the TDTD CDURUN parameter list.

2. Transient Data locates the target queue using Directory Manager.

3. The request is passed to the appropriate QSAM routine for processing. QSAM PUT with LOCATE mode is used.

4. If an application program requests that an intrapartition queue be opened or closed, module DFHTDOC is invoked using the TDOC CDURUN parameter list.
**Transient data control**

### Modules

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHTDP</td>
<td>Provides request analysis and extrapartition processing, RMODE(24)</td>
</tr>
<tr>
<td>DFHTDA</td>
<td>Included in load module DFHTDP. Provides request analysis and processing for extrapartition queues</td>
</tr>
<tr>
<td>DFHTDEXC</td>
<td>Included in load module DFHTDP. Contains subroutines associated with the processing of extrapartition queues</td>
</tr>
<tr>
<td>DFHTDOC</td>
<td>Included in load module DFHTDP. Manages the opening and closing of extrapartition queues</td>
</tr>
<tr>
<td>DFHETD</td>
<td>Processes EXEC CICS commands and maps them to the TDTD CDURUN parameter list</td>
</tr>
<tr>
<td>DFHTDB</td>
<td>Included in load module DFHTDQ. Processes intrapartition queue requests</td>
</tr>
<tr>
<td>DFHTDSUC</td>
<td>Included in load module DFHTDQ. Contains subroutines associated with the processing of intrapartition transient data queues</td>
</tr>
<tr>
<td>DFHTDRM</td>
<td>Undertakes syncpoint processing on behalf of Transient Data</td>
</tr>
<tr>
<td>DFHTDTM</td>
<td>Manages requests to install, discard, set and inquire on transient data queues</td>
</tr>
</tbody>
</table>

### Exits

The following global user exit points are provided for this function: XTDREQ, XTDREQC,XDIN, and XTDOUT.

See the *CICS Customization Guide* for further information.

### Trace

The following point ID is provided for transient data control:
- AP F6xx, for which the trace levels are TD 1 and Exc.

For more information about the trace points, see the *CICS Trace Entries*. For more information about using traces in problem determination, see the *CICS Problem Determination Guide*. 
Chapter 104. User domain

The user domain provides an optional facility for checking user authority to sign on to a terminal.

User domain’s specific gates

Table 118 summarizes the user domain’s specific gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and whether or not the functions are available through the exit programming interface (XPI).

Table 118. User domain’s specific gates

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>XPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>USAD</td>
<td>US 0201</td>
<td>ADD_USER_WITH_PASSWORD</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>US 0202</td>
<td>ADD_USER_WITHOUT_PASSWORD</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>US 0203</td>
<td>DELETE_USER</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>US 0204</td>
<td>INQUIRE_USER</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>US 0205</td>
<td>INQUIRE_DEFAULT_USER</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VALIDATE_USER</td>
<td>NO</td>
</tr>
<tr>
<td>USFL</td>
<td>US 0501</td>
<td>FLATTEN_USER</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>US 0502</td>
<td>UNFLATTEN_USER</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>US 0503</td>
<td>TAKEOVER</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>US 0504</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>US 0505</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>US 0509</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>US 050A</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>US 050B</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>US 050C</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>US 050D</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>US 050E</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>US 050F</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>US 0510</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>US 0511</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>US 0512</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>US 0513</td>
<td></td>
<td></td>
</tr>
<tr>
<td>USIS</td>
<td>US 0201</td>
<td>SET_USER_DOMAIN_PARMS</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>US 0202</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>US 0203</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>US 0204</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>US 0205</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>US 0206</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>US 0207</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>US 0208</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>US 0209</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>US 020A</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 118. User domain’s specific gates (continued)

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>XPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>USXM</td>
<td>US 0401</td>
<td>ADD_TRANSACTION_USER</td>
<td>NO</td>
</tr>
<tr>
<td>US</td>
<td>US 0402</td>
<td>DELETE_TRANSACTION_USER</td>
<td>NO</td>
</tr>
<tr>
<td>US</td>
<td>US 0403</td>
<td>END_TRANSACTION</td>
<td>NO</td>
</tr>
<tr>
<td>US</td>
<td>US 0404</td>
<td>FLATTEN_TRANSACTION_USER</td>
<td>NO</td>
</tr>
<tr>
<td>US</td>
<td>US 0405</td>
<td>INIT_TRANSACTION_USER</td>
<td>NO</td>
</tr>
<tr>
<td>US</td>
<td>US 0406</td>
<td>INQUIRE_TRANSACTION_USER</td>
<td>NO</td>
</tr>
<tr>
<td>US</td>
<td>US 0407</td>
<td>TERM_TRANSACTION_USER</td>
<td>NO</td>
</tr>
<tr>
<td>US</td>
<td>US 0408</td>
<td>UNFLATTEN_TRANSACTION_USER</td>
<td></td>
</tr>
<tr>
<td>US</td>
<td>US 0409</td>
<td></td>
<td></td>
</tr>
<tr>
<td>US</td>
<td>US 040B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>US</td>
<td>US 040C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>US</td>
<td>US 040D</td>
<td></td>
<td></td>
</tr>
<tr>
<td>US</td>
<td>US 040E</td>
<td></td>
<td></td>
</tr>
<tr>
<td>US</td>
<td>US 040F</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### USAD gate, ADD_USER_WITH_PASSWORD function

The ADD_USER_WITH_PASSWORD function of the USAD gate is used to add a user to the CICS region and verify the associated password or oidcard.

**Input parameters**

**USERID**

is the identifier of the user (a userid of 1 through 10 alphanumeric characters) to be added to the security domain.

**USERID_LENGTH**

is the length of the USERID value.

**[PASSWORD]**

is the current password, 1 through 10 alphanumeric characters, for the userid specified by the USERID value.

**[PASSWORD_LENGTH]**

is the 8-bit length of the PASSWORD value. This parameter is only valid if PASSWORD is also specified.

**[NEW_PASSWORD]**

is a new password, 1 through 10 alphanumeric characters, to be assigned to the userid (specified by the USERID value). This parameter is only valid if PASSWORD is also specified.

**[NEW_PASSWORD_LENGTH]**

is the 8-bit length of the NEW_PASSWORD value. This parameter is only valid if NEW_PASSWORD is also specified.

**[OIDCARD]**

is an optional oidcard (operator identification card); a 65-byte field containing further security data from a magnetic strip reader (MSR) on 32xx devices.

**[GROUPID]**

is an optional identifier, 1 through 10 alphanumeric characters, of a RACF user group to which the userid (specified by the USERID value) is to be assigned.

**[GROUPID_LENGTH]**

is the 8-bit length of the GROUPID value. This parameter is only valid if GROUPID is also specified.
[ENTRY_PORT_NAME]
is an optional name of an entry port, 1 through 8 alphanumeric characters, to be assigned to the userid (specified by the USERID value).

[ENTRY_PORT_TYPE]
is the type of the optional entry port to be assigned to the userid (specified by the USERID value). It can have either of these values:
TERMINAL|CONSOLE

This parameter is only valid if ENTRY_PORT_NAME is also specified.

[SCOPE_CHECK]
indicates whether or not scope checking is to be performed for this function call. It can have either of these values:
YES|NO

SIGNON_TYPE
is the type of signon for the userid (specified by the USERID value). It can have any of these values:
ATTACH_SIGN_ON|DEFAULT_SIGN_ON|IRC_SIGN_ON|LU61_SIGN_ON|LU62_SIGN_ON|NON_TERMINAL_SIGN_ON|PRESET_SIGN_ON|USER_SIGN_ON|XRF_SIGN_ON

Output parameters

USER_TOKEN
is the token identifying the userid in the user domain.

[SAF_RESPONSE]
is the optional 32-bit SAF response code to the call.

[SAF_REASON]
is the optional 32-bit SAF reason returned with SAF_RESPONSE.

[ESM_RESPONSE]
is the optional 32-bit ESM response code to the call.

[ESM_REASON]
is the optional 32-bit ESM reason returned with ESM_RESPONSE.

RESPONSE
is the domains response to the call. It can have any of these values:
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]
is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>DEL_TIMEOUT_ENTRY_FAILED, EXTRACT_FAILED, GETMAIN_FAILED</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>ALREADY_SIGNED_ON, PASSWORD_REQUIRED, NEW_PASSWORD_REQUIRED, OIDCARD_REQUIRED, INVALID_USERID, INVALID_PASSWORD, INVALID_NEW_PASSWORD, INVALID_OIDCARD, INVALID_GROUPID, INQUIRE_PW_DATA_FAILED, USERID_NOT_IN_GROUP, UNKNOWN_ESM_RESPONSE, SECURITY_INACTIVE, ESM_INACTIVE, ENTRY_PORT_NOTAUTH, APPLICATION_NOTAUTH, USERID_REVOKED, GROUP_ACCESS_REVOKED, SECLABEL_CHECK_FAILED, ESM_TRANQUIL</td>
</tr>
</tbody>
</table>
USAD gate, ADD_USER_WITHOUT_PASSWORD function

The ADD_USER_WITHOUT_PASSWORD function of the USAD gate is used to add a user to the CICS region without verifying any password or oidcard.

Input parameters

USERID
is the identifier of the user (a userid of 1 through 10 alphanumeric characters) to be added to the security domain.

USERID_LENGTH
is the 8-bit length of the USERID value.

[APPLID]
is the application identifier for the CICS region.

[ENTRY_PORT_NAME]
is an optional name of an entry port, 1 through 8 alphanumeric characters, to be assigned to the userid (specified by the USERID value).

[ENTRY_PORT_TYPE]
is the type of the optional entry port to be assigned to the userid (specified by the USERID value). It can have either of these values:

TERMINAL|CONSOLE

This parameter is only valid if ENTRY_PORT_NAME is also specified.

GROUPID
is an optional identifier, 1 through 10 alphanumeric characters, of a RACF user group to which the userid (specified by the USERID value) is to be assigned.

GROUPID_LENGTH
is the 8-bit length of the GROUPID value. This parameter is only valid if GROUPID is also specified.

LOGIN_CONTEXT
is a token identifying the context of the user’s login attempt (for example, whether the user was already logged in).

SCOPE_CHECK
indicates whether or not scope checking is to be performed for this function call. It can have either of these values:

YES|NO

SIGNON_TYPE
is the type of signon for the userid (specified by the USERID value). It can have any of these values:

ATTACH_SIGN_ON|DEFAULT_SIGN_ON|IRC_SIGN_ON|
LU61_SIGN_ON|LU62_SIGN_ON|NON_TERMINAL_SIGN_ON|
PRESET_SIGN_ON|USER_SIGN_ON|XRF_SIGN_ON

[SUSPEND]
indicates whether a wait during add user processing is acceptable. It can have either of these values:

YES|NO
**User domain**

[UUID] is the unique universal ID (UUID) for the user.

**Output parameters**

**USER_TOKEN** is the token identifying the userid in the user domain.

**[SAF_RESPONSE]** is the optional 32-bit SAF response code to the call.

**[SAF_REASON]** is the optional 32-bit SAF reason returned with SAF_RESPONSE.

**[ESM_RESPONSE]** is the optional 32-bit ESM response code to the call.

**[ESM_REASON]** is the optional 32-bit ESM reason returned with ESM_RESPONSE.

**RESPONSE** is the domains response to the call. It can have any of these values:
- OK
- EXCEPTION
- DISASTER
- INVALID
- KERNERROR
- PURGED

**[REASON]** is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>DEL_TIMEOUT_ENTRY_FAILED, EXTRACT_FAILED, GETMAIN_FAILED</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>ALREADY_SIGNED_ON, APPLICATION_NOTAUTH, ENTRY_PORT_NOTAUTH, ESM_INACTIVE, ESM_TRANQUIL, GROUP_ACCESS_REVOKED, INVALID_GROUPID, INVALID_USERID, SECLABEL_CHECK_FAILED, SECURITY_INACTIVE, UNKNOWN_ESM_RESPONSE, USER_NOT_LOCATED, USERID_NOT_IN_GROUP, USERID_REVOKED</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION, INVALID_PARAMETERS</td>
</tr>
</tbody>
</table>

**USAD gate, DELETE_USER function**

The DELETE_USER function of the USAD gate is used to delete the user from the CICS region.

**Input parameters**

**USER_TOKEN** is the token identifying the userid in the user domain.

**SIGNOFF_TYPE** is the type of signoff for the userid identified by the SECURITY_TOKEN value. It can have any of these values:
- ABNORMAL_SIGN_OFF
- ATTACH_SIGN_OFF
- DEFERRED_SIGN_OFF
- DELETE_SIGN_OFF
- LINK_SIGN_OFF
- NON_TERMINAL_SIGN_OFF
- PRESET_SIGN_OFF
- UNFLATTEN_USER_SIGN_OFF
- USER_SIGN_OFF
- XRF_SIGN_OFF

**Output parameters**

**[SAF_RESPONSE]** is the optional 32-bit SAF response code to the call.

**[SAF_REASON]** is the optional 32-bit SAF reason returned with SAF_RESPONSE.
User domain

**ESM_RESPONSE**
- is the optional 32-bit ESM response code to the call.

**ESM_REASON**
- is the optional 32-bit ESM reason returned with ESM_RESPONSE.

**RESPONSE**
- is the domains response to the call. It can have any of these values:
  - OK
  - EXCEPTION
  - DISASTER
  - INVALID
  - KERNERROR
  - PURGED

**REASON**
- is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ADD_TIMEOUT_ENTRY_FAILED, FREEMAIN_FAILED</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>INVALID_USER_TOKEN, DEFAULT_USER_TOKEN, SECURITY_INACTIVE, ESM_TRANQUIL, ESM_INACTIVE, UNKNOWN_ESM_RESPONSE</td>
</tr>
</tbody>
</table>

**USAD gate, INQUIRE_USER function**

The INQUIRE_USER function of the USAD gate is used to inquire about the attributes of the user represented by the user token.

**Input parameters**

**USER_TOKEN**
- is the token identifying the userid to the user domain.

**USERNAME**
- is an optional buffer into which the attributes of the user are placed.

**Output parameters**

**USERID**
- is the identifier of the user (a userid of 1 through 10 alphanumeric characters).

**USERID_LENGTH**
- is the length of the USERID value.

**CURRENT_GROUPID**
- is the identifier, 1 through 10 alphanumeric characters, of the current RACF user group to which the userid (specified by the SECURITY_TOKEN value) is assigned.

**CURRENT_GROUPID_LENGTH**
- is the 8-bit length of the GROUPID value.

**NATIONAL_LANGUAGE**
- is a three-character code identifying the national language for the userid. It can have any of the values in Table 93 on page 1023.

**OPERATOR_CLASSES**
- identifies the operator classes to which the user belongs. This is a 24-bit value, with each bit determining whether or not the user is a member of that class.

**OPERATOR_IDENT**
- is the operator identification code, 1 through 3 alphanumeric characters, for the userid.
**OPERATOR_PRIORITY**
is the operator priority value, in the range 0 through 255 (where 255 is the highest priority), for the userid.

**TIMEOUT**
is the number of minutes, in the range 0 through 60, that must elapse since the user last used the terminal before CICS "times-out" the terminal.

**Notes:**
1. CICS rounds values up to the nearest multiple of 5.
2. A TIMEOUT value of 0 means that the terminal is not timed out.

**XRF_REFLECTABLE**
indicates whether or not you want CICS to sign off the userid following an XRF takeover. It can have either of these values:
   - YES
   - NO

**ACEE_PTR**
is a pointer to the access control environment element, the control block that is generated by an external user (ESM) when the user signs on. If the user is not signed on, the address of the CICS DFLTUSER's ACEE is returned. If an ACEE does not exist, CICS sets the pointer reference to the null value, X'FF000000'.

**SAF_RESPONSE**
is the optional 32-bit SAF response code to the call.

**SAF_REASON**
is the optional 32-bit SAF reason returned with SAF_RESPONSE.

**ESM_RESPONSE**
is the optional 32-bit ESM response code to the call.

**ESM_REASON**
is the optional 32-bit ESM reason returned with ESM_RESPONSE.

**RESPONSE**
is the domains response to the call. It can have any of these values:
   - OK
   - EXCEPTION
   - DISASTER
   - INVALID
   - KERNERROR
   - PURGED

**REASON**
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>INVALID_USER_TOKEN</td>
</tr>
</tbody>
</table>

**USAD gate, INQUIRE_DEFAULT_USER function**
The INQUIRE_DEFAULT_USER function of the USAD gate is used to inquire about the attributes of the default user (specified on the DFLTUSER system initialization parameter).

**Input parameters**

**USERNAME**
is an optional buffer into which the attributes of the default user are placed.
User domain

**Output parameters**

**[USERID]**
- is the identifier of the user (a userid of 1 through 10 alphanumeric characters).

**USERID_LENGTH**
- is the length of the USERID value.

**[CURRENTGROUPID]**
- is the identifier, 1 through 10 alphanumeric characters, of the current RACF user group to which the userid (specified by the SECURITY_TOKEN value) is assigned.

**CURRENTGROUPID_LENGTH**
- is the 8-bit length of the GROUPID value.

**[NATIONAL_LANGUAGE]**
- is a three-character code identifying the national language for the userid. It can have any of the values in Table 93 on page 1023.

**[OPERATOR_CLASSES]**
- identifies the operator classes to which the user belongs. This is a 24-bit value, with each bit determining whether or not the user is a member of that class.

**[OPERATOR_IDENT]**
- is the operator identification code, 1 through 3 alphanumeric characters, for the userid.

**[OPERATOR_PRIORITY]**
- is the operator priority value, in the range 0 through 255 (where 255 is the highest priority), for the userid.

**[TIMEOUT]**
- is the number of minutes, in the range 0 through 60, that must elapse since the user last used the terminal before CICS "times-out" the terminal.

**Notes:**
1. CICS rounds values up to the nearest multiple of 5.
2. A TIMEOUT value of 0 means that the terminal is not timed out.

**[XRF_REFLECTABLE]**
- indicates whether or not you want CICS to sign off the userid following an XRF takeover. It can have either of these values: YES|NO

**[ACEE_PTR]**
- is a pointer to the access control environment element, the control block that is generated by an external user (ESM) when the default user signs on. If an ACEE does not exist, CICS sets the pointer reference to the null value, X'FF000000'.

**[SAF_RESPONSE]**
- is the optional 32-bit SAF response code to the call.

**[SAF_REASON]**
- is the optional 32-bit SAF reason returned with SAF_RESPONSE.

**[ESM_RESPONSE]**
- is the optional 32-bit ESM response code to the call.
USAD gate, VALIDATE_USERID function

The VALIDATE_USERID function of the USAD gate is used to verify that the specified userid is a valid userid.

**Input parameters**

- **[USERID]** is the userid to be validated.
- **[USERID_LENGTH]** is the length of the userid to be validated.

**Output parameters**

- **RESPONSE** is the domains response to the call. It can have any of these values:
  - OK
  - EXCEPTION
  - DISASTER
  - INVALID
  - KERNERROR
  - PURGED

- **[REASON]** is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>SECURITY_INACTIVE, USERID_NOT_DEFINED, USERID_NOT_DETERMINED</td>
</tr>
</tbody>
</table>

USFL gate, FLATTEN_USER function

The FLATTEN_USER function of the USFL gate is used to flatten the user’s security state and place into the FLATTENED_USER buffer provided.

**Input parameters**

- **SECURITY_TOKEN** is the token identifying the userid.
- **FLATTENED_USER** is the buffer into which the flattened security state is placed.

**Output parameters**

- **[SAF_RESPONSE]** is the optional 32-bit SAF response code to the call.
- **[SAF_REASON]** is the optional 32-bit SAF reason returned with SAF_RESPONSE.
- **[ESM_RESPONSE]** is the optional 32-bit ESM response code to the call.
- **[ESM_REASON]** is the optional 32-bit ESM reason returned with ESM_RESPONSE.
- **RESPONSE** is the domains response to the call. It can have any of these values:
  - OK
  - EXCEPTION
  - DISASTER
  - INVALID
  - KERNERROR
  - PURGED
**User domain**

[REASON]

is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID.
Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP, DIR_MANAGER_LOCATE_FAILED, SEC_DOM_FLATTEN_FAILED</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>INVALID_USER_TOKEN, SECURITY_INACTIVE, ESM_INACTIVE, ESM_TRANQUIL, UNKNOWN_ESM_RESPONSE</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION, INVALID_FLATTENED_BUFFER</td>
</tr>
</tbody>
</table>

**USFL gate, TAKEOVER function**

The TAKEOVER function of the USFL gate is used, when an XRF takeover occurs, to obtain the SNSCOPE ENQ resources for those users who could not obtain it during tracking, because the resources were already held by the active region.

**Input parameters**

None.

**Output parameters**

**RESPONSE**  
is the domain's response to the call. It can have any of these values:  
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]

is returned when RESPONSE is DISASTER or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

**USFL gate, UNFLATTEN_USER function**

The UNFLATTEN_USER function of the USFL gate is used to unflatten the user security state data in the FLATTENED_SECURITY buffer, and add the userid to the user domain.

**Input parameters**

**FLATTENED_SECURITY**  
is a buffer containing flattened security state data for a userid.

**Output parameters**

**USER_TOKEN**  
is the token identifying the userid in the user domain.

**RESPONSE**  
is the domain's response to the call. It can have any of these values:  
OK|EXCEPTION|DISASTER|INVALID|KERNERROR|PURGED

[REASON]

is returned when RESPONSE is DISASTER, EXCEPTION or INVALID. Possible values are:
### USIS gate, SET_USER_DOMAIN_PARMS function

At CICS startup, loads information for the user domain from the system initialization table (SIT) into the user state data.

**Input parameters**

**DEFAULT_USERID**
- is the default userid, as 1 through 10 alphanumeric characters.

**SIGNON_SCOPE**
- is the scope for which the default userid can be signed on. It can have any of these values:
  - NONE\|CICS\|MVSIMAGE\|SYSPLEX

**DIRECTORY_TIMEOUT_VALUE**
- is the intersystem refresh delay, in the range 0 through 10080 minutes (up to 7 days), for the default userid.

**Output parameters**

**RESPONSE**
- is the domains response to the call. It can have any of these values:
  - OK\|DISASTER

**[REASON]**
- is returned when RESPONSE is DISASTER. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
</tbody>
</table>

### USXM gate, ADD_TRANSACTION_USER function

The ADD_TRANSACTION_USER function of the USXM gate sets the user characteristics (as security tokens) for a transaction.

**Input parameters**

**[PRINCIPAL_USER_TOKEN]**
- is the optional principal user token representing the characteristics of the principal user of the transaction.

**[SESSION_USER_TOKEN]**
- is the optional session user token representing the characteristics of the session user of the transaction.
### User domain

**[EDF_USER_TOKEN]**

is the optional EDF user token representing the characteristics of the EDF user of the transaction.

### Output parameters

**RESPONSE**

is the domain's response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

**[REASON]**

is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>ALREADY_SIGNED_ON, DUPLICATE_USER, INVALID_USER_TOKEN, NO_INPUT_PARAMETER</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

### USXM gate, DELETE_TRANSACTION_USER function

The DELETE_TRANSACTION_USER function of the USXM gate deletes the user token of the specified token type for the transaction.

### Input parameters

**TOKEN_TYPE**

is the type of user token for the transaction. It can have any of these values:

PRINCIPAL | SESSION | EDF

### Output parameters

**RESPONSE**

is the domain's response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

**[REASON]**

is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>NO_USER_TOKEN</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

### USXM gate, END_TRANSACTION function

The END_TRANSACTION function of the USXM gate deletes all the user token to security token maps for the transaction.

### Input parameters

None.
Output parameters

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON]
is returned when RESPONSE is DISASTER or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, FREEMAIN_FAILED, LOOP</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

USXM gate, FLATTEN_TRANSACTION_USER function

The FLATTEN_TRANSACTION_USER function of the USXM gate creates the contents of a FLAT_TRANSUSER buffer from the principal user of the current transaction.

Input parameters

FLAT_TRANUSER
is the buffer to be created.

Output parameters

RESPONSE
is the domain’s response to the call. It can have any of these values:
OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON]
is returned when RESPONSE is DISASTER or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FLAT_TRANSUSER</td>
</tr>
</tbody>
</table>

USXM gate, INIT_TRANSACTION_USER function

The INIT_TRANSACTION_USER function of the USXM gate initializes the transaction for the user characteristics identified by the PRINCIPAL_USER_TOKEN value.

Input parameters

PRINCIPAL_USER_TOKEN
is the principal user token representing the characteristics of the principal user of the transaction.

Output parameters

USDOM_TRANSACTION_TOKEN
is the user token to be used for reference to user characteristics only. It is treated as the principal user token until the next ADD_TRANSACTION_USER call for the transaction.
User domain

**PRIORITY**
is the priority value, in the range 0 through 255 (where 255 is the highest priority), for the user with the token identified by the `PRINCIPAL_USER_TOKEN` value.

**RESPONSE**
is the domains response to the call. It can have any of these values:
- `OK`
- `EXCEPTION`
- `DISASTER`
- `INVALID`
- `KERNERROR`
- `PURGED`

**[REASON]**
is returned when RESPONSE is DISASTER, EXCEPTION, or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, GETMAIN_FAILED, LOOP</td>
</tr>
<tr>
<td>EXCEPTION</td>
<td>INVALID_USER_TOKEN</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

**USXM gate, INQUIRE_TRANSACTION_USER function**
The `INQUIRE_TRANSACTION_USER` function of the USXM gate inquires about the user characteristics associated with the transaction identified by the `USDOM_TRANSACTION_TOKEN` value.

**Input parameters**

**USDOM_TRANSACTION_TOKEN**
is the user token to be used for reference to user characteristics only.

**Output parameters**

**[USERID]**
is the identifier of the user (a userid of 1 through 10 alphanumeric characters).

**USERID_LENGTH**
is the length of the USERID value.

**[USERNAME]**
is an optional buffer that contains the attributes of the user.

**[CURRENT_GROUPID]**
is the identifier, 1 through 10 alphanumeric characters, of the current RACF user group to which the user is assigned.

**[CURRENT_GROUPID_LENGTH]**
is the 8-bit length of the GROUPID value.

**[NATIONAL_LANGUAGE]**
is a three-character code identifying the national language for the user. It can have any of the values in Table 93 on page 1023.

**[OPERATOR_CLASSES]**
identifies the operator classes to which the user belongs. This is a 24-bit value, with each bit determining whether or not the user is a member of that class.

**[OPERATOR_IDENT]**
is the operator identification code, 1 through 3 alphanumeric characters, for the user.
**User domain**

**[OPERATOR_PRIORITY]**

is the operator priority value, in the range 0 through 255 (where 255 is the highest priority), for the user.

**[TIMEOUT]**

is the number of minutes, in the range 0 through 60, that must elapse since the user last used the terminal before CICS "times-out" the terminal.

**Notes:**

1. CICS rounds values up to the nearest multiple of 5.
2. A TIMEOUT value of 0 means that the terminal is not timed out.

**[XRF_SOFF]**

indicates whether or not you want CICS to sign off the user following an XRF takeover. It can have either of these values:

YES | NO

**[ACEE_PTR]**

is a pointer to the access control environment element, the control block that is generated by an external user (ESM) when the user signs on. If the user is not signed on, the address of the CICS DFLTUSER’s ACEE is returned. If an ACEE does not exist, CICS sets the pointer reference to the null value, X’FF000000’.

**[SAF_RESPONSE]**

is the optional 32-bit SAF response code to the call.

**[SAF_REASON]**

is the optional 32-bit SAF reason returned with SAF_RESPONSE.

**[ESM_RESPONSE]**

is the optional 32-bit ESM response code to the call.

**[ESM_REASON]**

is the optional 32-bit ESM reason returned with ESM_RESPONSE.

**RESPONSE**

is the domains response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

**[REASON]**

is returned when RESPONSE is DISASTER or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND LOOP</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

**USXM gate, TERM_TRANSACTION_USER function**

The TERM_TRANSACTION_USER function of the USXM gate removes the state information created by an INIT_TRANSACTION_USER function.

**Input parameters**

**USDOM_TRANSACTION_TOKEN**

is the token that identifies the state data to be removed.
User domain

**Output parameters**

RESPONSE is the domains response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON] is returned when RESPONSE is DISASTER or INVALID. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, FREEMAIN_FAILED, LOOP</td>
</tr>
<tr>
<td>INVALID</td>
<td>INVALID_FORMAT, INVALID_FUNCTION</td>
</tr>
</tbody>
</table>

**USXM gate, UNFLATTEN.TRANSACTION_USER function**

The UNFLATTEN.TRANSACTION_USER function of the USXM gate adds (by the ADD_USER_WITHOUT_PASSWORD function of the USAD gate) the user defined by the contents of the supplied FLAT_TRANSUSER buffer.

**Input parameters**

**FLAT.TRANUSER** is the buffer containing data that defines the user to be added.

[SUSPEND] indicates whether a wait during add user processing is acceptable. It can have either of these values:

YES | NO

**Output parameters**

**PRINCIPAL.USER_TOKEN** is the token identifying the userid in the user domain.

[SAF_RESPONSE] is the optional 32-bit SAF response code to the call.

[SAF_REASON] is the optional 32-bit SAF reason returned with SAF_RESPONSE.

[ESM_RESPONSE] is the optional 32-bit ESM response code to the call.

[ESM_REASON] is the optional 32-bit ESM reason returned with ESM_RESPONSE.

**RESPONSE** is the domains response to the call. It can have any of these values:

OK | EXCEPTION | DISASTER | INVALID | KERNERROR | PURGED

[REASON] is returned when RESPONSE is DISASTER or EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISASTER</td>
<td>ABEND, LOOP</td>
</tr>
</tbody>
</table>
User domain’s generic gates

Table 119 summarizes the user domain’s generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

Table 119. User domain’s generic gates

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMDM</td>
<td>US 0101</td>
<td>INITIALIZE_DOMAIN</td>
<td>DMDM</td>
</tr>
<tr>
<td></td>
<td>US 0102</td>
<td>QUIESCE_DOMAIN</td>
<td></td>
</tr>
<tr>
<td></td>
<td>US 0103</td>
<td>TERMINATE_DOMAIN</td>
<td></td>
</tr>
<tr>
<td></td>
<td>US 0104</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>US 0105</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>US 0106</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>US 0107</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>US 0108</td>
<td></td>
<td></td>
</tr>
<tr>
<td>STST</td>
<td>US 0601</td>
<td>COLLECT_STATISTICS</td>
<td>STST</td>
</tr>
<tr>
<td></td>
<td>US 0602</td>
<td>COLLECT_RESOURCE_STATS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>US 0603</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>US 0604</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>US 0605</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>US 0606</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>US 0607</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>US 0608</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For descriptions of these functions and their input and output parameters, refer to the sections dealing with the corresponding generic formats:

Functions and parameters

Format DMDM—“Domain manager domain’s generic formats” on page 377.

Format STST—“Statistics domain’s generic format” on page 1087.

In initialization processing, performs internal routines to set up the user domain, and gets the initial user options, as for “USIS gate, SET_USER_DOMAIN_PARMS function” on page 1351.

For a cold start, the user options come from the system initialization parameters; for any other type of start, the information comes from the local catalog, but is then modified by any relevant system initialization parameters.

User domain also issues console messages during initialization to report whether or not security is active.

In quiesce and termination processing, the user domain performs only internal routines.
## User domain

### Modules

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHUSAD</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>ADD_USER_WITH_PASSWORD</td>
</tr>
<tr>
<td></td>
<td>ADD_USER_WITHOUT_PASSWORD</td>
</tr>
<tr>
<td></td>
<td>DELETE_USER</td>
</tr>
<tr>
<td></td>
<td>INQUIRE_USER</td>
</tr>
<tr>
<td></td>
<td>INQUIRE_DEFAULT_USER</td>
</tr>
<tr>
<td>DFHUSDM</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>INITIALIZE_DOMAIN</td>
</tr>
<tr>
<td></td>
<td>QUIESCE_DOMAIN</td>
</tr>
<tr>
<td></td>
<td>TERMINATE_DOMAIN</td>
</tr>
<tr>
<td>DFHUSDUF</td>
<td>US domain offline dump formatting routine</td>
</tr>
<tr>
<td>DFHUSFL</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>FLATTEN_USER</td>
</tr>
<tr>
<td></td>
<td>UNFLATTEN_USER</td>
</tr>
<tr>
<td>DFHUSIS</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>SET_USER_DOMAIN_PARMS</td>
</tr>
<tr>
<td>DFHUSST</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>COLLECT_STATISTICS</td>
</tr>
<tr>
<td></td>
<td>COLLECT_RESOURCE_STATS</td>
</tr>
<tr>
<td>DFHUSTI</td>
<td>Handles user timeout processing.</td>
</tr>
<tr>
<td>DFHUSTRI</td>
<td>Interprets US domain trace entries</td>
</tr>
<tr>
<td>DFHUSXM</td>
<td>Handles the following requests:</td>
</tr>
<tr>
<td></td>
<td>ADD_TRANSACTION_USER</td>
</tr>
<tr>
<td></td>
<td>DELETE_TRANSACTION_USER</td>
</tr>
<tr>
<td></td>
<td>END_TRANSACTION</td>
</tr>
<tr>
<td></td>
<td>INIT_TRANSACTION_USER</td>
</tr>
<tr>
<td></td>
<td>INQUIRE_TRANSACTION_USER</td>
</tr>
</tbody>
</table>

### Exits

No global user exit points are provided in this domain.

### Trace

The point IDs for the user domain are of the form US xxxx; the corresponding trace levels are US 1 and Exc.

For more information about the trace points, see the [CICS Trace Entries](#). For more information about using traces in problem determination, see the [CICS Problem Determination Guide](#).
Chapter 105. User exit control

User exit control enables the user to run exit programs at selected points in CICS modules in the application domain and in other domains. The exit program can be enabled or disabled dynamically, and useful information can be transferred to a user work area.

This function:
- Controls which exit programs are to run at which exit points. This is generally specified using EXEC CICS commands and can be changed during a CICS run.
- Invokes the specified exit programs when control reaches an exit point in a CICS module, and handles any change in flow indicated by a return code from the user exit program.

Design overview

User exit control provides an interface that allows the user to run exit programs at selected points (known as exit points) in CICS control modules. The exit programs are separate from the control modules and are associated with them dynamically by means of the EXEC CICS ENABLE command. (See the CICS Customization Guide for a description of how to use exit programs.)

An exit point can have more than one exit program, and an exit program can be shared by more than one exit point. Work areas can be set up for the exit programs, and several exit programs can share a work area. For some exit points, the continuation of the control module can be controlled by a return code.

Each exit point is identified internally by an exit number. The user exit table (UET) contains a UET header and an entry for each exit point, in exit-number order. The UET is addressed from CSAUETBA in the CSA and exists throughout the life of CICS.

Each enabled exit program is represented by an exit program block (EPB). This exists only while an exit program is enabled or while any other exit program is using the work area owned by this exit program. The EPBs are chained together in order of enablement. The UET header points to the first EPB.

Each activation of an exit program for a particular exit point is represented by an exit program link (EPL) which points to the EPB for the exit program. The first EPL for each exit point is contained in the UET entry. If an exit point has more than one exit program, additional EPLs are obtained to represent each subsequent activation. These additional EPLs are chained off the UET entry in order of activation. Thus, for each exit, its EPL chain defines the exit programs that are to be executed at that exit point, and the order of execution.

The user exit interface (UEI) control blocks are illustrated in Figure 120.
All user exit programs are executed in the AP domain. When exit programs are activated for exit points in other domains, control is passed from the domain to the User exit table.
User exit control

AP domain’s user exit service module, which creates the necessary environment to invoke the exit programs via the user exit subroutine.

User exit control modules

This section describes the function of the user exit control modules.

**DFHUEM (user exit manager)**

The user exit manager (DFHUEM) processes EXEC commands that are entered by an application program or the command interpreter to control user exit activity. DFHUEM contains three routines, corresponding to the three commands, as follows:

**ENABLE**

Checks whether an EPB already exists for the exit program specified in the PROGRAM operand.

- If an EPB is not found and the ENTRY operand is not specified, the exit program is loaded, and:
  1. A new EPB is obtained and added to the chain.
  2. The name and entry address of the exit program are placed in the EPB.
  3. If the GALENGTH operand is specified, a work area is obtained, and its address and length are placed in the EPB.
  4. If the GAPROGRAM operand is specified, the address of the EPB for the exit program specified in the GAPROGRAM operand is placed in the new EPB, thus allowing exit programs to share a global work area.
- If the EXIT operand is specified, the EPL chain for the specified exit point is found.
  1. A new EPL is obtained, if necessary, and added to the chain.
  2. The address of the EPB for the exit program specified in the PROGRAM operand is placed in the EPB.
  3. The activation count in the EPB is increased by 1.
  4. If the exit point is not in the AP domain, the domain is notified that the exit point is active.
- If the START operand is specified, the start flag in the EPB is set on.

**DISABLE**

Finds the EPB for the exit program specified in the PROGRAM operand.

- If the STOP or EXITALL operand is specified, the start-flag in the EPB is set off.
- If the EXIT operand is specified, the EPL chain for the specified exit point is found. The EPL pointing to the EPB for the exit program specified in the PROGRAM operand is removed from the chain and the activation count is reduced by 1.
- If the EXITALL operand is specified:
  1. All EPL chains are scanned.
  2. All EPLs pointing to the EPB for the exit program specified in the PROGRAM operand are removed from its chain.
  3. If the ENTRY operand was not specified when the exit program was enabled, the exit program is deleted.
  4. The EPB is removed from the chain.
User exit control

5. If a work area used by the exit program is not still being used by another exit program, it is released.

6. Any EPB or EPL that is no longer required is moved to a free-chain anchored in the UETH.
   • When EXIT or EXITALL is specified for exit points not in the AP domain, the domain is notified when there are no exit programs active.

EXTRACT-EXIT
Finds the EPB for the exit program specified in the PROGRAM operand. The work area’s address and length are extracted from this EPB (or from the EPB that owns the work area) and placed in the user’s fields specified in the GASET and GALENGTH operands.

DFHUEH (user exit handler)
The user exit handler module, DFHUEH, is used to process exit points in the AP domain.

At each exit in a control module, there is a branch to the DFHUEH program. This module scans the EPL chain for that exit and invokes each started exit program in the chain, passing it a parameter list and a register save area. On return from each exit program, the return code is checked and a current return code (maintained by DFHUEH for return to the control module) is set as appropriate.

DFHAPLEX (user exit service module)
The user exit service module, DFHAPLEX, is used to process exit points in domains other than the AP domain.

When an exit point is reached in a non-AP domain, control is passed to the user exit service module (DFHAPLEX) in the AP domain, if the domain has previously been notified that there is an exit program activated for the exit point.

The user exit service module constructs the user exit parameter list, using special parameters from the domain, and invokes the user exit subroutine (DFHSUEX).

The return code from DFHSUEX is passed back to the calling domain.

DFHSUEX (user exit subroutine)
The DFHSUEX module invokes all started user exit programs for an exit point in a domain (other than the AP domain) by scanning the EPL chain, using the same processing as the user exit handler (DFHUEH). The parameter list defined by DFHAPLEX is passed to the exit programs. Return codes from the exit programs are checked and returned to DFHAPLEX.

Control blocks
The control blocks associated with the user exit interface are illustrated in Figure 121 on page 1363 and listed below. Further information about the control blocks is given in the “Design overview” on page 1359 and in Figure 120 on page 1360.
The main control blocks are as follows:

**UETH**  User exit table header

**UETE**  User exit table entry—one for every exit point

**EPB**  Exit program block—one for every enabled user exit program, containing information about the location and activity of the program, and any global work area owned or shared by the program

**EPL**  Exit program link—each EPL indicates one exit program to be invoked at

---

**Note:**
Most of the linkages shown are created dynamically in response to ENABLE commands.

---

**Figure 121. Control blocks associated with the user exit interface**
User exit control

an exit point and which EPL, if any, contains information about the next program to be invoked at that exit point.

See the [CICS Data Areas](#) manual for a detailed description of these control blocks.

## Modules

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHAPEX</td>
<td>The interface between an exit point in a domain (other than the AP domain) and the AP domain.</td>
</tr>
<tr>
<td>DFHSUEX</td>
<td>Handles the invocation of user exit programs at exit points in CICS domains (other than the AP domain). Processing is similar to DFHUEH, passing a parameter list defined in DFHAPEX.</td>
</tr>
<tr>
<td>DFHUEH</td>
<td>Links an exit point in a CICS management module in the AP domain and the user code. DFHUEH invokes in turn each started exit program for that exit point, passing a parameter list defined in the CICS management module.</td>
</tr>
<tr>
<td>DFHUEM</td>
<td>The EXEC interface processor for the ENABLE, DISABLE, and EXTRACT user exit commands.</td>
</tr>
</tbody>
</table>

## Exits

No global user exit points are provided for this function.

## Trace

The following point IDs are provided for this function:

- AP D5xx, for which the trace levels are UE 1, AP 1, AP 2, and Exc.

For more information about the trace points, see the [CICS Trace Entries](#). For more information about using traces in problem determination, see the [CICS Problem Determination Guide](#).

For user exit programs running at an exit point within the AP domain, UE level-1 trace entries are produced.

For user exit programs running at an exit point in a CICS domain other than the AP domain, the UE level-1 trace entries are not produced. Instead, the D5xx trace entries for AP level 1 and AP level 2 are available, providing more information than the UE trace. For AP level 1, the DFHUEPAR parameter list is traced, containing the addresses of fields special to that exit point. For AP level-2 tracing, the contents of the fields are printed, each field being truncated to 200 bytes if necessary.
Chapter 106. VTAM generic resource

This section describes how the generic resource support provided by release 4.2 of VTAM is used by CICS.

A CICS system may register as a VTAM generic resource. It may then be known either by its unique applid or by the generic resource name which is shared by a number of CICS systems, all of which are registered to the same generic resource.

For more information about CICS support for VTAM generic resource consult the CICS Release Guide or the CICS Intercommunication Guide. Consult the VTAM Programming Manual for information about generic resource from the VTAM point of view.

Design Overview

If CICS is to register as a generic resource member, the GRNAME system initialization parameter must be specified.

If GRNAME is specified CICS attempts to register immediately after the ACB is open by issuing the VTAM SETLOGON OPTCD=GNAMEADD command.

If registration succeeds, CICS is then a member of the generic resource specified by the SIT GRNAME parameter and may be addressed either by its generic resource name or (subject to certain restrictions) by its unique applid. Use of the generic resource name allows VTAM to balance the workload by selecting whichever generic resource member is most lightly loaded.

If registration fails, CICS initialization continues but CICS will not be a generic resource member.

The registration status may be examined by means of the CEMT INQUIRE VTAM command.

CICS de-registers as a generic resource by means of the VTAM SETLOGON OPTCD=GNAMEDEL command immediately before the ACB is closed.

Generic resource and LU6.1/LU6.2

Although terminals may log on freely using either the generic resource name or the member name this is not the case with LU6.1 and LU6.2 connections which are more restricted in their use of member names.

LU6.2 GR to GR connections

For LU6.2 connections between generic resources the design makes use of LU6.2 autoinstall. Only connections which are intended to issue an ACQUIRE need be defined and these must all have the generic resource name specified as the NETNAME.

Two types of connection are possible.
VTAM generic resource

Generic resource name connections. These are connections which have the
generic resource name as the NETNAME. NETNAMEs must be unique and so
there can only be one of these per partner generic resource.

Member name connections. These are connections which have the unique
applid (member name) as the NETNAME.

Since there can only be one generic resource name connection for each partner
generic resource it follows that most connections will be member name
connections.

EXEC CICS INQUIRE CONNECTION or CEMT INQUIRE CONNECTION may be
used to determine which is the generic resource name and which the member
name.

When the first BIND from a different generic resource comes into the SCIP exit
(DFHZBLX), a generic resource name connection will be established. If no
predefined generic resource name connection exists one will be autoinstalled.
Subsequent BINDs coming into DFHZBLX from different members of the same
generic resource will cause member name connections to be autoinstalled. A
member name connection should never be defined for a member of a different
generic resource because this creates the possibility of having two definitions
(TCSE’s) for the same connected system.

Communications between members of the same generic resource must be by
member names only.

Two new bits TCSE_GR and TCSE_GRNAME_CONN have been introduced to
indicate the different connection types. They are only valid for LU6.2 connections
between generic resources.

The table shows different values of TCTENNAM, TCSESID and TCSEX62N for
LU6.2 connections between generic resources, depending on the settings of
TCSE_GR and TCSE_GRNAME_CONN.

<table>
<thead>
<tr>
<th>TCSE_GR</th>
<th>TCSE_GRNAME_CONN</th>
<th>ON</th>
<th>ON</th>
<th>OFF</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCTENNAM</td>
<td>GRTname</td>
<td>GRTname</td>
<td>Grname</td>
<td></td>
</tr>
<tr>
<td>TCSESID</td>
<td>Grname</td>
<td>Grname</td>
<td>membername</td>
<td></td>
</tr>
<tr>
<td>TCSEX62N</td>
<td>membername</td>
<td>membername</td>
<td>GRTname</td>
<td></td>
</tr>
</tbody>
</table>

LU6.2 GR to non-GR connections

If a single (non-generic resource) system has an LU6.2 connection to a generic
resource member it may use either the generic resource name or the member name
as the NETNAME.

If the member name is used the initial acquire of the connection must be done by
the non-generic resource partner. This means that the generic resource side must
not have autoconnect set on. This is because the generic resource partner relies on
VTAM to tell it if it is to known by its member name. VTAM does this by setting a
bit which is valid for the first BIND only. Sessions can be acquired by either
partner once the SNASVCMG sessions have bound.

For these connections TCSE_GR is always set off and TCSE_GRNAME_CONN has
no meaning on both systems. The rule here is that TCSESID always contains the
NETNAME (as defined in the RDO connection definition) and TCSEX62N always contains the member name (unique applid). The table illustrates this.

<table>
<thead>
<tr>
<th>TCSE_GR</th>
<th>OFF</th>
<th>OFF</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCSE_GRNAME_CONN</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>RDO_HOSTNAME</td>
<td>GRname</td>
<td>membername</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------</td>
<td>------------</td>
</tr>
<tr>
<td>TCTENNAM</td>
<td>GRname</td>
<td>membername</td>
</tr>
<tr>
<td>TCSESID</td>
<td>GRname</td>
<td>membername</td>
</tr>
<tr>
<td>TCSEX62N</td>
<td>membername</td>
<td>membername</td>
</tr>
</tbody>
</table>

If the generic resource name is to be used, the single system may itself be made into a generic resource allowing it to exploit the design for communications between generic resources. If this is not possible the solution is to use a “hub” or code a generic resource resolution exit to ensure that not more than one member of a generic resource communicates with the single system at any one time using the generic resource name. (The use of “hubs” is described in the CICS Intercommunications Guide).

**LU6.1**

There is no autoinstall for LU6.1, and so less flexibility is allowed for LU6.1 connections between generic resources. CICS-CICS LU6.1 connections can only communicate by generic resource names and must use a “hub” or a generic resource resolution exit.

TCSE_GR and TCSE_GRNAME_CONN do not apply to LU6.1. For LU6.1 connections with a generic resource the generic resource name is in TCTENNAM and TCSESID and the member name is in TCSEX61N.

**Ending affinities**

Affinities are records held by VTAM to show it where to direct data flows within a generic resource. Some of these affinities are “owned” by CICS. These are affinities for LU6.2 synclevel 2, LU6.2 limited resources and LU6.1 connections. They may be ended by means of the SET CONNECTION ENDAFFINITY and PERFORM ENDAFFINITY commands.

**Generic resource and ATI**

This section applies only to those terminals which are logged on using the generic resource name.

When an ATI request is issued in an AOR for a terminal that is logged on to a TOR, CICS uses the terminal definition in the AOR to determine the identity of the TOR to which the request should be shipped. If there is no terminal definition in the AOR, the “terminal-not-known” global user exits (XICTENF and XALTENF) may be used to supply the name of the TOR.

However, if the TOR in question is a member of a generic resource and the user has logged on using the generic resource name, VTAM will have connected the terminal to the generic resource member which was most lightly loaded at the time. If the user then logs off and on again the terminal may be connected to a different generic resource member. If this happens, the TOR which is to receive the ATI request cannot be determined from the terminal definition in the AOR or the “terminal-not-known” user exit.
VTAM generic resource

CICS solves the problem in the following manner:

1. The ATI request is first shipped to the TOR specified in the terminal definition in the AOR (or by the “terminal-not-known” exit). If the terminal is logged on to this TOR (the “first-choice” TOR) the ATI request completes as normal.

2. If the terminal is not logged on to the first-choice TOR, the TOR issues a VTAM INQUIRE OPTCODE=SESSNAME to find which generic resource member, if any, the terminal is now logged on to. This information is passed back to the AOR and the request is then shipped to the correct TOR.

3. If the first-choice TOR is not available, the AOR issues a VTAM INQUIRE OPTCODE=SESSNAME to find where the terminal is now logged on. The INQUIRE is not attempted in the following situations:
   - The VTAM in the AOR is a pre-4.2 version and does not support generic resource.
   - The AOR was started with the VTAM system initialization parameter set to NO.

The INQUIRE will not succeed if the TORs and the AOR are in different networks.

If the INQUIRE is successful the ATL request is shipped to the TOR where the terminal is logged on.

Modules

DFHZBLX

DFHZBLX is a new module which has been created to deal with LU6.2 BIND processing. Part of its function was formerly part of DFHZSCX. It is link-edited with DFHZSCX and is still logically part of it, but it returns directly to VTAM, not via DFHZSCX.

There is a new part of the module, apart from that which was once contained in DFHZSCX, which deals with generic resource BIND processing. If CICS is registered as a generic resource and the partner is also a generic resource, DFHZBLX has to decide on the appropriate type of connection. This may be either a generic resource name connection, in which the NETNAME is the partner’s generic resource name, or a member name connection, in which the NETNAME is the partner’s member name.

DFHZBLX is also responsible for setting the bits in the connection entry which are specific to generic resource.

If CICS is not registered as a generic resource, the generic resource code is not invoked.

DFHZGCH

DFHZGCH is a domain subroutine which is called by DFHEIQSC after one of the following commands:

EXEC CICS SET CONNECTION ENDAFFINITY
CEMT SET CONNECTION ENDAFFINITY
EXEC CICS PERFORM ENDAFFINITY
CEMT PERFORM ENDAFFINITY

1368 CICS Transaction Server: CICS Diagnosis Reference
Its function is to issue the VTAM CHANGE OPTCD=ENDAFFINITY command.

If the affinity is ended successfully,
the connection is deleted if it is autoinstalled.
If the connection is defined,
the generic resource specific information in the connection entry is reset,
the catalog entry is updated,
the connection is deleted from the TCSM index.

The VTAM return codes are reflected back to DFHEIQSC.

DFHZGIN

DFHZGIN is a domain subroutine.

In a TOR it is called by DFHCRS when a request has been shipped from a remote system, if a terminal cannot be located.

In an AOR it is called by DFHALP when the schedule of an AID fails because the TOR has gone away.

It has two functions:

1. **INQUIRE_NQN**
   A VTAM INQUIRE OPTCD=NQN is issued to find the fully qualified NETNAME of a terminal given the NETNAME as input. The fully qualified NETNAME is required for INQUIRE OPTCD=SESSNAME.

2. **INQUIRE_SESSNAME**
   A VTAM INQUIRE OPTCD=SESSNAME is issued to find which member of a generic resource a terminal is logged on to given a fully qualified NETNAME as input.

The following responses are returned to the caller:

- **OK** - VTAM return code was X'00' fdb2 X'00'
- **NOT FOUND** - VTAM return code X'14' fdb2 X'88'
- **EXCEPTION** - The call was rejected for some other reason than not found.

For the exception case an exception trace is written and a message in the range DFHZC0182 - DFHZC0185 is output to the CSNE log giving the VTAM return codes.

**Problem solving for generic resource**

Trace TC level 1, 2 & exception in the ranges AP FA50-FA59, FAB0-FABA and FB87-FB8F.

Messages DFHZC0170 to DFHZC0185 are written to the console and CSNE logs.

Information output by DFHZNAC following BIND failures.

If a dump is produced examine the generic resource status and generic resource flag bytes.
The following symptoms may indicate that an affinity should be ended and has not been.

- Sessions failing to acquire with message DFHZC2405 "Node not activated". This may also indicate a setup error.
- Sessions failing to acquire with various instances of DFHZC2411. This may also indicate that a rule has been violated.
- CICS fails to register as a generic resource when it has previously been a member of a different generic resource. Message DFHZC0171 is written to the console with VTAM rntcd X'14' fdb2 X'86'.
- Connections autoinstalling unexpectedly. If a non-generic resource is addressing a generic resource member by its member name this may also indicate that the first ACQUIRE was issued from the generic resource side.

**Generic resource status byte (TCTV_GRSTATUS)**

**TCTV_GR_REGD (X'80')**
This CICS is registered as a member of a generic resource.

**TCTV_GR_REGERR (X'40')**
This CICS attempted to register as a generic resource member (SIT GRNAME parameter specified) but the attempt was rejected by VTAM.

**TCTV_GR_NOTAVAIL (X'20')**
This CICS attempted to register as a generic resource member (SIT GRNAME parameter specified) but the level of VTAM was not 4.2 or above.

**TCTV_GR_DREGD (X'08')**
This CICS was previously a member of a generic resource but has successfully de-registered.

**TCTV_GR_DREGERR (X'04')**
This CICS attempted to de-register as a member of a generic resource by issuing SETLOGON OPTCD=GNAMEDEL but the attempt was rejected by VTAM.

**TCTV_GR_NOTAPPL (X'02')**
The GRNAME system initialization parameter was not specified.

**TCTV_GR_NOTREG (X'00')**
CICS is not registered as a generic resource and has not attempted to register. (Holds this value before registration is attempted, if required.)

**Generic resource flag byte (TCSEI_GR)**

**TCSE_GR (X'80')**
Both partners are registered as generic resources. Valid from initial acquire to ENDAFFINITY.

**TCSE_GR_NAME_CONN (X'40')**
Set on for a generic resource name connection in which TCSESID contains the generic resource name and TCSEX62N contains the member name.

Set off for a member name connection in which TCSESID contains the member name and TCSEX62N contains the generic resource name.

This bit is only meaningful if TCSE_GR is set on.
VTAM generic resource

TCSE_USE_OUR_MEMBER_NAME (X'20')
The partner is using our member name. (An indication that the member
name, not the generic resource name must be passed in the BIND).

TCSE_MSG179_ISSUED (X'10')
Message DFHZC0179 has been issued. This message is issued when the
secondary SNASVCMG session binds if TCSE_GR is set. It makes clear
which is the generic resource name and which the member name of the
partner session.

TCSE_CATLG_DONE (X'08')
A defined connection with an affinity has been catalogued.

TCSE_MSG177_ISSUED (X'04')
Message DFHZC0177 has been issued. This message is output whenever
an LU6.2 limited resources, LU6.2 synlevel 2 or LU6.1 connection is
acquired. It is output when the secondary SNASVCMG session binds. It is
intended to alert the user to the fact that acquiring the connection has
caused an affinity to be created and gives the NETNAME and NETID of
the partner.

Trace
Trace point ids
• FA50 - FA59
  are provided for problem determination during ENDAFFINITY processing.
  (Module DFHZGCH)
• FAB0 - FABA
  are provided for problem determination during INQUIRE SESSNAME
  processing. (Module DFHZGIN)
• FB87 - FB8F
  are provided for problem determination during generic resource registration and
de-registration. (Module DFHZGSL)

Waits

<table>
<thead>
<tr>
<th>Module</th>
<th>Type</th>
<th>Resource Name</th>
<th>Resource Type</th>
<th>ECB</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHZGCH</td>
<td>MVS</td>
<td>CHANGECB</td>
<td>ZC_ZGCH</td>
<td>CHANGECB</td>
<td>Wait for completion of INQUIRE SESSNAME</td>
</tr>
<tr>
<td>DFHZGIN</td>
<td>MVS</td>
<td>INQ_ECB</td>
<td>ZC_ZGIN</td>
<td>INQ_ECB</td>
<td>Wait for ENDAFFINITY to complete</td>
</tr>
</tbody>
</table>
VTAM generic resource
Chapter 107. VTAM LU6.2

This section describes the layer of CICS that manages the interface to VTAM for LU6.2 communication. VTAM LU6.2 provides advanced program-to-program communication (APPC) between transaction-processing systems, and enables device-level products (APPC terminals) to communicate with host-level products and with each other. APPC sessions can therefore be used for CICS-to-CICS communication, and for communication between CICS and other APPC systems (for example, AS/400) or terminals.

For information about the CICS functions that you can use to exploit LU6.2 communication, see "Chapter 24. Distributed program link" on page 327, "Chapter 25. Distributed transaction processing" on page 331, "Chapter 42. Function shipping" on page 673, "Chapter 46. Intersystem communication (ISC)" on page 721, "Chapter 102. Transaction routing" on page 1311.

Design overview

The main feature that distinguishes LU6.2 from other LU types is the support for parallel sessions i.e. many sessions (and conversations) between the two LUs at the same time. These sessions are further grouped by use of the class of service facility in VTAM. The TCT structure for LU6.2 reflects this. Under the system entry (TCTSE) are a series of mode group entries (TCTMEs). Within a mode group there are a number of sessions represented by terminal entries (TCTTEs).

All the sessions within a mode group have the same transmission characteristics, that is, the same class of service. When a request to ALLOCATE a session is made, a MODENAME can be specified, indicating which class of service is required.

When a session has been allocated and a conversation started, data can be received and sent between the connected LUs. This is more or less directly under the control of the CICS application in the case of DTP, or indirectly under the control of the user for the other ISC facilities.

CICS also supports LU6.2 single session connections. These are represented by a TCTSE, a single TCTME and a single TCTTE. They support the same functions as parallel session connections.

Detailed information about VTAM LU6.2 commands and macros is given in the relevant VTAM manuals.

Session management

Systems Network Architecture (SNA) defines several processes to be used in managing LU6.2 sessions. The CICS implementation provides transaction code for the following Transaction Program Names (TPNs) defined by LU6.2.

- X'06F1' = CHANGE_NUMBER_OF_SESSIONS (CNOS)
- X'06F2' = EXCHANGE_LOG_NAME (XLN)
The required transaction definitions are:

<table>
<thead>
<tr>
<th>TRANSACTION</th>
<th>XTRANID</th>
<th>PROGRAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLS1</td>
<td>X'06F10000'</td>
<td>DFHZLS1</td>
</tr>
<tr>
<td>CLS2</td>
<td>X'06F20000'</td>
<td>DFHCLS3</td>
</tr>
</tbody>
</table>

These resource definitions are provided in the DFHISC group.

So that the SNA service transaction programs can always communicate with each other, even when all the sessions between two systems are busy, two extra sessions are always created whenever parallel sessions exist between two systems. CICS generates these two extra sessions (with a reserved MODENAME of SNASVCMG) unless SINGLESESS(YES) is specified for the connection. Only SNA service transaction programs are allowed to use these two sessions.

**Change Number Of Sessions (CNOS)**

When there are parallel sessions between two LU6.2 systems, it is possible to vary the number of sessions available using CEMT or EXEC CICS commands, either for the entire connection, or by modegroup. The number of available sessions for a modegroup is called the SESSION LIMIT. It corresponds to the number of in-service sessions in that modegroup. The two systems must agree on the session limit for a modegroup at any given time. To achieve this, the LU6.2 architecture defines a CNOS service transaction program which runs in each system, communicating with its counterpart using architected CNOS commands and replies. They negotiate the session limit and the numbers of contention winners and losers at each end. For CICS, the CNOS service transaction program is DFHZLS1.

CNOS commands are not required for the SNASVCMG modegroup on parallel session connections, or for single session connections, because the session limits are fixed.

*Figure 122 on page 1373* shows the flow of control for CNOS operations.
Exchange Log Name (XLN)
When DFHZLNAC determines that it is necessary to exchange log names with a remote system, it starts the syncpoint resynchronization transaction, using the DFHCRERI macro specifying FUNCTION(XLN). The main program for this transaction is DFHCRRSY (in load module DFHLUP). When DFHCRRSY determines that resynchronization is required it will schedule other instances of itself to perform the resynchronization.

When TPN X'06F2' is received from a remote system, DFHCRRSY is called to handle the inbound Exchange Log Names and resynchronization.
LU6.2 session states

The following CICS modules maintain specific states of LU6.2 sessions.

<table>
<thead>
<tr>
<th>Module</th>
<th>State</th>
<th>Macro</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHZBKT</td>
<td>SNA bracket state</td>
<td>DFHZBSM</td>
</tr>
<tr>
<td>DFHZCNT</td>
<td>Contention state</td>
<td>DFHZCNM</td>
</tr>
<tr>
<td>DFHZCHS</td>
<td>Chain state</td>
<td>DFHZCHM</td>
</tr>
<tr>
<td>DFHZCRT</td>
<td>RPL_B state</td>
<td>DFHZCRM</td>
</tr>
</tbody>
</table>

These modules are invoked via the macros shown in the last column. Any query or change to the states is performed using these macros.

The LU6.2 states for each session are stored in the TCTTE for that session. The modules and associated TCTTE field are usually referred to as state machines. When a module, such as DFHZARL, wants to check that the session is in a suitable state to perform a given operation, it uses the appropriate state machine to perform the check by invoking the CHECK function of the relevant macro. If the operation subsequently causes a change in the state of the session, the SET function of the relevant macro is invoked to record the new state.

LU6.2 SEND and RECEIVE processing

LU6.2 SEND processing is done by DFHZSDL, using POST=SCHED to drive the VTAM exit DFHZSLX asynchronously when the request has been passed to VTAM.

DFHZRVL does LU6.2 RECEIVE processing, issuing the request to VTAM for asynchronous processing which drives the VTAM exit DFHZRLX on completion. DFHZRLX queues completed RPLs for further processing by DFHZRLP to a chain anchored off TCTVRPLQ in the TCT prefix. Entries are removed from the queue by DFHZDSP, and passed to the program designated to process the completed RPL. When authorized path VTAM support is used, the SEND and RECEIVE requests use the CICS high performance option (HPO) routines.

SEND and RECEIVE processing for LU6.2 use different RPLs:

- RECEIVE uses the receive RPL (also known as RPL_B, and addressed by TCTERPLB in the TCTTE LUC extension).
- SEND uses the send RPL (addressed by TCTERPLA in the TCTTE).

There are two exceptions when a SEND uses the receive RPL instead of the send RPL:

1. DFHZSDL sending a response
2. DFHZRLP sending DR1 response via synchronous SEND.

The processing state of the receive RPL is maintained in the LU6.2 RPL_B state machine field (TCTERPBS in the TCTTE LUC extension) by the DFHZCRT module and DFHZCRM macro combination, thus allowing rapid identification of the stage and type of RECEIVE being processed.

LU6.2 state machine transitions for contention, bracket, and chain states are performed via the DFHZCNM, DFHZBSM, and DFHZCHM macros as part of SEND and RECEIVE processing for LU6.2 sessions.
Limited resources

For efficient use of some network resources (for example, switched lines), SNA allows for such resources to be defined in the network as limited resources. Whenever a session is bound, VTAM indicates to CICS whether the bind is over a limited resource. Both single and parallel sessions may use limited resources.

The limited resources (LR) function is part of the LU6.2 base option set. When communicating over switched lines, it may be important to stop using this expensive resource as soon as possible. LR provides this facility. A bit in the BIND image is copied into the TCTTE to indicate LR usage. This bit (TCTE_LR) is used to determine whether CICS should UNBIND the link when the TCTTE is freed and no outstanding tasks are using the link.

SNASVCMG (parallel) sessions are not scheduled to be unbound until the initial CNOS exchange has been performed for all mode groups in the connection. They are then treated in the same way as user sessions.

Two bits in the terminal control table are used to reflect LR: TCTE_LR in the terminal entry (TCTTE) and TCSE_LR in the system entry (TCTSE). The following table shows the meanings of the TCTE_LR bit (ON or OFF) in combination with the TCTENIS ‘node now in session’ bits (YES or NO).

<table>
<thead>
<tr>
<th>TCTE_LR</th>
<th>TCTENIS</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>ON</td>
<td>YES</td>
<td>Current session over LR</td>
</tr>
<tr>
<td>ON</td>
<td>NO</td>
<td>Previous session over LR</td>
</tr>
<tr>
<td>OFF</td>
<td>YES</td>
<td>Current session not LR</td>
</tr>
<tr>
<td>OFF</td>
<td>NO</td>
<td>Never bound, or previous session not LR</td>
</tr>
</tbody>
</table>

TCSE_LR (in the system entry) is set ON when the first LR session is bound, and OFF as a result of CNOS negotiation to release the connection. If TCSE_LR is ON and there are no bound sessions, the connection state is then ‘available’.

Modules

The modules listed below handle the VTAM LU6.2 support in CICS.

Session management state machines
- DFHZBKT
- DFHZCHS
- DFHZCNT
- DFHZCRT

Send and Receive processing
- DFHZRLP
- DFHZRLX
- DFHZRVL
- DFHZSDL
- DFHZSLX

CNOS
- DFHZLS1
- DFHZGCN
- DFHZGCA
Persistent Verification

- DFHCLS3

XLN and Resynchronization

- DFHCRRSY

**DFHZRVL**

DFHZRVL is invoked to issue an LU6.2 receive specific request to receive:

- Data
- Commands
- Responses
- Purge to end-chain (used by DFHZERH to clear incoming data)
- A single RU.

Two broad categories of RECEIVE data are recognized by CICS; both are processed as RECEIVE_WAIT requests to VTAM:

1. RECEIVE_WAIT, where CICS waits until input is received from VTAM before returning control to the caller. This applies to all RECEIVE response and command requests, and to data requests where the minimum length to be received is greater than zero.
2. RECEIVE_IMMEDIATE, where CICS immediately returns control to the caller without waiting for VTAM to complete the request unless the data is already in the VTAM buffer, in which case it processes the data in the same way as for RECEIVE_WAIT before returning to the caller. This is requested via a minimum length of zero. It is used by the RECEIVE_IMMEDIATE call for the SAA communications interface, by a LOOK_AHEAD call, and in support of timely receipt of responses, ensuring earlier detection of an ISSUE_ERROR response from the partner LU.

The receive buffer is set up to receive the data, and the address of the receive exit DFHZRLX (driven on completion of the request) is stored into the receive RPL (RPL_B) before the RECEIVE macro is issued to VTAM. DFHZRVL is used by DFHZERH to determine the state of the session.

**DFHZRLP**

This module completes the LU6.2 receive specific processing for LU6.2 requests.

RECEIVE_IMMEDIATE requests are processed in two phases, that is, on two passes through DFHZRLP:

1. The RPL_B state machine (TCTERPBS) is set to indicate that the RECEIVE has been completed by VTAM; then the exit is taken from DFHZRLP.
2. This phase corresponds to the single phase used for processing RECEIVE_WAIT requests, that is, the requests are checked for successful completion, examined to determine whether data, a command, or a response has been received, and parameters indicating what has been received are then returned to the caller.

**Data received**

When data is received, DFHZRLP:

1. Sets the bracket and chain state machines, and returns indicators to DFHZARL according to the DFC flags received with the data:
   - Response type
   - CD
   - EC
2. If more data is required, DFHZRLP recalls DFHZRVL via the activate scan routine (DFHZACT) to reissue the RECEIVE, for example when:
   - End-chain has not yet been received, and there is still room in the receive buffer. If the minimum length requested has already been received, the type of RECEIVE is altered from RECEIVE_WAIT to RECEIVE_IMMEDIATE resulting in a READ_AHEAD call in anticipation of there being more data available, and any data already in the VTAM buffer is processed by DFHZRLP before returning to the caller.
   - The original request was for data, and what has been received and processed is a command (only LUSTAT or BIS can validly be processed by DFHZRLP).

3. Returns control to DFHZARL when:
   - Sufficient data has been received for a BUFFER or LL type request.
   - End-chain has been received because of CD, RQD2, or CEB.
   - FMH has been received.
   - The call was incomplete, but insufficient space remains in the receive buffer for further data.

If the data was received with RQD1, a response is sent synchronously by DFHZRLP using the receive RPL.

**Command received**

When a command is received, the actions of DFHZRLP depend on the command:
- For LUSTAT6 received, the command is treated as data. If BB is included, then an exception response is sent (sense X'0813' or X'0814').
- For BIS received, CLSDST is requested and the receive re-driven.

All other commands are incorrect.

**Response received**

When a response is received, DFHZRLP:

1. Carries out checks:
   - Does the sequence number match the number of the BB request?
   - If it is a definite response, was it expected?
   - If it is an exception response, was it a session-level error?

2. Sets the state machines.

3. Passes back the return code to the caller.

**DFHZSDL**

This module issues the SEND request to VTAM to transmit data, commands, and responses on LU6.2 sessions.

DFHZSDL transmits:
- Data from a send buffer or an application area
- The commands:
  - LUSTAT
  - RTR
  - BIS
- Responses.
Data transmission
If a SEND LAST command is issued, any outstanding completed receive RPL is first processed by queuing the TCTTE for RECEIVE processing by DFHZRLP, and any incomplete receive RPL is canceled via RESETSR.

For data transmission, DFHZSDL uses:

LMPEO  
Large message performance enhancement outbound. VTAM slices large messages into RUs.

BUFFLST  
Buffer list. VTAM accepts data from non-contiguous buffers.

USERRH  
User request header. The request header is passed in BUFFLST.

A maximum of two buffer list entries are used. The first buffer list entry addresses the data in the send buffer, and the second the data in the application area.

The request header is built in the first buffer list entry using parameters passed from DFHZARL. If an implicit send was requested, then CD, RQD2, and CEB are not checked. The first-in-chain (FIC) indicator is set after checking the chain state machine, and last-in-chain (LIC) is set whenever CD, RQD2, or CEB is included. Null data sent only-in-chain (OIC) is converted to an LUSTAT6 command. The address of the send exit DFHZSLX is stored in the send RPL, and the VTAM SEND macro is issued. On completion of the SEND request, the bracket and chain state machines are set according to the DFC indicators. These state machines are used extensively by DFHZERH to determine the state of the session before executing an error request.

Command transmission
The LUSTAT6 command is sent with:
- CEB to terminate the BIND_in_bracket state
- Null data for OIC
- CB, RQD1 to BID for bracket.

The RTR command requests BB after a BID request is rejected with sense code X'0814'.

The BIS command shows bracket termination before CLSDST.

On completion of the SEND request, the exit DFHZSLX is invoked. LUSTAT causes the bracket and chain state machines to be set as for normal data flow.

Response transmission
DFHZSDL transmits ER1 and DR2 responses. The sequence number associated with the response is that of the path information unit (PIU) that initiated the current bracket. DFHZSDL uses the receive RPL (RPL_B) to send responses thus ensuring that the RU is returned with the response, unless the response is an ISSUE_ERROR request, in which case the send RPL is used. The response is sent synchronously, and POST=SCHED is included in the VTAM command, so that an exit routine is not involved. On return from VTAM, DFHZSDL sets the bracket and chain state machines accordingly.
DFHZSLX

The DFHZSLX module is the VTAM exit that is driven on completion of a SEND request. If the request completed successfully, the bracket and chain state machines are set to show the new state of the session. If the SEND request was data DRI, DFHZRVL is invoked via DFHZACT to receive the response.

DFHZRLX

The DFHZRLX module is the VTAM exit that is scheduled on completion of an LU6.2 RECEIVE_SPECIFIC request. DFHZRLX queues the completed RPL to a chain anchored from TCTVRLPQ in the TCT prefix. DFHZDSP dequeues the RPLs for further processing by DFHZRLP.

DFHCLS3

In the local CICS system, DFHCLS3 is invoked using the DFHLUS macro, which issues a DFHIC TYPE=PUT macro to start the appropriate transaction (CLS3) with data recorded on temporary storage indicating the requested operation.

The DFHLUS operations can be:

**SIGNOFF**
- Sign off a user on the other LU

**TIMEOUT**
- Time out users.

The SIGNOFF and TIMEOUT operations apply to persistent verification signons only.

DFHCLS3 retrieves the temporary-storage record.

The SIGNOFF and TIMEOUT operations are performed directly by DFHCLS3. These operations are supported outbound only.

For SIGNOFF, DFHCLS3 is started by DFHZCUT when a user on the other LU must be signed off.

For TIMEOUT, DFHCLS3 is started by DFHZCUT during time-out processing of a persistent verification signed-on-from list, also known to CICS as a local userid table (LUIT).

DFHCLS3 performs the following actions:
1. Calls DFHZCUT to find a userid that needs to be timed out
2. Makes a sign-off call to the other LU
3. Calls DFHZCUT to remove the userid from the LUIT.

This sequence is repeated until there are no more userids to be timed out.

If DFHCLS3 abends during time-out processing, control passes to a SETXIT routine in DFHCLS3, which calls DFHZCUT to tidy up the relevant LUIT.

DFHZLS1

DFHZLS1 is the main program for the CICS implementation of the CNOS SNA service transaction. When acting as the initiator of a CNOS request (the CNOS source), it is invoked by the DFHZLS1M macro issuing a DFHIC TYPE=PUT for transaction id CLS1. The possible commands on the CNOS source system are:-
- **INITIALIZE_SESSION_LIMIT**
  Acquire the specified connection, using the MAXIMUM values from the RDO SESSIONS definitions (for the required session limit and number of winner sessions) on the CNOS command for each modegroup.

- **CHANGE_SESSION_LIMIT**
  Negotiate a change of the current session limit for a specified modegroup.

- **RESET_SESSION_LIMIT**
  Release the connection, negotiating all modegroups to a session limit of zero.

When acting as the receiver of a CNOS request (the CNOS target), DFHZLS1 is invoked by an attach FMH for TPN X’06F1’ sent from the CNOS source system, which is not necessarily CICS. The CNOS command sent with the attach FMH requests changes to the sessions in specified modegroups. In SNA terms, DFHZLS1 is handling a PROCESS_SESSION_LIMIT command. It issues a DFHLUC RECEIVE for the CNOS GDS that contains the details of the required command.

DFHZLS1 passes the parameters for each of the above commands through to DFHZGCN, where the detailed processing takes place.

**DFHZGCN**

DFHZGCN is an AP domain subroutine. It handles the four architected CNOS functions, as described below.

**INITIALIZE_SESSION_LIMIT**

This is a two pass function in CICS. First time through, DFHZGCN initiates the bind of the SNASVCMG winner session and returns. The bind processing eventually causes the “session started” routine in DFHZNAC to run. This re-issues the DFHZLS1M INITIALIZE_SESSION_LIMIT request, and the CNOS negotiation can then take place.

DFHZGCN performs the following actions:

1. Does a ‘privileged’ allocate (for a SNASVCMG session).
2. Builds an attach header.
3. Completes the building of the CNOS command, using MAXIMUM values in the TCTME.
4. Issues a SEND INVITE WAIT.
5. Issues a RECEIVE LLID.
6. Analyzes the responses to the command; SNA decrees that the CNOS source must accept the values returned.
7. Calls DFHZGCA to action the new values.
8. Sends messages DFHZC4900 and DFHZC4901 as appropriate.
9. Frees the session.

The above steps are repeated for each user modegroup in the connection.

**RESET_SESSION_LIMIT**

A connection release request is passed via DFHZLS1 to DFHZGCN.

DFHZGCN performs the following actions:

1. Does a ‘privileged’ allocate.
2. Builds an attach header.
3. Completes the building of one CNOS command, setting MAX, WIN, and LOS values to zero, and mode names affected to ALL.
4. Issues SEND INVITE WAIT.
5. Issues RECEIVE LLID.
6. Analyzes the response to the command; the CNOS target must accept zero sessions (DRAIN can be changed from ALL to NONE).
7. Calls DFHZGCA to action the new values.
8. Sends message DFHZC4900.
9. Frees the session.

**CHANGE_SESSION_LIMIT**
DFHZLS1 is started from the EXEC API or CEMT via DFHEIQSM to change the session limit for a specific modegroup.

DFHZGCN performs the following actions:
1. Does a ‘privileged’ allocate.
2. Builds an attach header.
3. Completes the building of one CNOS command, setting MAX and WIN values.
4. Issues SEND INVITE WAIT.
5. Issues RECEIVE LLID.
6. Analyzes the responses to the command; SNA decrees that the CNOS source must accept the values returned.
7. Calls DFHZGCA to action the new values.
8. Sends messages DFHZC4900 and DFHZC4901 as appropriate.
9. Frees the session.

**PROCESS_SESSION_LIMIT**
DFHZLS1 is attached, and calls DFHZGCN.

DFHZGCN performs the following actions:
1. Addresses the CNOS command that DFHZLS1 passed.
2. For each mode group specified, determines whether the values for session limit, source contention winners and source contention losers are acceptable. If not, the values are adjusted (negotiated) according to rules laid down by SNA.
3. If this system is currently performing shutdown, negotiates down to session limit zero.
4. Calls DFHZGCA to action the new values.
5. Sends the CNOS reply containing the negotiated values.
6. Sends messages DFHZC4900 and DFHZC4901 as appropriate.

**DFHZGCA**
DFHZGCA is an AP domain subroutine. It has three separate functions, as described below.

**ACTION_CNOS_AND_CONNECT**
After a CNOS negotiation DFHZGCA is responsible for changing the state of a specified modegroup to reflect the new values. There are three types of action required.
1. Put sessions in/out of service for session limit increase/decrease.
2. Set sessions to winner/loser in line with negotiated values.
3. Bind/unbind sessions for session limit decrease, autoconnect processing or contention polarity switch.

**SET_NEGOTIATED_VALUES**
This function is used by DFHZGPC during persistent sessions restart to set the saved CNOS values in the modegroup without any binding/unbinding of sessions.

**ENSURE_SESSIONS_BOUND**
DFHZXRE0 invokes this function during persistent sessions restart because recovery processing can lead to LU6.2 sessions becoming unbound. It is important to ensure that they are re-bound in accordance with the autoconnect setting.

### Exits
No global user exit points are provided for this function.

### Trace
All of the above mentioned modules have entry and exit trace points. Several of them also have exception and level 2 trace points. All of these trace points are from the AP domain and have ids in the range FB00-FCFF.

For more information about the trace points, see the [CICS Trace Entries](#). For more information about using traces in problem determination, see the [CICS Problem Determination Guide](#).
Chapter 108. VTAM persistent sessions

This section describes how CICS handles VTAM persistent session support. It uses VTAM 3.4.1 persistent LU-LU session improvements to provide restart-in-place of a failed CICS without the need for network flows to re-bind CICS sessions.

Persistent sessions can either be Single Node Persistent Sessions (SNPS) or Multi Node Persistent Sessions (MNPS) depending on how your VTAM network is set up.

If CICS is to support MNPS then PSTYPE=MNPS must be specified in the SIT.

The following mainly describes SNPS. Sections are added where MNPS differs from SNPS.

For an overview of persistent sessions, and a comparison with XRF, see the CICS Transaction Server for z/OS Release Guide.

For an introduction to this topic from the VTAM point of view, see the Advanced Communications Function for VTAM Programming manual, SC31-6348.

Design overview

CICS support of persistent sessions includes the support of all LU-LU sessions except LU0 pipeline and LU6.1 sessions. CICS determines for how long the sessions should be retained from the PSDINT system initialization parameter. (This is a user-defined time interval.) If a failed CICS is restarted within this time, it can use the retained sessions immediately—there is no need for network flows to re-bind them.

This interval can be changed using the CEMT SET VTAM command, or the EXEC CICS SET VTAM command, but the changed interval is not stored in the CICS global catalog, and therefore is not restored on an emergency restart.

If CICS is terminated through CEMT PERFORM SHUTDOWN IMMEDIATE, or if CICS fails, VTAM holds CICS’ sessions in “recovery pending” state.

During emergency restart, CICS restores those sessions pending recovery from the CICS global catalog and the CICS system log to an “in session” state. This happens when CICS opens its ACB.

Subsequent processing is LU dependent: cleanup and recovery for non-LU6 persistent sessions is similar to that for non-LU6 backup sessions under XRF. Cleanup and recovery for LU6.2 persistent sessions maintains the bound session when possible but there may be cases where it is necessary to unbind and re-bind the sessions, for example, where CICS fails during a session resynchronization.

The end user of a terminal sees different symptoms of a CICS failure following a restart, depending on whether VTAM persistent sessions, or XRF, are in use:

- If CICS is running without VTAM persistent sessions, or XRF, and fails, the user sees the VTAM logon panel followed by the “good morning” message (if AUTOCONNECT(YES) is specified for the TYPETERM resource definition).
VTAM persistent sessions

- If CICS does have persistent session support and fails, the user perception is that CICS is hanging: the screen on display at the time of the failure remains until persistent session recovery is complete. After a successful CICS emergency restart, the recovery options defined (in RECOVOPTION) for the terminals or sessions take effect. If SYSDEFAULT is specified as the value for RECOVOPTION, the user can clear the screen and continue to enter CICS transids. If MESSAGE is specified for the RECOVNOTIFY attribute of the TYPETERM resource definition, the user is notified of the successful recovery. If CICS does not restart within the specified interval, the sessions are unbound, as if there has been a CICS failure without persistent session support in the system.

Note: SNPS support does not retain LU-LU sessions after VTAM, MVS, or CEC failure. Nor are sessions retained after the following commands:
- SET VTAM FORCECLOSE
- SET VTAM IMMCLOSE
- SET VTAM CLOSED
- PERFORM SHUTDOWN
- VARY INACT ID=applid

MNPS differs from SNPS in that MNPS support retains LU-LU sessions after a VTAM and MVS failure. The sessions are also retained after:
- SET VTAM FORCECLOSE

Persistent Sessions Restart flow

The following describes the flow of control for:
1. The enabling of persistence
2. The sessions that persist at start up time
3. The sessions that persist during dynamic open.

Enabling of persistence

Summary:
1. VTAM ACB opened with PARM=PERSIST=YES
2. VTAM levels checked.
3. VTAM SETLOGON OPTCD=PERSIST or NPERSIST

More detail: Persistence is enabled by:
1. The VTAM ACB is opened with PARM=PERSIST=YES - specified in DFHTCTPX.
2. DFHZSLS calls DFHZGSL to issue SETLOGON OPTCD=PERSIST/NPERSIST.

DFHZSLS copies 8 bytes of VTAM information into the TCT prefix. These bytes contain details of the VTAM level and the functions which it supports. Previous releases of CICS only copy 4 bytes of VTAM data. The use of persistent sessions is dependent upon the level of VTAM present being at least V3R4.1. This level of VTAM returns more function bit data to CICS than previous versions and supports the use of persistent sessions. Checks are made by CICS of the current VTAM level and the VTAM level against which the TCT was generated. If either level is not high enough, parameters relating to the use of persistent sessions are not used when macros are called.
Sessions that persist at start up time

Summary:
1. Task CGRP runs DFHZCGRP
2. DFHZCGRP calls DFHZGRP
3. DFHZGRP issues VTAM INQUIRE
4. DFHZGRP either:
   - terminates session via DFHZGUB issuing CLSDST/TERMSESS or
   - restores the session with OPNDST TYPE=RESTORE
5. DFHZGRP queues restored sessions for further processing.
6. DFHZGRP issues RECEIVE_ANYs.
7. DFHZGRP does some CNOS work.
8. DFHZGRP does some URD work.
9. Queued sessions get restored.

More detail: Sessions that persist at startup time are processed by:
1. Attach task CGRP - program DFHZCGRP in DFHSII11 after TCRP is attached.
2. DFHZCGRP calls DFHZGRP with a START_TYPE of:
   - COLD
   - WARM
   - EMER_XRF
   - EMER
3. DFHZGRP issues VTAM INQUIREs in ‘chunks’, that is VTAM is passed an area whose size is defined in the TCT Prefix.
   The area is filled with NIBS by VTAM. DFHZGRP scans the NIBS and decides whether to UNBIND or OPNDST each session.
   For COLD, WARM and EMER_XRF all sessions are unbound.
   For EMER some sessions are unbound and some restored depending on the circumstances.
4. Restored sessions are queued to DFHZACT for further processing by DFHZXRC or DFHZXPS.
5. RECEIVE_ANY Initialization done.
6. CNOS records are processed by making calls to DFHZGPC.
7. URDS are reset to AWAITING RE_SYNCHRONIZATION for EMER only.
8. DFHZACT calls DFHZXRC or DFHZXPS for each session queued by DFHZGRP.

Task and module Flow diagram:

-> indicates an ATTACH

```
   TASK
   1 TCP III TCRP
   -- -- ---- ----
   . . . . CSSY CGRP
   . . . . . . .
   . . . . . . .
   SIII->ZCSTP
   ZDSP
   .ZSL
   .ZGSL
   Spin on
   TCTV_RA_DONE
```
VTAM persistent sessions

```
SII1---------->SII1 --->TCRP
SII1-----------------> ZCGRP
. . . install .ZGRP
. . . TCTTEs . INQUIRE on persistent sessions
. . . . . . . . wait on TCTVCECB (EMER)
. . . . . Post TCTVCECB
. . . . task end
. . . . process persistent sessions
. . . . RECEIVE_ANY processing
ZDSP continues <------------------- set TCTV_RA_DONE
. . . post TCTV_ZGRP_FIN_ECB
. . . task end
. Wait on TCTV_ZGRP_FIN_ECB
$111
. SETLOGON START
. Start CXRE task
. Control is Given to CICS
ZACT
. ZXRC
. ZXPS
```

Task and module flow - more detail:

1. Startup runs as normal until DFHSII1 has started the TCP (CSTP) task and DFHZDSP runs.
2. DFHZDSP calls DFHZSLS.
   - If VTAM is at least V3R4.1, DFHZSLS calls DFHZGSL to issue SETLOGON OPTCD=PERSIST if the SIT PSDINT value is a valid non 0 value.
   - If the VTAM level is V3R4.0 or PSDINT is 0 or defaulted with higher levels of VTAM, DFHZSLS calls DFHZGSL to issue SETLOGON OPTCD=NPERSIST.
   - If the VTAM level is lower than V3R4.0, the SETLOGON OPTCD call is not made since PERSIST and NPERSIST are not supported for these VTAM releases.

DFHZSLS does NOT issue RECEIVE OPTCD=ANY. It returns to DFHZDSP which ‘spins’ until TCTV_RA_DONE is set by DFHZGRP when the RECEIVE_ANYs have been successfully issued.
3. DFHSIII1 attaches the III task which continues to run code in DFHSII1.
4. DFHSIII1 (III) attaches and calls DFHTCRP as a system task then attaches task CGRP, which runs program DFHZCGRP which calls ZGRP.
5. DFHZGRP calls DFHZGUB if there are any sessions to unbind.
6. DFHZGRP queues any sessions to be restored to DFHZACT.
7. DFHZGRP sets TCTV_RA_DONE after issuing RECEIVE_NOWs to allow DFHZDSP to continue.
8. DFHZGRP posts TCTV_ZGRP_FIN_ECB.
9. When DFHZGRP finishes, control is returned to code in DFHZCGRP. DFHZCGRP checks the RESPONSE and REASON code. It sets TCTV_ZGRP_FAILED off if RESPONSE(OK) or RESPONSE(EXCEPTION) with REASON(ACB_CLOSED|INQUIRE_FAILED). Otherwise it sets TCTV_ZGRP_FAILED on.
10. DFHSIII1 waits on TCTV_ZGRP_FIN_ECB and checks TCTV_ZGRP_FAILED set by DFHSIII1.
If TCTV_ZGRP_FẢILED is off then DFHSII1 continues. Otherwise it sets INITIDERR which causes CICS to terminate when the other tasks have finished.

11. Just before CONTROL IS GIVEN to CICS, DFHSIJ1 attaches the CXRE task to run DFHZXRE0 which does some additional PRSS processing.

12. DFHZXRC or DFHZXPS are then called to process any TCTTEs queued to DFHZACT.

13. DFHZXRC is called by DFHZACT to process non-APPC sessions which have not been unbound by DFHZGRP. It takes one of the following actions depending on the state of the session, the terminal type, and how the TYPETERM for the session has been defined to CICS:
   - Send END_BRACKET
   - Send CLEAR (followed by START_DATA_TRAFFIC for SNA devices which support it)
   - Unbind.

For those devices for which the cleanup action is not to unbind, the TCTTE is queued to DFHZNAC and message DFHZC0146 is issued for the session.

As part of the processing for message DFHZC0146, any recovery notification requested for the session is initiated:
   - If the requested recovery notification is MESSAGE, DFHZNCA sends a BMS map to the terminal.
   - If the requested recovery notification is TRANSACTION, DFHZNCA initiates the requested transaction.

14. DFHZXPS is called by DFHZACT to process APPC sessions. DFHZXPS takes one of the following courses of action depending on the setting of TCTE_PRSS on entry:
   - Examine the data pointed to by TCTV_PRSS_CV29_PTR to determine the state of the session at system failure.
     a. If a task is attached call DFHZGDA to issue DEALLOCATE,ABEND for the task still running on the partner.
     b. If no task is attached but there is further recovery to be done e.g. bid recovery, outstanding responses, set the TCTTE to a state which allows this further recovery to proceed. If the existing mechanism will carry out the recovery without further intervention by DFHZXPS then remove the TCTTE from the DFHZACT queue, otherwise requeue the TCTTE to DFHZACT and DFHZXPS will be recalled at a later stage to finish recovery processing.
     c. If no task is attached and there is no further recovery to be done, remove the TCTTE from the DFHZACT queue as recovery is now complete.
   - Recall DFHZGDA to continue with DEALLOCATE,ABEND or REJECT.Attach processing.
   - Requeue the TCTTE to DFHZACT if a SEND (for example of an outstanding response) which was set in motion by an earlier instance of DFHZXPS is still in progress.
   - CLSDST the session if an error has occurred during the recovery process.
   - Carry out further recovery as described above, if required, following successful completion of DEALLOCATE,ABEND processing.
VTAM persistent sessions

- Remove the TCTTE from the DFHZACT queue when all recovery has completed.

Sessions that persist at Dynamic Open
If VTAM fails but CICS stays up SNPS sessions do not exist. For MNPS they do persist. When VTAM crashes, CICS does not delete the autoinstalled resources and resets all the terminal and connection sessions to unopened state.

Summary:
1. CEMT SET VTAM OPEN
2. DFHEIQVT calls DFHZOPA
3. DFHZOPA calls DFHZSLS
4. DFHZSLS call DFHZGSL
5. DFHZGSL issues SETLOGON PERSIST or NPERSIST
6. DFHZOPA calls DFHZGRP
7. DFHZGRP issues INQUIRE PERSESS
8. DFHZGRP terminates session via DFHZGUB issuing CLSDST/TERMSESS. However, if MNPS is in use the sessions are OPNDST RESTOREd instead.
9. DFHZGRP issues RECEIVE_ANYs
10. DFHZGRP deletes CNOS catalogue records
11. DFHZOPA issues SETLOGON START

More detail: Sessions that persist after the ACB has been opened using CEMT SET VTAM OPEN or EXEC CICS SET VTAM OPEN are processed by:

1. CICS is running with the VTAM ACB closed. CEMT SET VTAM OPEN or EXEC CICS SET VTAM OPEN is issued.
2. DFHEIQVT calls DFHZOPA to open the ACB.
3. DFHZOPA calls DFHZSLS.
4. DFHZSLS calls DFHZGSL.
5. DFHZGSL issues VTAM macro calls dependent upon the VTAM level and PSDINT value.
   - If VTAM is at least V3R4.1, DFHZGSL issues SETLOGON OPTCD=PERSIST if the SIT PSDINT value is a valid non 0 value.
   - If the VTAM level is V3R4.0 or PSDINT is 0 or defaulted with higher levels of VTAM, DFHZGSL issues SETLOGON OPTCD=NPERSIST.
   - If the VTAM level is lower than V3R4.0, the SETLOGON OPTCD call is not made since PERSIST and NPERSIST are not supported for these VTAM releases.
6. DFHZOPA then calls DFHZGRP with startup type of DYNOPEN.
7. DFHZGRP issues INQUIRE PERSESS with a storage area that will take up to about 400 sessions - INQUIRE PERSESS is reissued until all the NIBs have been obtained from VTAM.
8. DFHZGRP calls DFHZGUB if there are any sessions to unbind. For MNPS DFHZGRP instead issues OPNDST RESTORE for each session that persists.
9. DFHZGRP issues RECEIVE_ANYs.
10. DFHZGRP calls DFHZGCC to delete CNOS records.
11. If ZGRP returns RESPONSE(OK) or RESPONSE(EXCEPTION) with REASON(ACB_CLOSED | INQUIRE_FAILED) then DFHZOPA issues SETLOGON OPTCD=START. Otherwise it causes DFHZSHU to be run to close the VTAM ACB and then returns to DFHEIQVT.
TCB Concurrency

Summary: If SUBTSKS = 1 Specified in SIT
- DFHZGRP switches to concurrent TCB if enough NIBs to process.
- INQUIRE PERSESS work done concurrently with TCRP ZC INSTALL.
- DFHZGUB switches to concurrent TCB if enough NIBs to process. (EMER only).
- OPNDST RESTORE and CLSDST/TERMSESS done concurrently.

More detail: During startup DFHZGRP is attached as a task and runs at the same time as other startup tasks such as DFHTCRP and DFHRCRP. However, DFHZGRP also switches to use the CONCURRENT TCB if there are enough NIBs to process during EMER start.

This allows DFHZGRP to issue INQUIRE OPTCD=PERSESS as many times as is necessary, concurrently with the TCTTEs being restored by DFHTCRP.

When DFHZGRP finishes INQUIREing it waits for DFHTCRP to finish before matching each persisting NIB with the restored TCTTEs.

Each NIBLIST is then OPNDST OPTCD=RESTOREd and while this is running asynchronously DFHZGUB is called to run under the concurrent TCB if there are enough NIBs to be unbound in the NIBLIST.

Modules

ZC (terminal control) together with the following:

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHZCGRP</td>
<td>Program initiated by task CGRP to set up the start type and to call DFHZGRP during initialization. It then analysis the response from DFHZGRP and decides if CICS can continue or not.</td>
</tr>
<tr>
<td>DFHZGCA</td>
<td>Sets the appropriate ZC control blocks to reflect the currently agreed Change Number Of Session (CNOS) values for an LU6.2 connection.</td>
</tr>
<tr>
<td>DFHZGCC</td>
<td>Performs catalogue and retrieval of CNOS data.</td>
</tr>
</tbody>
</table>

This module is called when CICS needs either to store or to recover CNOS values. During a CICS run, all CNOS values are written to the global catalogue. Under normal circumstances they are not needed. However, if a persistent sessions restart is performed, it is necessary to recover the CNOS values which were in operation at the time of the CICS failure. This is achieved by having a record on the global catalogue which can be read in during PRSS restart and used to restore the sessions to their pre-failure state.

This module will handle the maintenance of the CNOS records during normal CICS operation and the recovery of the records during PRSS recovery.

| DFHZGCN | Handles the process of LU6.2 Change Number Of Sessions (CNOS) negotiation, acting as either the source or target end of the conversation, and calls DFHZGCA to action the resulting changes. |
### VTAM persistent sessions

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DFHZGDA</strong></td>
<td>The role of DFHZGDA is to take control of APPC conversations which have persisted across a CICS failure, and to ensure that they are terminated cleanly, by issuing a Deallocate(Abend) informing the partner LU that the CICS transaction has abended. If DFHZGDA is working correctly, the fact that CICS has failed and been restarted should be transparent to the partner LU; all he knows is that the CICS transaction to which he was talking has terminated. DFHZGDA also performs REJECT_ATTACH processing for syncllevel 2 conversation which are started by the partner before Exchange Lognames has been done after a persistent sessions restart.</td>
</tr>
<tr>
<td><strong>DFHZGPC</strong></td>
<td>Performs recovery of CNOS values for modegroups. This module is called when CICS is performing a persistent sessions (PRSS) restart. When a PRSS restart is performed, it is necessary to do more than merely recover the session. It is also necessary to recover the CNOS state which the sessions had prior to the CICS failure. DFHZGCC will have maintained a record of the CNOS state on the global catalogue. This record will now be used in this module in an attempt to restore CNOS values.</td>
</tr>
</tbody>
</table>
| **DFHZGPR** | The role of DFHZGPR is to update the global catalog whenever it is necessary to add, delete, or test for a record indicating that an APPC connection has a Persistent Resource associated with it. A Persistent Resource can be defined as some session state, or piece of work upon which the partner LU is dependent, and which will be lost in the event of CICS failing. The only Persistent Resource so far identified is:  
  - A shipped AID  
Prior to persistent sessions, the failure of the APPC session tells the partner that these resources have been lost, and drives his recovery. With the advent of persistent sessions, it is necessary for a persisting CICS to know that an APPC session had a Persistent Resource associated with it, so that the connection can be unbound (to drive the partners cleanup) and then rebound. |
| **DFHZGRP** | Initialize VTAM persistent sessions. DFHZGRP is a domain subroutine but is called by DFHZCGRP (task CGRP) during initialization. DFHZGRP is called during ZC initialization or when the VTAM ACB is opened dynamically by CEMT SET VTAM OPEN or EXEC CICS SET VTAM OPEN by DFHEIQT. The module does the following:  
1. OPNDST RESTOREs or CLSDST/TERMSESS any session that VTAM has held persisting, depending on start up type and session parameters.  
2. It calls DFHZGPC to re-instate CNOS records during an EMER restart, or calls DFHZGCC to delete CNOS catalogue records.  
3. It initializes the RECEIVE_ANY RPLs and issues the RECEIVEANYs. |
| **DFHZGSL** | Informs VTAM whether sessions are to persist or not. This module is called when CICS needs to set, unset or change the Persistent Sessions PSTIMER value. |
### Module Function

**DFHZGUB**
- Issue CLSDST or TERMSESS for individual NIBs in a NIBLIST.

This module is called by DFHZGRP to unbind nibs in a niblist in two ways:
- Unbind the entire NIBLIST for COLD, WARM, EMER\(+\)XRF and dynamic open.
- Unbind only the NIBs with NIBUSER = 0 for EMER starts.

**DFHZXPS**
- DFHZXPS handles Persistent Sessions recovery for APPC sessions. It does not deal with non-APPC sessions which are dealt with by DFHZXRC.

DFHZXPS is called by DFHZACT after OPNDST OPTCD=RESTORE has been issued successfully for a persisting APPC session. Both single and parallel APPC sessions are dealt with but there is no difference in the processing.

The task of DFHZXPS is to examine VTAM session tracking data which was hung off TCTE_PRSS CV29_PTR by DFHZGRP following a Persistent Sessions restart and if possible to update the TCTTE to allow work to continue on the session.

If it is not possible to determine the state of the session prior to system failure, or the session was not in a state which allows it to be recovered, the session will be unbound.

**DFHZXRC**
- DFHZXRC analyses the Session State Vector data that is hung off TCTE_PRSS CV29_PTR by DFHZGRP during an EMER restart, for each persisting session. The necessary action to cleanup and recover the session is then initiated.

### Diagnosing Persistent Sessions Problems

The following should be consulted when diagnosing problems with persistent session.

- **Trace**, TC level 1, 2 and exception in the range of AP FB10-FBFF.
- CEMT INQUIRE VTAM showing the PSDINT value.
- Console and CSNE logs:
  - Persistent session messages (DFHZC0001 to DFHZC0162)
  - Information produced by DFHZNAC
- Dumps taken by some of the above messages.
  - If a NIBLIST was present at the time the dump was taken then it can be examined by printing the TCP section of the dump.
- Last flow information - that is the CV29, FMH5, BIS and BID information is useful if a session is in the wrong state after a persistent session restart. This may have been diagnosed by an error message, or maybe missed and message DFHZC0146 or DFHZC0156 issued.

TCTE_PRSS CV29_PTR points to the CV29 etc which was created by DFHZGRP and used by DFHZXPS or DFHZXRC. It is freed when DFHZNAC issues message DFHZC0146 or DFHZC0156. Otherwise it is freed when the session is unbound.

It is traced by DFHZXPS as a TC level 1 trace.

If you have a dump, but no trace level 1 available, it is dumped in the TCP section for each TCTTE for which it still exists.
VTAM persistent sessions

- The contents of byte TCTE_PRSS are useful. Values other than X'00' and X'FF' indicate that something went wrong during the PRSS recovery. The possible values are listed in the CICS Data Areas. If a value is left in this byte, the meaning may give some indication as to where the recovery went wrong. The values are described later in this chapter.
- The contents of the state machines are useful.
  - TCTECNTS - contention state machine.
  - TCTEBKTS - bracket state machine.
  - TCTECHSS - chain state machine.
  - TCTEUSRS - user state machine.
- The contents of TCTE_BID_STATUS are useful. They are described later in this chapter.

Here are some possible problems:
- DFHZGGRP may cause CICS to terminate during initialization for the following reasons:
  - DFHZGGRP has been called with invalid parameters.
  - DFHZGGRP is unable to complete the receive any process.
  - DFHZGGRP has had a loop or abend.
  - DFHZGGRP is unable to switch back to the QR TCB.
  - DFHZGGRP has failed before any NIBs have been obtained from VTAM (with INQUIRE OPTCD=PERSESS).
  - DFHZGGRP or DFHZGUB has issued a VTAM request that failed to respond within 5 minutes. Issued with message DFHZC0128 and a system dump.

In each case DFHZGGRP or a function it has called issues a message giving a reason for the failure.
- Sessions may be unbound by DFHZGGRP for the following reasons:
  - This is a COLD, WARM, EMER + XRF restart.
  - This is a dynamic open of the ACB (e.g. CEMT SET VTAM OPE). However, if MNPS is in use sessions should be restored at this point.
  - The TCTTE has not been found - probably because it has not been cataloged (Autoinstall with AIRDELAY=0 or APPC clone). No message is written because this is considered to be normal.
  - CICS does not support recovery for LU61 or pipeline sessions.
  - The TCTTE does not match the NIB - possibly an operational mix-up - has the right GCD been used?
  - A terminal or session had RECOVOPT UNCONDREL|NONE specified.
  - A connection had PSRECOVERY NONE specified.
  - A matching mode group was not found - have you got the right GCD?
  - A suitable session was not found - this can occur if the CNOS values create many “up for grabs” sessions which were in use when CICS failed - this would occur if the session limit was high and the contention winners was low.

It may also occur if CICS was in the process of CNOSing from a high session limit to a low session limit at the time CICS failed.
Message DFHZC0111 is issued in this case.
  - An URD was found for the session so the entire connection is unbound to allow the connection to recover correctly.
- APPC Sessions may be unbound by DFHZXPS for the following reasons:
Some of the reasons are known states for which the session cannot be recovered. Others are unexpected errors.

Known states for which the session cannot be recovered:
- The last flow was a positive response to a bid with data.
- Exchange log names (transaction CLS2) was running when the system failed.
- A bind or bind security had not completed when the system failed.
- Because of the last thing to flow e.g. SIGNAL, the state of the session at the time of system failure cannot be determined.

Unexpected errors:
- A bad return code was received from a call to DFGZGDA.
- An attempt to reset the session from CS mode to CA mode or vice versa failed.
- The TCTE_PRSS byte contained an unexpected value on entry to DFHZXPS.
- The BIS, bid or CV29 data pointed to by TCTV_PRSS_CV29_PTR contained an unknown value or was inconsistent.
- An error occurred during some other part of the recovery process.
- An internal logic error occurred in DFHZXPS.

• Sessions may be unbound by DFHZGDA for the following reasons:
  - A SEND issued as part of Deallocate(Abend) processing has failed
  - A RECEIVE issued as part of Deallocate(Abend) processing has failed
  - A logic error is detected during Deallocate(Abend) processing

• Sessions may be unbound by DFHZXRC for the following reasons:
  - The user has specified RECOVOPT(RELEASESESS) and the session was in bracket at the time CICS failed.
  - End-Bracket and Clear/SDT could not be used to clean up the session.
  - Cold Start has been requested for the session.

• Message DFHZC0124 can be issued with inconsistent counts if:
  - DFHZGRP loops or abends.
  - The ACB is closed by VTAM operator commands whilst DFHZGRP is in control.

• LU6.2 Connections which might be expected to persist, may be unbound if there was a persistent resource associated with the connection when CICS failed (i.e. there was an asynchronous processing request in progress at the time CICS failed).

• Following a persistent sessions restart, LU6.2 partners may experience a series of unexpected abends with sense code 08640001 from the persisting CICS; this can occur either because there was a conversation in progress at the time CICS failed, and CICS has terminated the conversation with this code, or for syncllevel 2 conversations, the partner has attempted to initiate a conversation before Exchange Lognames has run following a persistent sessions restart.

• Some APPC sessions may hang following a persistent sessions restart because CICS has determined that it was in RECEIVE state at the time of the CICS failure, and issued a RECEIVE for the expected data, but the partner has not sent the expected data; the RECEIVE will not timeout in this situation, because RTIMOUT does not apply to sends issued by DFHZGDA.
Persistent Sessions status byte (TCTE_PRSS)

A new byte TCTE_PRSS has been introduced into the TCTTE to track the stage reached in the persistent sessions recovery of a session. If for some reason persistent sessions recovery does not complete, this field can give a useful indication of the stage reached in recovery when the problem occurred.

TCTE_NO_PRSS_RECOVERY (X'00')
- The value TCTE_PRSS would normally contain, meaning:
  - Persistent sessions are not being used.
  - The session was successfully recovered following a persistent sessions restart.
  - The session has been CLSDSTed and restarted since a persistent sessions restart.
  - The session was started after any persistent sessions restart.

If this was a persisting VTAM session, then TCTE_PRSS will have been set to this value on completion of recovery notification for non-LU6.2 (see NAPES84 and NAPES83 routines), or in the session restarted logic of NAPES51 for LU6.2 sessions.

TCTE_NIB_MATCHED (X'01')
- Placed in TCTE_PRSS by DFHZGRP once a TCTTE has been found which matches the NIB of a persisting VTAM session. This should be a transient value, as the OPNDST OPTCD=RESTORE is issued soon after, and should cause TCTE_PRSS to be updated.

TCTE_OPNDST_RESTORE_COMPLETED (X'02')
- Placed in TCTE_PRSS once an OPNDST OPTCD=RESTORE has been successfully issued for a VTAM Session by DFHZGRP. Once this value has been placed in TCTE_PRSS, the TCTTE should be put onto the activate scan queue to await processing by DFHZXRC or DFHZXPS.

TCTE_ZXRC_CLEANUP (X'20')
- Placed in TCTE_PRSS by DFHZXRC when it begins processing a TCTTE. All TCTE_PRSS values relating to DFHZXRC processing are X'2x'. This value remains in TCTE_PRSS until the TCTTE is queued to DFHZNAC for the issuing of message DFHZC0146. If for some reason the TCTTE does not get recovered and TCTE_PRSS contains this value, then DFHZXRC may be the culprit.

TCTE_ZXPS_CLEANUP (X'30')
- All TCTE_PRSS values beginning (X'3x') indicate that DFHZXPS is doing its recovery/cleanup processing for this TCTTE. TCTE_PRSS is updated to this value on entering DFHZXPS for the first time. DFHZXPS should only be processing LU6.2 sessions.

TCTE_ZXPS_DEALLOCATE_ABEND (X'31')
- DFHZXPS places this value into TCTE_PRSS prior to calling DFHZGDA for the first time. It indicates that DFHZXPS has determined that an APPC
conversation was taking place at the time CICS failed, and that DFHZXPS is calling DFHZGDA to terminate that conversation. Again, this should be a transient value, as DFHZGDA will update TCTE_PRSS as it proceeds with its DEALLOCATE(ABEND) processing.

**TCTE_ZXPS_SEND_IN_PROGRESS (X'32')**
DFHZXPS has determined that bidding activity was taking place at the time CICS failed, and that some kind of SEND is required to complete the bid flows. If the session hangs with this value in TCTE_PRSS there may have been some kind of problem with unexpected bid flows taking place.

**TCTE_ZXPS_ISSUE_RECOVERY_MSG (X'33')**
When DFHZXPS has completed recovery and cleanup for the session, it puts this value into TCTE_PRSS before queueing the TCTTE to DFHZNAC for recovery message processing.

**TCTE_ZGDA_FMH7_SEND (X'41')**
All TCTE_PRSS values with X'4x' indicate that DFHZGDA is terminating the APPC conversation which was in progress on the session at the time CICS failed. This value indicates that DFHZGDA is in the process of issuing a SEND for the FMH7 which is to terminate the conversation.

**TCTE_ZGDA_FMH7_COMP (X'42')**
DFHZGDA has completed its Deallocate(Abend) processing. This value in TCTE_PRSS indicates to DFHZXPS that it may continue with any outstanding recovery/cleanup processing of its own.

**TCTE_ZGA_FMH7_REC (X'43')**
DFHZGDA has determined that CICS was in RECEIVE state at the time CICS failed, and has issued a RECEIVE for the RU expected from the partner. This value may appear in sessions which appear to be hanging following a persistent sessions restart. If the partner never issues the expected SEND, the RECEIVE will never be executed. Since this RECEIVE is issued under the TCP task, the RECEIVE will not be subject to any RTIMEOUT.

**TCTE_ZGDA_REC_EOC (X'44')**
Placed in TCTE_PRSS if the first RECEIVE of the DFHZGDA module following the persistent sessions reveals that the partner is in the middle of sending a chain of RUs. If TCTE_PRSS contains this value, DFHZGDA has issued a RECEIVE_PURGE for the session. Again, depending on how quickly the partner sends the expected data, this session may appear to hang.

**TCTE_ZGDA_SEND_RESP (X'45')**
Placed in TCTE_PRSS if DFHZGDA has to issue a SEND for a response during Deallocate(Abend) processing.

**TCTE_PRSS_CLSDST_SCHEDULED (X'FF')**
This value is placed in TCTE_PRSS if there is an error, or if in the course of persistent sessions recovery it is decided to terminate the persisting session. This may be for a variety of reasons; some of which are:

- An error occurred issuing a SEND or RECEIVE during persistent sessions recovery.
- RECOVOPT(NONE) or RECOVOPT(UNCONDREL) was specified for the session.
- The only recovery action which DFHZXRC could take was to terminate the session.
VTAM persistent sessions

The X'FF' value remains in TCTE_PRSS as an indicator that the session was terminated during PRSS recovery. Only when the session is restarted is the value overwritten with X'00'.

Bid status byte (TCTE_BID_STATUS)

DFHZXPS uses a byte in the TCTTE, TCTE_BID_STATUS, to track the various stages of recovery. It is possible to examine this byte to determine the stage of recovery reached by DFHZXPS.

The byte values have the following meanings.

• X'00'
  This session has not been processed by DFHZXPS.

• X'01' TCTE_SEND_POSITIVE_RESPONSE
  A positive response is to be sent to a bid which was received before system failure. This value is changed to X'07' TCTE_SENT_POSITIVE_RESPONSE before the TCTTE is requeued to DFHZACT for the SEND and so will only be seen if DFHZXPS abends. When the response has been sent DFHZXPS will be recalled.

• X'02' TCTE_SEND_NEGATIVE_RESPONSE
  A negative response is to be sent to a bid with data which was sent before system failure. This needs to be followed by RTR and so the status byte is changed to X'03' SEND_RTR before the TCTTE is requeued to DFHZACT for the SEND. This is another value which should only be seen if DFHZXPS abends. DFHZXPS will be recalled when the response has been sent.

• X'03' TCTE_SEND_RTR
  Recovery is complete apart from the need to send RTR. This will be done by DFHZDET and DFHZXPS will not be recalled.

• X'04' TCTE_SENT_RTR
  RTR was sent before system failure. There is no recovery to be done. DFHZXPS will not be recalled.

• X'05' TCTE_SEND_LUSTAT_EB
  Either we received a positive response to a bid or we sent a positive response to RTR before the system failed. The bid now has to be canceled. This will be done by DFHZDET and DFHZXPS will not be recalled.

• X'06' TCTE_AWAITING_BB_RESPONSE
  A bid was sent before the system failed. No further recovery is required. When the response arrives from the partner the bid will be canceled. DFHZXPS will not be recalled.

• X'07' TCTE_SENT_POSITIVE_RESPONSE
  Either a positive response has been sent to a bid or one is about to be sent (see above under SEND_POSITIVE_RESPONSE). In the former case DFHZXPS will not be recalled, in the latter case it will.

• X'08' TCTE_0814_RECEIVED
  A negative response was sent to a bid before the system failed. Any further recovery will be carried out by DFHZDET and DFHZXPS will not be recalled.

• X'09' TCTE_0813_RECEIVED
  As above except that no RTR is expected in this case. No further recovery processing is needed from either DFHZXPS or DFHZDET.

• X'0A' TCTE_SEND_RECOVERY_MESSAGE
  All recovery is now complete.
• X'0B' TCTE_DR1_OUTSTANDING
  The last flow was inbound with CEB,RQD1 and so although there is no task to
  ABEND a response is still expected by the partner. We requeue for DFHZSDL to
  send the response and any further recovery processing will be done by
  DFHZDET. DFHZXPS will not be recalled.

• X'0C' TCTE_DR1EXPECTED
  As above except that the last flow was inbound. DFHZDET will arrange for the
  response to be received. DFHZXPS will not be recalled.

TCTE_BID_STATUS must be used in conjunction with TCTE_PRSS to determine
the state of the recovery. If TCTE_PRSS is set to
TCTE_XZPS_ISSUE_RECOVERY_MESSAGE, or to a state which indicates that
recovery is complete, DFHZXPS has finished processing. If not DFHZXPS will be
recalled at a later stage.

---

Summary of persistent session waits

The DFHDSSRM waits are summarized here. All but PSUNBECB are posted by
DFHZGRP.

<table>
<thead>
<tr>
<th>Module Type</th>
<th>Resource_name</th>
<th>Resource_type</th>
<th>ECB</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHSII1 MVS</td>
<td>ZGRPECB</td>
<td>AP_INIT</td>
<td>TCTV_ZGRP_FIN_ECB</td>
</tr>
<tr>
<td>DFHZGUB OLDC</td>
<td>PSUNBECB</td>
<td>ZC_ZGUB</td>
<td>WAIT_RPL_ECB</td>
</tr>
<tr>
<td>DFHZGRP MVS</td>
<td>PSOP1ECB</td>
<td>ZC_ZGRP</td>
<td>OPNDST_ECB</td>
</tr>
<tr>
<td>DFHZGRP MVS</td>
<td>PSOP2ECB</td>
<td>ZC_ZGRP</td>
<td>OPNDST_ECB</td>
</tr>
<tr>
<td>DFHZGRP MVS</td>
<td>PSINQECB</td>
<td>ZC_ZGRP</td>
<td>INQUIRE_ECB</td>
</tr>
<tr>
<td>DFHZGRP OLDC</td>
<td>TCTVCECB</td>
<td>ZC_ZGRP</td>
<td>TCTVCECB</td>
</tr>
</tbody>
</table>

where the waits are issued for the following reasons:

ZGRPECB
  Wait for DFHZGRP to complete.

PSUNBECB
  Wait for free unbind RPL from RPL pool anchored from
  TCTV_PRSS_RPL_POOL_PTR.

PSOP1ECB
  Wait for OPNDST RESTORE to complete.

PSOP2ECB
  Wait for OPNDST RESTORE to complete after UNBINDs have failed.

PSINQECB
  Wait for INQUIRE PERSESS to complete.

TCTVCECB
  Wait for TCTTEs to finish installing (DFHTCRP).

---

VTAM exits

The VTAM exits SYNAD (DFHZSYX) or LERAD (DFHZLEX) may be driven
during persistent sessions recovery.
VTAM persistent sessions

In DFHZGRP, before INQUIRE OPTCD=PERSIST is issued, or in DFHZGUB before CLSDST or TERMSESS are issued CICS sets the RPL user field to -2 to indicate to the exits that they must do NO processing at all. This is because these macros may be issued under the concurrent TCB.

In DFHZGRP before OPNDST OPTCD=RESTORE is issued CICS sets the RPL user field to -1 to indicate to the exits that they should try minimum recovery - that is they set the return code to TCZSYXPR if an error can be retried, or TCZSYXCF if it is a permanent error.

If an error occurs in DFHZGSL for SETLOGON OPTCD=PERSIST DFHZSYX returns immediately (as for RPL user field = -2).

If MNPS is in use and VTAM crashes DFHZTPX is driven with a code of 8. If SIT parameter PSTYPE=MNPS was specified then DFHZTPX does NOT schedule the autoinstalled TCTTEs for deletion. They are scheduled for CLSDST CLEANUP instead by DFHZSHU.

See the OS/390 eNetwork Communications Server: SNA Programming manual, SC31-8573, for general VTAM exit information.

Trace

The following point IDs are provided for persistent sessions recovery (DFHZGCA, DFHZGCC, DFHZGCN, DFHZGDA, DFHZGFC, DFHZGPR, DFHZGR, DFHZGSL, DFHZGUB, DFHZCGRP, DFHZXPS, DFHZXRC):

• AP FB10 through AP FBFF, for which the trace levels are TC 1 and TC 2.

For more information about the trace points, see the CICS Trace Entries. For more information about using traces in problem determination, see the CICS Problem Determination Guide.

Statistics

The following statistics are produced by DFHZGRP. They are treated in the same way as other terminal control VTAM statistics.

A03_PRSS_NIB_COUNT
The number of active VTAM sessions when INQUIRE OPTCD=COUNTS was issued - this represents the number of persisting sessions.

A03_PRSS_INQUIRE_COUNT
The number of times DFHZGRP issues INQUIRE OPTCD=PERSIST. Each INQUIRE should be given about 400 sessions.

A03_PRSS_UNBIND_COUNT
The number of times CLSDST or TERMSESS were issued by DFHZGUB.

A03_PRSS_OPNDST_COUNT
The number of sessions that OPNDST RESTORE restored successfully.

A03_PRSS_ERROR_COUNT
The number of sessions with NIBUSER=tctte address, that VTAM failed to restore with OPNDST RESTORE. This occurs if VTAM operator commands are issued whilst DFHZGRP is in control and sessions are closed as a result.
Chapter 109. Web domain (WB)

The Web domain manages Web-related work processed within CICS. For further information regarding these objects see the CICS Business Transaction Services manual.

Web domain’s specific gates

Table 120 summarizes the Web domain’s specific gates. It shows the level-1 trace point IDs of the modules providing the functions for the gate, the functions provided by the gate, and whether or not the functions are available through the exit programming interface (XPI).

Table 120. Web domain’s specific gates

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>XPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>WBAP</td>
<td>WB 0300</td>
<td>START_BROWSE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>WB 0301</td>
<td>READ_NEXT</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>END_BROWSE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GET_MESSAGE_BODY</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GET_HTTP_RESPONSE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SEND_RESPONSE</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>READ_HEADER</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WRITE_HEADER</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INQUIRE</td>
<td>NO</td>
</tr>
<tr>
<td>WBAP</td>
<td>WB 0500</td>
<td>SEND</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>WB 0501</td>
<td>RECEIVE</td>
<td>NO</td>
</tr>
</tbody>
</table>

WBAP gate, START_BROWSE function

The START_BROWSE function starts a browse of the HTTP headers or the HTML forms data in an HTTP request.

Input parameters
None

Output parameters

RESPONSE is the domain’s response to the call. It can have any of these values:

OK | EXCEPTION | INVALID | DISASTER | KERNERROR | PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>INVALID_REQUEST_FORMAT NON_WEB_TRANSACTION HEADER_BROWSE_ACTIVE FORMFIELD_BROWSE_ACTIVE CLIENT_CODEPAGE_UNSUPP SERVER_CODEPAGE_UNSUPP NO_FORMS_DATA, INVALID_CODEPAGE_COMBIN</td>
</tr>
<tr>
<td>DISASTER</td>
<td>FORMFIELD_STRUCT_FORM_ERR FORMFIELD_CANNOT_GET_BODY FORMFIELD_STRUCT_CORRUPT FORMFIELD_CORRUPT_HEADER, FORMFIELD_NO_BOUNDARY_STR FORMFIELD_NO_CONTENT_HDR FORMFIELD_UNKNOWN_FORMTYPE NO_CONVERT_PARM</td>
</tr>
</tbody>
</table>
**WBAP gate, READ_NEXT function**

The READ_NEXT function returns the next HTTP header in a browse of HTTP headers.

**Input parameters**
None

**Output parameters**

- **HTTP_HEADER_BUFFER_NAME**
  returns the name of the next HTTP header

- **HTTP_HEADER_BUFFER_VALUE**
  returns the value of the next HTTP header

- **RESPONSE**
  is the domain’s response to the call. It can have any of these values:
  - OK
  - EXCEPTION
  - INVALID
  - DISASTER
  - KERNERROR
  - PURGED

  **[REASON]**
  is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
</table>
| EXCEPTION| BROWSE_END, NON_WEB_TRANSACTION, HEADER_BROWSE_NOT_ACTIVE, FORMFLD_BROWSE_NOT_ACTIVE, FORMFLD_NAME_LENGTH_ERROR, HEADER_VALUE_LENGTH_ERROR, INVALID_HEADER, FORMFLD_NAME_LENGTH_ERROR, NO/forms_DATA, INVALID_FORMFLD, FORMFIELD_STRUCT_CORRUPT, FORMFIELD_CORRUPT_HEADER, NO_CONVERT_PARM,
| DISASTER | FORMFIELD_STRUCT_CORRUPT, FORMFIELD_CORRUPT_HEADER, NO_CONVERT_PARM,                   |

**WBAP gate, END_BROWSE function**

The END_BROWSE function defines the end of a browse of the HTTP headers received for an HTTP request.

**Input parameters**
None

**Output parameters**

- **RESPONSE**
  is the domain’s response to the call. It can have any of these values:
  - OK
  - EXCEPTION
  - INVALID
  - DISASTER
  - KERNERROR
  - PURGED

  **[REASON]**
  is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NON_WEB_TRANSACTION, HEADER_BROWSE_NOT_ACTIVE FORMFLD_BROWSE_NOT_ACTIVE</td>
</tr>
</tbody>
</table>

**WBAP gate, GET_MESSAGE_BODY function**

The GET_MESSAGE_BODY function retrieves the previously constructed body of an HTTP response.
Input parameters

**DATA_BUFFER**
Buffer into which the data is to be placed

**TRUNCATE**
indicates whether or not data is to be truncated if the buffer is too small. It can have the following values:
YES|NO

**CLIENT_CODEPAGE**
ASCII Codepage into which the data is to be converted before being passed back to the caller

**SERVER_CODEPAGE**
EBCDIC Codepage of the data to be passed back

**CONVERT**
indicates whether or not data is to undergo codepage conversion. It can have the following values:
YES|NO

Output parameters

**SET_BLOCK**
Address of a block of storage containing the message body

**REQUEST_TYPE**
Indicates whether we are processing an HTTP Request It can have the following values:
HTTP|NON_HTTP

**RESPONSE**
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

**[REASON]**
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>CODEPAGE_NOT_FOUND, NON_WEB_TRANSACTION</td>
</tr>
</tbody>
</table>

**WBAP gate, GET_HTTP_RESPONSE function**
The GET_HTTP_RESPONSE function retrieves the HTTP Response which has been constructed by a Web API application program.

Input parameters

**DOCUMENT_TOKEN**
is the 8 byte field into which CICS places the document token identifying the document which contains the body of the HTTP response

Output parameters

**RESPONSE**
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

**[REASON]**
is returned when RESPONSE is EXCEPTION. Possible values are:
**WBAP gate, SEND_RESPONSE function**

The SEND_RESPONSE function identifies a CICS Document which is to be used as the body of a HTTP response, and the HTTP reason code with which that response is to be returned.

**Input parameters**

- **DOCUMENT_TOKEN**: identifies the CICS document to be used as the body of the HTTP response.
- **CLIENT_CODEPAGE**: identifies the ASCII codepage into which the body of the HTTP response is to be converted.
- **STATUS_CODE**: HTTP response code with which the HTTP response is returned.
- **STATUS_TEXT**: Text to accompany HTTP response code with which the HTTP response is returned.

**Output parameters**

- **RESPONSE**: is the domain’s response to the call. It can have any of these values: OK | EXCEPTION | INVALID | DISASTER | KERNERROR | PURGED

- **[REASON]**: is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NON_WEB_TRANSACTION, DOCUMENT_NOT_FOUND, CODEPAGE_NOT_FOUND</td>
</tr>
</tbody>
</table>

**WBAP gate, READ_HEADER function**

The READ_HEADER function returns the value of a specific HTTP request header.

**Input parameters**

- **HTTP_HEADER_BLOCK_VALUE**: Buffer containing name of the header for which a value is returned.
- **HTTP_HEADER_BLOCK_VALUE**: Buffer containing the value of the requested header.

**Output parameters**

- **RESPONSE**: is the domain’s response to the call. It can have any of these values: OK | EXCEPTION | INVALID | DISASTER | KERNERROR | PURGED

- **[REASON]**: is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>HEADER_NOT_FOUND, INVALID_REQUEST_FORMAT, NON_WEB_TRANSACTION</td>
</tr>
</tbody>
</table>
WBAP gate, WRITE_HEADER function
The WRITE_HEADER function causes a HTTP response header to be stored by CICS.

Input parameters
HTTP_HEADER_BLOCK_NAME
Buffer containing name of header
HTTP_HEADER_BLOCK_VALUE
Buffer containing value of header

Output parameters
RESPONSE
is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

REASON
is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NON_WEB_TRANSACTION</td>
</tr>
</tbody>
</table>

WBAP gate, INQUIRE function
The INQUIRE function passes back information pertaining to an HTTP request.

Input parameters
SERVER_NAME
Buffer to contain TCP/IP name of CICS

CLIENT_NAME
Buffer to contain TCP/IP name of client from which HTTP request was received.

HTTP_METHOD
Buffer to contain HTTP method specified on the HTTP request

HTTP_VERSION
Buffer to contain HTTP version specified on the HTTP request

QUERYSTRING
Buffer to contain HTTP query string specified on the HTTP request

URI
Buffer to contain URI specified on the HTTP request

Output parameters
CLIENT_ADDR
Fullword containing IP address of the client from which the HTTP request was received

SERVER_ADDR
Fullword containing IP address of the TCP/IP stack on which the HTTP request was received

SERVER_PORT
Fullword containing port number on which the HTTP request was received
Web Domain (WB)

CERTIFICATE_TOKEN
eight byte token identifying SSL certificate of client issuing this HTTP request

REQUEST_TYPE
Indicates whether CICS recognized the incoming data as a valid HTTP request. Can be set to:
HTTP|NON_HTTP

SSL_TYPE
Indicates what level of SSL support applies to the incoming HTTP request. Can be set to:
YES|NO|CLIENTAUTH

RESPONSE is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NON_WEB_TRANSACTION, INVALID_REQUEST_FORMAT</td>
</tr>
</tbody>
</table>

WBSR gate, RECEIVE function

The RECEIVE function receives an HTTP Request off a socket, and parses it in order to determine what to do with it.

Input parameters

INITIAL_RECEIVE
Indicates whether this is the first receive issued by the caller, Can be set any of these values:
YES|NO

Output parameters

TOKEN
Token uniquely identifying the WebRequestBlock associated with this HTTP request.

ATTACH_TRANSID
Transaction ID of Web alias transaction to be attached to continue processing the HTTP request.

FAILING_PROGRAM
Name of program which returned an error in the course of receiving the HTTP request.

CONNECTION_PERSIST
Indicates whether the HTTP Request included the HTTP 1.0 Keepalive header. Can be set to any of these values:
YES|NO

RESPONSE is the domain’s response to the call. It can have any of these values:
OK|EXCEPTION|INVALID|DISASTER|KERNERROR|PURGED
[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>NO_ANALYZER_SPECIFIED, ANALYZER_LINK_ERROR, ANALYZER_ERROR, ANALYZER_DATALENG_ERROR, HDR_LENGTH_ERROR, RECEIVE_ERROR, STORAGE_ERROR, CONNECTION_CLOSED, LOGIC_ERROR</td>
</tr>
</tbody>
</table>

**WBSR gate, SEND function**

The SEND function returns the response constructed following receipt of an HTTP request.

**Input parameters**

**TOKEN**

Token identifying WebRequestBlock with which this SEND is associated

**Output parameters**

**RESPONSE**

is the domain’s response to the call. It can have any of these values: OK | EXCEPTION | INVALID | DISASTER | KERNERROR | PURGED

[REASON] is returned when RESPONSE is EXCEPTION. Possible values are:

<table>
<thead>
<tr>
<th>RESPONSE</th>
<th>Possible REASON values</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION</td>
<td>STORAGE_ERROR, SEND_ERROR, CONNECTION_CLOSED, LOGIC_ERROR</td>
</tr>
</tbody>
</table>

**Web domain’s generic gates**

Table 121 summarizes the Web domain’s generic gates. It shows the level-1 trace point IDs of the modules providing the functions for the gates, the functions provided by the gates, and the generic formats for calls to the gates.

**Table 121. Web domain’s generic gates**

<table>
<thead>
<tr>
<th>Gate</th>
<th>Trace</th>
<th>Function</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMDM</td>
<td>WB 0100</td>
<td>INITIALIZE_DOMAIN</td>
<td>DMDM</td>
</tr>
<tr>
<td></td>
<td>WB 0101</td>
<td>QUIESCE_DOMAIN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TERMINATE_DOMAIN</td>
<td></td>
</tr>
<tr>
<td>XMAC</td>
<td>WB 0600</td>
<td>INIT_XM_CLIENT</td>
<td>XMAC</td>
</tr>
<tr>
<td></td>
<td>WB 0601</td>
<td>BIND_XM_CLIENT</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TRANSACTION_HANG</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>RELEASE_XM_CLIENT</td>
<td></td>
</tr>
</tbody>
</table>

For descriptions of these functions and their input and output parameters, refer to the §s dealing with the corresponding generic formats:
Web Domain (WB)

In initialization, quiesce, and termination processing, the Web domain performs only internal routines.

### Modules

<table>
<thead>
<tr>
<th>Module</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHWBDM</td>
<td>Handles the following requests: INITIALIZE_DOMAIN QUIESCE_DOMAIN TERMINATE_DOMAIN</td>
</tr>
<tr>
<td>DFHWBAP</td>
<td>Handles the following requests START_BROWSE READ_NEXT END_BROWSE GET_MESSAGE_BODY GET_HTTP_RESPONSE SEND_RESPONSE READ WRITE_HEADER INQUIRE</td>
</tr>
<tr>
<td>DFHWBAPF</td>
<td>Handles forms processing for: START_BROWSE READ_NEXT END_BROWSE READ</td>
</tr>
<tr>
<td>DFHWBSR</td>
<td>Handles the following requests SEND RECEIVE</td>
</tr>
<tr>
<td>DFHWBXM</td>
<td>Handles the following requests INIT_XM_CLIENT BIND_XM_CLIENT TRANSACTION_HANG RELEASE_XM_CLIENT</td>
</tr>
<tr>
<td>DFHWBQM</td>
<td>Domain subroutine which writes Web data to TS. Handles the following requests: PUT_QUEUE GET_QUEUE DELETE_QUEUE GET_TOKEN</td>
</tr>
</tbody>
</table>

### Exits

No global user exit points are provided in this domain.
The point IDs for the Web domain are of the form WB xxxx; the corresponding trace levels are WB 1, WB 2, and Exc.

For more information about the trace points, see the CICS Trace Entries. For more information about using traces in problem determination, see the CICS Problem Determination Guide.
Chapter 110. WTO and WTOR

Design overview

The DFHSUWT module provides the following support for executing MVS WTO and WTOR SVCs:

**SEND** supports Write To Operator (WTO):
- A single-line message up to 113 characters, or a multiline message consisting of a control line and up to nine lines of 69 characters
- Route code specification (route code list of 1 through 28 numbers, each in the range 1 through 28)
- Descriptor code specification (descriptor code list of 1 through 16 numbers, each in the range 1 through 16).

**CONVERSE** supports Write To Operator With Reply (WTOR):
- A single-line message up to 121 characters
- Route code specification (route code list of 1 through 28 numbers, each in the range 1 through 28)
- Descriptor code specification (descriptor code list of 1 through each in the range 1 through 28) 16 numbers, each in the range 1 through 16)
- A reply with maximum length of 119 characters.

The DFHWTO macro may be used to send a message, normally to the system operator, when neither the CICS message domain nor the old message program (DFHMGP) can be used. The message domain cannot be used during certain phases of initialization and XRF processing, because it requires a kernel stack environment. DFHMGP cannot be used during initialization, nor during any sort of abend or dump processing, because it uses task LIFO storage and may therefore invoke the storage control program.

The DFHWTO macro may also be used to terminate CICS abnormally or to request a reply from the operator.

Any WTO or WTOR macros that are issued by CICS might be intercepted by the console message handling facility described under "Console message handling" on page 1119. This service optionally inserts the CICS region’s applid into CICS messages before they are displayed on the console.

Modules

DFHSUWT and DFHWTO

Exits

No global user exit points are provided for this function.

Trace

The following point IDs are provided for this function:
- AP FF0x, for which the trace levels are AP 1 and Exc.
WTO and WTOR

For more information about the trace points, see the CICS Trace Entries. For more information about using traces in problem determination, see the CICS Problem Determination Guide.
Chapter 111. CICS Web Interface and CICS business logic interface

The CICS Web Interface allows Web browsers to use transaction processing services by calling CICS programs or by running CICS transactions. The Web browsers use TCP/IP to communicate with the CICS Web Interface.

The CICS business logic interface allows other external users to use transactions processing services.

Design overview

For information about the design and implementation of the CICS Web Interface and CICS business logic interface, see the CICS Internet Guide.

Control blocks

Figure 123 on page 1414 shows the control blocks used by the CICS Web Interface 3270 support.
The modules of the CICS Web Interface are organized as follows:
1. CICS Initialization
2. Web attach processing
3. Alias transaction
4. CICS business logic interface
5. CICS Web bridge
6. HTML template manager
7. Environment variables
CICS Initialization

DFHWBIP
DFHWBIP initializes the web environment at CICS startup.

Web attach processing

DFHWBXN
DFHWBXN is the Web attach processing module. It is the initial program invoked for transaction CWXN (or an alias of CWXN), which is attached for a new sockets connection received on a port associated with a Web TCPIPSERVICE.
- Calls Web domain WBSR gate to process the incoming data
- Attaches the Web alias transaction (default transaction ID is CWBA).

Alias transaction

DFHWBA
DFHWBA is the alias program. An alias transaction is started by Web attach processing for each request received from TCP/IP. It sets up a parameter list for DFHWBA1, and then calls it.

CICS business logic interface

DFHWBA1
DFHWBA1 is the CICS business logic interface program. The interface to the CICS business logic interface program is described in CICS Internet Guide.

The CICS business logic interface program is called by DFHWBA. If its parameters list specifies a user-replaceable converter, it calls the Decode function of the converter. If its parameter list specifies a CICS program, it calls the CICS program. If its parameters list specifies a user-replaceable converter, it calls the Encode function of the converter.

CICS Web bridge

DFHWBGB
DFHWBGB removes redundant state data from the system.

DFHWBST
DFHWBST manages the state data held for the CICS Web bridge.

DFHWBTC
DFHWBTC performs 3270/HTML conversion for the CICS Web bridge.

DFHWBTTA
DFHWBTTA sets up the parameters for bridging to transactions from the CICS Web Interface.

DFHWBLT
DFHWBLT is the CICS Web bridge exit.

HTML template manager

DFHWBTL
DFHWBTL is the HTML template manager. The interface to the template manager is described in CICS Internet Guide.
Environment variables

**DFHWBENV**
DFHWBENV is the environment variables program. The interface to the environment variables program is described in *CICS Internet Guide*.

**DFHWBPA**
DFHWBPA provides a parsing function for HTML forms and HTTP environment variable information built by DFHWBENV.

**DFHWBUN**
DFHWBUN provides an unescaping function for data which has been transmitted to CICS in its escaped form, but which the application needs to manipulate in its unescaped form.

---

Exits

No global user exit points are provided for this function.

---

Trace

The trace point IDs for this function are of the form WB xxxx. The trace levels are WB 1, WB 2, and Exc.

For more information about the trace points, see the [CICS Trace Entries](#). For more information about using traces in problem determination, see the [CICS Problem Determination Guide](#).
## Part 3. CICS modules

### Chapter 112. CICS directory
- Classification of elements: 1419
- Name: 1419
- Type: 1419
- Library: 1419

### Chapter 113. CICS executable modules
- Optional listings: 1420
- Contents of the distribution tapes: 1421

This part contains:

© Copyright IBM Corp. 1989, 2001
Chapter 112. CICS directory

This section lists, in alphanumeric order by element name, the contents of the distribution tapes listed in Table 122 on page 1420.

The list shows, for each element:
- The name of the element
- The type of element
- A description of the element
- The names of the source and object distribution libraries containing the element.

Classification of elements

Name

This is the name of the element in the distribution library.

Type

The types of elements are:

CSECT.

A control section or, in the case of a source element only, the first part of a control section (other source elements may be copied by the CSECT). Where an object module is OCO, this is indicated following the type CSECT; no source code is provided for modules thus classified.

DSECT.

A dummy section (or appropriate high-level language equivalent) defining a CICS data area.

Macro.

A macro definition.

Source.

Source code that is not a CSECT.

Sample.

Sample tables, programs, map sets, partition sets, or data files.

Symbolic.

A definition (with no DSECT statement) of a CICS data area, or a group of EQU statements that symbolically define values used throughout a program.

Other.

Job control language statements or cataloged procedures. See the CICS Transaction Server for z/OS Installation Guide and the CICS System Definition Guide for the handling of these elements.

Library

Two columns are given under the heading Library. These correspond to source code and object code distribution respectively. The distribution tapes are in SMP/E RELFILE format, and a RELFILE number indicates the position of each data set on a particular tape. For further details about the format of the distribution tapes, see the CICS Transaction Server for z/OS Program Directory.
Some elements have several COBOL, PL/I, C/370, and assembler-language versions with the same name; these elements are shown here as cataloged in more than one source distribution library.

The meanings of the letters in the library columns is given in Table 122.

Table 122. CICS Transaction Server for z/OS, Version 2 Release 1 distribution tapes

<table>
<thead>
<tr>
<th>Letter</th>
<th>Tape volser</th>
<th>File name</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>02</td>
<td>CI6100</td>
<td>HCI6100.F2</td>
<td>CICSTS21.CICS.ADFHINST</td>
</tr>
<tr>
<td>03</td>
<td>CI6100</td>
<td>HCI6100.F3</td>
<td>CICSTS21.CICS.ADFHMOD *</td>
</tr>
<tr>
<td>04</td>
<td>CI6100</td>
<td>HCI6100.F4</td>
<td>CICSTS21.CICS.ADFHAPD1</td>
</tr>
<tr>
<td>05</td>
<td>CI6100</td>
<td>HCI6100.F5</td>
<td>CICSTS21.CICS.ADFHAPD2</td>
</tr>
<tr>
<td>06</td>
<td>CI6100</td>
<td>HCI6100.F6</td>
<td>CICSTS21.CICS.ADFHCLIB</td>
</tr>
<tr>
<td>07</td>
<td>CI6100</td>
<td>HCI6100.F7</td>
<td>CICSTS21.CICS.ADFHCOB</td>
</tr>
<tr>
<td>08</td>
<td>CI6100</td>
<td>HCI6100.F8</td>
<td>CICSTS21.CICS.ADFHAC370</td>
</tr>
<tr>
<td>09</td>
<td>CI6100</td>
<td>HCI6100.F9</td>
<td>CICSTS21.CICS.ADFHENV</td>
</tr>
<tr>
<td>10</td>
<td>CI6100</td>
<td>HCI6100.F10</td>
<td>CICSTS21.CICS.ADFHLANG</td>
</tr>
<tr>
<td>11</td>
<td>CI6100</td>
<td>HCI6100.F11</td>
<td>CICSTS21.CICS.ADFHMAC</td>
</tr>
<tr>
<td>12</td>
<td>CI6100</td>
<td>HCI6100.F12</td>
<td>CICSTS21.CICS.ADFHMLIB</td>
</tr>
<tr>
<td>13</td>
<td>CI6100</td>
<td>HCI6100.F13</td>
<td>CICSTS21.CICS.ADFHMSGS</td>
</tr>
<tr>
<td>14</td>
<td>CI6100</td>
<td>HCI6100.F14</td>
<td>CICSTS21.CICS.ADFHMSRC</td>
</tr>
<tr>
<td>15</td>
<td>CI6100</td>
<td>HCI6100.F15</td>
<td>CICSTS21.CICS.ADFHPARM</td>
</tr>
<tr>
<td>16</td>
<td>CI6100</td>
<td>HCI6100.F16</td>
<td>CICSTS21.CICS.ADFHPARM</td>
</tr>
<tr>
<td>17</td>
<td>CI6100</td>
<td>HCI6100.F17</td>
<td>CICSTS21.CICS.ADFHPL1</td>
</tr>
<tr>
<td>18</td>
<td>CI6100</td>
<td>HCI6100.F18</td>
<td>CICSTS21.CICS.ADFHPROC</td>
</tr>
<tr>
<td>19</td>
<td>CI6100</td>
<td>HCI6100.F19</td>
<td>CICSTS21.CICS.ADFHSAMP</td>
</tr>
<tr>
<td>20</td>
<td>CI6100</td>
<td>HCI6100.F20</td>
<td>CICSTS21.CICS.ADFHSACK</td>
</tr>
<tr>
<td>C2</td>
<td>CI6100</td>
<td>JC6101.F1</td>
<td>CICSTS21.CICS.ADFHC0B</td>
</tr>
<tr>
<td>C3</td>
<td>CI6100</td>
<td>JC6101.F2</td>
<td>COBOL elements of CICSTS21.CICS.ADFHINST</td>
</tr>
<tr>
<td>C4</td>
<td>CI6100</td>
<td>JC6101.F2</td>
<td>COBOL elements of CICSTS21.CICS.ADFHINST</td>
</tr>
<tr>
<td>P2</td>
<td>CI6100</td>
<td>JC6102.F1</td>
<td>CICSTS21.CICS.ADFHPL1</td>
</tr>
<tr>
<td>P3</td>
<td>CI6100</td>
<td>JC6102.F2</td>
<td>PL/I elements of CICSTS21.CICS.ADFHINST</td>
</tr>
<tr>
<td>D2</td>
<td>CI6100</td>
<td>JC6103.F1</td>
<td>CICSTS21.CICS.ADFHC370</td>
</tr>
<tr>
<td>D3</td>
<td>CI6100</td>
<td>JC6103.F2</td>
<td>C/370 elements of CICSTS21.CICS.ADFHINST</td>
</tr>
<tr>
<td>OS</td>
<td>CI6105</td>
<td>CICSTS21.CICS.OPTSRC01 -</td>
<td></td>
</tr>
</tbody>
</table>

An asterisk (*) following the RELFILE number indicates that the distribution library contains object modules.

**Note:** Object modules only are supplied for the Japanese language feature; corresponding source code is not provided for these modules.

### Optional listings

Assembled listings of programs and source listings of macros, DSECTs, and symbolic definitions are available with CICS, and can be supplied on CD-ROM or microfiche. For further information about the optional listings, see the CICS Transaction Server for z/OS Program Directory.
### Contents of the distribution tapes

**Table 123. CICS modules directory**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCTINDX</td>
<td>Sample</td>
<td>Primer - batch index file recovery - COBOL</td>
<td>C3</td>
</tr>
<tr>
<td>ACCTREC</td>
<td>Sample</td>
<td>Primer - account record - COBOL</td>
<td>C3</td>
</tr>
<tr>
<td>ACCTSET</td>
<td>Sample</td>
<td>Primer - map set - COBOL</td>
<td>19</td>
</tr>
<tr>
<td>ACCT00</td>
<td>Sample</td>
<td>Primer - menu display - COBOL</td>
<td>C3</td>
</tr>
<tr>
<td>ACCT01</td>
<td>Sample</td>
<td>Primer - initial request processing - COBOL</td>
<td>C3</td>
</tr>
<tr>
<td>ACCT02</td>
<td>Sample</td>
<td>Primer - update processing - COBOL</td>
<td>C3</td>
</tr>
<tr>
<td>ACCT03</td>
<td>Sample</td>
<td>Primer - requests for printing - COBOL</td>
<td>C3</td>
</tr>
<tr>
<td>ACCT04</td>
<td>Sample</td>
<td>Primer - error processing - COBOL</td>
<td>C3</td>
</tr>
<tr>
<td>ACIXREC</td>
<td>Sample</td>
<td>Primer - index record - COBOL</td>
<td>C3</td>
</tr>
<tr>
<td>ACMF</td>
<td>CSECT</td>
<td>Buffer management routine</td>
<td>-</td>
</tr>
<tr>
<td>AXME</td>
<td>CSECT</td>
<td>Server task error recovery</td>
<td>-</td>
</tr>
<tr>
<td>AXEV</td>
<td>CSECT</td>
<td>Event control and task management routine</td>
<td>-</td>
</tr>
<tr>
<td>AXMEV1</td>
<td>CSECT</td>
<td>Event management MVS POST exit</td>
<td>-</td>
</tr>
<tr>
<td>AXMLE</td>
<td>CSECT</td>
<td>Sequential file I/O routine</td>
<td>-</td>
</tr>
<tr>
<td>AXMHP</td>
<td>CSECT</td>
<td>Heap storage routine</td>
<td>-</td>
</tr>
<tr>
<td>AXMHS</td>
<td>CSECT</td>
<td>Hash value generation subroutine</td>
<td>-</td>
</tr>
<tr>
<td>AXMLF</td>
<td>CSECT</td>
<td>Server environment LIFO storage routine</td>
<td>-</td>
</tr>
<tr>
<td>AXMFLMVS</td>
<td>CSECT</td>
<td>LIFO storage routine - MVS batch version</td>
<td>-</td>
</tr>
<tr>
<td>AXMLK</td>
<td>CSECT</td>
<td>Lock management routine</td>
<td>-</td>
</tr>
<tr>
<td>AXMM</td>
<td>CSECT</td>
<td>Message editing and processing routine</td>
<td>-</td>
</tr>
<tr>
<td>AXMMSTAB</td>
<td>CSECT</td>
<td>Message filtering table</td>
<td>-</td>
</tr>
<tr>
<td>AXMP</td>
<td>CSECT</td>
<td>Operator communication routine</td>
<td>-</td>
</tr>
<tr>
<td>AXMOD</td>
<td>CSECT</td>
<td>Server operating system interface</td>
<td>-</td>
</tr>
<tr>
<td>AXMOP</td>
<td>CSECT</td>
<td>Page storage routine</td>
<td>-</td>
</tr>
<tr>
<td>AXMM</td>
<td>CSECT</td>
<td>Resource manager initialization/termination</td>
<td>-</td>
</tr>
<tr>
<td>AXMS</td>
<td>CSECT</td>
<td>Resource tracking routine</td>
<td>-</td>
</tr>
<tr>
<td>AXMSC</td>
<td>CSECT</td>
<td>Server connection routine</td>
<td>-</td>
</tr>
<tr>
<td>AXMSC1</td>
<td>CSECT</td>
<td>Locate server connection system area</td>
<td>-</td>
</tr>
<tr>
<td>AXMSC2</td>
<td>CSECT</td>
<td>Server connection services interface</td>
<td>-</td>
</tr>
<tr>
<td>AXMSI</td>
<td>CSECT</td>
<td>Subsystem initialization routine</td>
<td>-</td>
</tr>
<tr>
<td>AXMTI</td>
<td>CSECT</td>
<td>Timer interval service</td>
<td>-</td>
</tr>
<tr>
<td>AXMTK</td>
<td>CSECT</td>
<td>Task attach and detach routine</td>
<td>-</td>
</tr>
<tr>
<td>AXMTM</td>
<td>CSECT</td>
<td>Mode-independent time and date service</td>
<td>-</td>
</tr>
<tr>
<td>AXMTR</td>
<td>CSECT</td>
<td>Server trace management routine</td>
<td>-</td>
</tr>
<tr>
<td>AXMWS</td>
<td>CSECT</td>
<td>Variable sized shared storage routine</td>
<td>-</td>
</tr>
<tr>
<td>AXMWH</td>
<td>CSECT</td>
<td>AXMWH - data areas</td>
<td>-</td>
</tr>
<tr>
<td>AXMT</td>
<td>CSECT</td>
<td>AXMTT - data areas</td>
<td>-</td>
</tr>
<tr>
<td>AXMM</td>
<td>CSECT</td>
<td>Cross memory interface</td>
<td>-</td>
</tr>
<tr>
<td>AXMM1</td>
<td>CSECT</td>
<td>Cross memory interface POST module</td>
<td>-</td>
</tr>
<tr>
<td>CALLDLI</td>
<td>Macro</td>
<td>CALL DL/I services</td>
<td>11</td>
</tr>
<tr>
<td>CAUDBL</td>
<td>CSECT</td>
<td>CAU builder front end</td>
<td>-</td>
</tr>
<tr>
<td>CAUDBLDIN</td>
<td>CSECT</td>
<td>CAU builder input processor</td>
<td>-</td>
</tr>
<tr>
<td>CAUDBLDMR</td>
<td>CSECT</td>
<td>CAU builder merge processor</td>
<td>-</td>
</tr>
<tr>
<td>CAUDBLDOT</td>
<td>CSECT</td>
<td>CAU builder output processor</td>
<td>-</td>
</tr>
<tr>
<td>CAUCAFBE</td>
<td>CSECT</td>
<td>CAU CAFB abend exit</td>
<td>-</td>
</tr>
<tr>
<td>CAUCAFBI</td>
<td>CSECT</td>
<td>CAU CAFB main program</td>
<td>-</td>
</tr>
<tr>
<td>CAUCAFBJ</td>
<td>CSECT</td>
<td>CAU CAFB data save program</td>
<td>-</td>
</tr>
<tr>
<td>CAUCAFDT</td>
<td>CSECT</td>
<td>CAU CAF date utility</td>
<td>-</td>
</tr>
<tr>
<td>CAUCAFEE</td>
<td>CSECT</td>
<td>CAU CAF abend exit</td>
<td>-</td>
</tr>
<tr>
<td>CAUCAFFE</td>
<td>CSECT</td>
<td>CAU CAF main program</td>
<td>-</td>
</tr>
<tr>
<td>CAUCAFF2</td>
<td>CSECT</td>
<td>CAU CAF options</td>
<td>-</td>
</tr>
<tr>
<td>CAUCAFF3</td>
<td>CSECT</td>
<td>CAU CAF start program</td>
<td>-</td>
</tr>
<tr>
<td>CAUCAFF4</td>
<td>CSECT</td>
<td>CAU CAF stop program</td>
<td>-</td>
</tr>
<tr>
<td>CAUCAFF5</td>
<td>CSECT</td>
<td>CAU CAF pause program</td>
<td>-</td>
</tr>
<tr>
<td>CAUCAFF6</td>
<td>CSECT</td>
<td>CAU CAF continue program</td>
<td>-</td>
</tr>
<tr>
<td>CAUCAFF7</td>
<td>CSECT</td>
<td>CAU CAF help program</td>
<td>-</td>
</tr>
</tbody>
</table>
### CICS directory

#### Table 123. CICS modules directory (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAUCAFDP</td>
<td>CSECT</td>
<td>CAU CAFB request handler</td>
<td>- 03</td>
</tr>
<tr>
<td>CAULCLBL</td>
<td>Sample</td>
<td>Sample JCL for running CAU builder</td>
<td>02 -</td>
</tr>
<tr>
<td>CAUCLCCA</td>
<td>Sample</td>
<td>Sample JCL for CAU Affinity data files</td>
<td>02 -</td>
</tr>
<tr>
<td>CAUCLCC</td>
<td>Sample</td>
<td>Sample JCL for CAU Affinity control file</td>
<td>02 -</td>
</tr>
<tr>
<td>CAUJCLLD</td>
<td>Sample</td>
<td>Sample JCL for running CAU scanner (Detail mode)</td>
<td>02 -</td>
</tr>
<tr>
<td>CAUJCLLS</td>
<td>Sample</td>
<td>Sample JCL for running CAU scanner (Summary mode)</td>
<td>02 -</td>
</tr>
<tr>
<td>CAUJCLRP</td>
<td>Sample</td>
<td>Sample JCL for running CAU Reporter</td>
<td>02 -</td>
</tr>
<tr>
<td>CAULMS</td>
<td>CSECT</td>
<td>CAU load module scanner</td>
<td>- 03</td>
</tr>
<tr>
<td>CAUMAPI</td>
<td>CSECT</td>
<td>CAU BMS map CAFF01</td>
<td>- 03</td>
</tr>
<tr>
<td>CAUMAPIU</td>
<td>CSECT</td>
<td>CAU BMS map CAFF01</td>
<td>19</td>
</tr>
<tr>
<td>CAUMAP2</td>
<td>CSECT</td>
<td>CAU BMS map CAFF02</td>
<td>- 03</td>
</tr>
<tr>
<td>CAUMAP2U</td>
<td>CSECT</td>
<td>CAU BMS map CAFF02</td>
<td>19</td>
</tr>
<tr>
<td>CAUMAP3</td>
<td>CSECT</td>
<td>CAU BMS map CAFFH1</td>
<td>- 03</td>
</tr>
<tr>
<td>CAUMAP4</td>
<td>CSECT</td>
<td>CAU BMS map CAFFH2</td>
<td>- 03</td>
</tr>
<tr>
<td>CAUMSCS</td>
<td>CSECT</td>
<td>CAU message manager CICS stub</td>
<td>- 03</td>
</tr>
<tr>
<td>CAUMSCMM</td>
<td>CSECT</td>
<td>CAU message manager</td>
<td>- 03</td>
</tr>
<tr>
<td>CAUMSGTB</td>
<td>CSECT</td>
<td>CAU message table</td>
<td>- 03</td>
</tr>
<tr>
<td>CAUREP</td>
<td>CSECT</td>
<td>CAU reporter main module</td>
<td>- 03</td>
</tr>
<tr>
<td>CAUREPFM</td>
<td>CSECT</td>
<td>CAU reporter file manager</td>
<td>- 03</td>
</tr>
<tr>
<td>CAUREPMM</td>
<td>CSECT</td>
<td>CAU reporter print manager</td>
<td>- 03</td>
</tr>
<tr>
<td>CAUREPRM</td>
<td>CSECT</td>
<td>CAU reporter report manager</td>
<td>- 03</td>
</tr>
<tr>
<td>CAUTABM</td>
<td>CSECT</td>
<td>CAU detector table manager</td>
<td>- 03</td>
</tr>
<tr>
<td>CAUTABS</td>
<td>CSECT</td>
<td>CAU detector table storage manager</td>
<td>- 03</td>
</tr>
<tr>
<td>CAUXDUMM</td>
<td>CSECT</td>
<td>CAU detector dummy exit</td>
<td>- 03</td>
</tr>
<tr>
<td>CAUITBA</td>
<td>CSECT</td>
<td>CAU detector BAM process exit</td>
<td>- 03</td>
</tr>
<tr>
<td>CAUITBB</td>
<td>CSECT</td>
<td>CAU detector BAM activity exit</td>
<td>- 03</td>
</tr>
<tr>
<td>CAUITIB1</td>
<td>CSECT</td>
<td>CAU detector XBADEACT exit</td>
<td>- 03</td>
</tr>
<tr>
<td>CAUITIR</td>
<td>CSECT</td>
<td>CAU detector pseudo-conv end exit</td>
<td>- 03</td>
</tr>
<tr>
<td>CAUITI1</td>
<td>CSECT</td>
<td>CAU detector TRUE</td>
<td>- 03</td>
</tr>
<tr>
<td>CAUITI1M</td>
<td>CSECT</td>
<td>CAU detector logoff exit</td>
<td>- 03</td>
</tr>
<tr>
<td>CAUITI2M</td>
<td>CSECT</td>
<td>CAU detector signoff exit</td>
<td>- 03</td>
</tr>
<tr>
<td>CAUITI3M</td>
<td>CSECT</td>
<td>CAU detector XMEOUT exit</td>
<td>- 03</td>
</tr>
<tr>
<td>CAUITIOA</td>
<td>CSECT</td>
<td>CAU detector ADDRESS exit</td>
<td>- 03</td>
</tr>
<tr>
<td>CAUITIOC</td>
<td>CSECT</td>
<td>CAU detector CANCEL exit</td>
<td>- 03</td>
</tr>
<tr>
<td>CAUITIOE</td>
<td>CSECT</td>
<td>CAU detector ENQ/DEQ exit</td>
<td>- 03</td>
</tr>
<tr>
<td>CAUITIOG</td>
<td>CSECT</td>
<td>CAU detector GETMAIN exit</td>
<td>- 03</td>
</tr>
<tr>
<td>CAUITILO</td>
<td>CSECT</td>
<td>CAU detector LOAD/RELEASE exit</td>
<td>- 03</td>
</tr>
<tr>
<td>CAUITIQ</td>
<td>CSECT</td>
<td>CAU detector TS exit</td>
<td>- 03</td>
</tr>
<tr>
<td>CAUITOR</td>
<td>CSECT</td>
<td>CAU detector RETRIEVE exit</td>
<td>- 03</td>
</tr>
<tr>
<td>CAUITOS</td>
<td>CSECT</td>
<td>CAU detector SPI exit</td>
<td>- 03</td>
</tr>
<tr>
<td>CAUITOW</td>
<td>CSECT</td>
<td>CAU detector WAIT exit</td>
<td>- 03</td>
</tr>
<tr>
<td>CAUITOY</td>
<td>CSECT</td>
<td>CAU detector LOAD/FREEMAIN exit</td>
<td>- 03</td>
</tr>
<tr>
<td>CAUITO1</td>
<td>CSECT</td>
<td>CAU detector XEIOUT exit</td>
<td>- 03</td>
</tr>
<tr>
<td>CAUITXX</td>
<td>CSECT</td>
<td>CAU detector ICE expiry exit</td>
<td>- 03</td>
</tr>
<tr>
<td>CAUXITX1</td>
<td>CSECT</td>
<td>CAU detector XICEXP exit</td>
<td>- 03</td>
</tr>
<tr>
<td>CMC</td>
<td>Symbolic</td>
<td>SAA communications pseudonyms for C</td>
<td>02 -</td>
</tr>
<tr>
<td>CMCOBOL</td>
<td>Symbolic</td>
<td>SAA communications pseudonyms for COBOL</td>
<td>02 -</td>
</tr>
<tr>
<td>CMHASM</td>
<td>Symbolic</td>
<td>SAA communications pseudonyms for assembler</td>
<td>02 -</td>
</tr>
<tr>
<td>CMPLI</td>
<td>Symbolic</td>
<td>SAA communications pseudonyms for PL/I</td>
<td>02 -</td>
</tr>
<tr>
<td>DFHABAB</td>
<td>CSECT</td>
<td>AP domain abend handling</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHABABA</td>
<td>DSECT</td>
<td>ABAB parameter list</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHABABM</td>
<td>Macro</td>
<td>ABAB request</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHABABT</td>
<td>CSECT</td>
<td>ABAB trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHABEND</td>
<td>Macro</td>
<td>Issue an ABEND macro</td>
<td>- 03</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
<td>Library</td>
</tr>
<tr>
<td>------------</td>
<td>----------</td>
<td>------------------------------------------------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>DFHABREV</td>
<td>CSECT</td>
<td>String abbreviation checker</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHACP</td>
<td>CSECT</td>
<td>Abnormal condition program</td>
<td>OS 05</td>
</tr>
<tr>
<td>DFHACPTB</td>
<td>Macro</td>
<td>ACP abend table</td>
<td>OS</td>
</tr>
<tr>
<td>DFHADINS</td>
<td>CSECT</td>
<td>AD EJB CICS resource definitions</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHADJAR</td>
<td>CSECT</td>
<td>AD JAR to DJAR mapping</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHADSTR</td>
<td>CSECT</td>
<td>AD JAR to DJAR mapping</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHADUR</td>
<td>CSECT</td>
<td></td>
<td>- 03</td>
</tr>
<tr>
<td>DFHADURM</td>
<td>Sample</td>
<td>Sample URM to set CICS user id (C version)</td>
<td>19 -</td>
</tr>
<tr>
<td>DFHACFD</td>
<td>Macro</td>
<td>Authorized function control block (AFCB)</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHACFS</td>
<td>Macro</td>
<td>Authorized function common storage anchor</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHAIID</td>
<td>Macro</td>
<td>Application interface control block</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHAIICB</td>
<td>Macro</td>
<td>Application interface control block</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHAIICBP</td>
<td>CSECT</td>
<td>Application interface control block module</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHAIID</td>
<td>Symbolic</td>
<td>3270 attention identifiers</td>
<td>11 07</td>
</tr>
<tr>
<td>DFHAIID</td>
<td>Symbolic</td>
<td>3270 attention identifiers - COBOL</td>
<td>C2 -</td>
</tr>
<tr>
<td>DFHAIID</td>
<td>Symbolic</td>
<td>3270 attention identifiers - PL/I</td>
<td>P2 -</td>
</tr>
<tr>
<td>DFHAIID</td>
<td>Symbolic</td>
<td>3270 attention identifiers - C/370</td>
<td>O2 -</td>
</tr>
<tr>
<td>DFHAIIDDS</td>
<td>DSECT</td>
<td>Automatic initiate descriptor</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHAIIDUF</td>
<td>CSECT</td>
<td>Autoinstall terminal model manager (AITMM)</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHAIINAI</td>
<td>DSECT</td>
<td>AIIN parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHAIINM</td>
<td>Macro</td>
<td>AIIN request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHAIINT</td>
<td>CSECT</td>
<td>AIIN trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHAIIN1</td>
<td>CSECT</td>
<td>AITMM - initialization management program</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHAIIN2</td>
<td>CSECT</td>
<td>AITMM - initialization subtask program</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHAIIQ</td>
<td>CSECT</td>
<td>AITMM - locate/unlock/inquire/browse</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHAIIOQA</td>
<td>DSECT</td>
<td>AIIQ parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHAIIOQM</td>
<td>Macro</td>
<td>AIIQ request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHAIIOQT</td>
<td>CSECT</td>
<td>AIIQ trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHAIIRP</td>
<td>CSECT</td>
<td>AITMM - initialization/recovery</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHAIIRPA</td>
<td>DSECT</td>
<td>AIRP parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHAIIRPM</td>
<td>Macro</td>
<td>AIRP request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHAIIRPT</td>
<td>CSECT</td>
<td>AIRP trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHAIITDS</td>
<td>DSECT</td>
<td>AITMM - static storage</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHAIITM</td>
<td>CSECT</td>
<td>AITMM - add replace/delete</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHAIITMA</td>
<td>DSECT</td>
<td>AITM parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHAIITMM</td>
<td>Macro</td>
<td>AITM request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHAIITMT</td>
<td>CSECT</td>
<td>AITM trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHALP</td>
<td>CSECT</td>
<td>Terminal allocation</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHALRC</td>
<td>CSECT</td>
<td>Automatic initiate descriptor recovery</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHALXM</td>
<td>CSECT</td>
<td>AL XM transaction attach</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHAM</td>
<td>Macro</td>
<td>Address mode switching macro</td>
<td>- 11</td>
</tr>
<tr>
<td>DFHAMBDA</td>
<td>CSECT</td>
<td>RDO install of Process type resources</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHAMCSD</td>
<td>CSECT</td>
<td>RDO command logger</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHAMDH</td>
<td>CSECT</td>
<td>RDO install of Document resources</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHAMD2</td>
<td>CSECT</td>
<td></td>
<td>- 03</td>
</tr>
<tr>
<td>DFHAMEJ</td>
<td>CSECT</td>
<td>RDO install of EJB objects</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHAMER</td>
<td>CSECT</td>
<td>RDO error message builder</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHAMFC</td>
<td>CSECT</td>
<td>RDO install for FCT resources</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHAMGL</td>
<td>CSECT</td>
<td>RDO list generator</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHAMLM</td>
<td>CSECT</td>
<td>Program to install log manager objects</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHAMNQ</td>
<td>CSECT</td>
<td>RDO install of Enq model resources</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHAMOP</td>
<td>CSECT</td>
<td>RDO install of Request model resources</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHAMPAB</td>
<td>CSECT</td>
<td>RDO AMP error handler</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHAMPAD</td>
<td>CSECT</td>
<td>RDO add command</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHAMPAP</td>
<td>CSECT</td>
<td>RDO append command</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHAMPC</td>
<td>CSECT</td>
<td>RDO check command</td>
<td>- 03</td>
</tr>
</tbody>
</table>
### CICS directory

#### Table 123. CICS modules directory (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHAMPCO</td>
<td>CSECT</td>
<td>RDO copy and rename commands</td>
<td>03</td>
</tr>
<tr>
<td>DFHAMPC1</td>
<td>CSECT</td>
<td>SPI generic names match</td>
<td>03</td>
</tr>
<tr>
<td>DFHAMPC2</td>
<td>CSECT</td>
<td>SPI check list name &amp; produce list of groups</td>
<td>03</td>
</tr>
<tr>
<td>DFHAMPC3</td>
<td>CSECT</td>
<td>SPI diagnose duplicate objects</td>
<td>03</td>
</tr>
<tr>
<td>DFHAMPDF</td>
<td>CSECT</td>
<td>RDO define/redefine command</td>
<td>0S 03</td>
</tr>
<tr>
<td>DFHAMPD</td>
<td>CSECT</td>
<td>RDO display command</td>
<td>0S 03</td>
</tr>
<tr>
<td>DFHAMPD1</td>
<td>CSECT</td>
<td>RDO delete/remove commands</td>
<td>0S 03</td>
</tr>
<tr>
<td>DFHAMPEN</td>
<td>CSECT</td>
<td>RDO end AMP handler</td>
<td>0S 03</td>
</tr>
<tr>
<td>DFHAMPEX</td>
<td>CSECT</td>
<td>RDO expand command</td>
<td>0S 03</td>
</tr>
<tr>
<td>DFHAMPFI</td>
<td>CSECT</td>
<td>RDO begin AMP handler</td>
<td>0S 03</td>
</tr>
<tr>
<td>DFHAMPG</td>
<td>CSECT</td>
<td>RDO install of PG resources</td>
<td>03</td>
</tr>
<tr>
<td>DFHAMPL</td>
<td>CSECT</td>
<td>RDO install command</td>
<td>0S 03</td>
</tr>
<tr>
<td>DFHAMPN</td>
<td>CSECT</td>
<td>RDO install for partner resources</td>
<td>0S 03</td>
</tr>
<tr>
<td>DFHAMPVW</td>
<td>CSECT</td>
<td>RDO view command</td>
<td>0S 03</td>
</tr>
<tr>
<td>DFHAMP80</td>
<td>CSECT</td>
<td>RDO allocation manager (DFHAM)</td>
<td>0S 03</td>
</tr>
<tr>
<td>DFHAMRDI</td>
<td>CSECT</td>
<td>RDO install logger</td>
<td>0S 03</td>
</tr>
<tr>
<td>DFHAMSO</td>
<td>CSECT</td>
<td>RDO set name/type/set/stype from arg list</td>
<td>0S 03</td>
</tr>
<tr>
<td>DFHAMS0</td>
<td>CSECT</td>
<td>RDO install of TCP/IP services</td>
<td>03</td>
</tr>
<tr>
<td>DFHAMST</td>
<td>CSECT</td>
<td>RDO update time and date in arg list</td>
<td>0S 03</td>
</tr>
<tr>
<td>DFHAMTD</td>
<td>CSECT</td>
<td>Program to install Transient Data objects</td>
<td>03</td>
</tr>
<tr>
<td>DFHAMTP</td>
<td>CSECT</td>
<td>RDO AMP request processor</td>
<td>0S 03</td>
</tr>
<tr>
<td>DFHAMTS</td>
<td>CSECT</td>
<td>RDO install of Tsmodel resources</td>
<td>03</td>
</tr>
<tr>
<td>DFHAMXM</td>
<td>CSECT</td>
<td>Install XM domain resources (transaction and tranclass objects)</td>
<td>03</td>
</tr>
<tr>
<td>DFHANAT</td>
<td>Macro</td>
<td>3270 attribute character resolution</td>
<td>11</td>
</tr>
<tr>
<td>DFHANRC</td>
<td>Macro</td>
<td>3270 control character resolution</td>
<td>11</td>
</tr>
<tr>
<td>DFHAPAC</td>
<td>DSECT</td>
<td>AP domain abnormal condition reporting</td>
<td>03</td>
</tr>
<tr>
<td>DFHAPACA</td>
<td>DSECT</td>
<td>APAC parameter list</td>
<td>0S</td>
</tr>
<tr>
<td>DFHAPCM</td>
<td>Macro</td>
<td>APAC request</td>
<td>0S</td>
</tr>
<tr>
<td>DFHAPACT</td>
<td>CSECT</td>
<td>APAP translate table</td>
<td>03</td>
</tr>
<tr>
<td>DFHAPAPA</td>
<td>DSECT</td>
<td>APAP parameter list</td>
<td>0S</td>
</tr>
<tr>
<td>DFHAPAM</td>
<td>Macro</td>
<td>APAP request</td>
<td>0S</td>
</tr>
<tr>
<td>DFHAPAPT</td>
<td>CSECT</td>
<td>APAP trace interpretation data</td>
<td>0S 03</td>
</tr>
<tr>
<td>DFHAPATT</td>
<td>CSECT</td>
<td>AP domain - entrypoint attach</td>
<td>03</td>
</tr>
<tr>
<td>DFHAPCT</td>
<td>CSECT</td>
<td>-</td>
<td>03</td>
</tr>
<tr>
<td>DFHAPDS</td>
<td>DSECT</td>
<td>DFHAPDM static storage</td>
<td>0S</td>
</tr>
<tr>
<td>DFHAPDM</td>
<td>CSECT</td>
<td>AP domain - initialization/termination</td>
<td>03</td>
</tr>
<tr>
<td>DFHAPDN</td>
<td>CSECT</td>
<td>AP domain - transaction definition notify</td>
<td>03</td>
</tr>
<tr>
<td>DFHAPDUF</td>
<td>CSECT (OCO)</td>
<td>AP domain - formatted dump print</td>
<td>03</td>
</tr>
<tr>
<td>DFHAPF1</td>
<td>Macro</td>
<td>AP domain - environment initialization</td>
<td>0S</td>
</tr>
<tr>
<td>DFHAPF2</td>
<td>CSECT</td>
<td>AP domain - user exit service</td>
<td>03</td>
</tr>
<tr>
<td>DFHAPF3</td>
<td>DSECT</td>
<td>APEX parameter list</td>
<td>0S</td>
</tr>
<tr>
<td>DFHAPF4</td>
<td>Macro</td>
<td>APEX request</td>
<td>0S</td>
</tr>
<tr>
<td>DFHAPF5</td>
<td>CSECT</td>
<td>APEX trace interpretation data</td>
<td>0S 03</td>
</tr>
<tr>
<td>DFHAPF6</td>
<td>CSECT</td>
<td>-</td>
<td>03</td>
</tr>
<tr>
<td>DFHAPF7</td>
<td>CSECT</td>
<td>Java hotpooling runtime options</td>
<td>0H 08</td>
</tr>
<tr>
<td>DFHAPID</td>
<td>DSECT</td>
<td>Inquire on AP data</td>
<td>03</td>
</tr>
<tr>
<td>DFHAPIS</td>
<td>DSECT</td>
<td>Interval control static storage</td>
<td>0S</td>
</tr>
<tr>
<td>DFHAPIDT</td>
<td>DSECT</td>
<td>-</td>
<td>03</td>
</tr>
<tr>
<td>DFHAPIN</td>
<td>CSECT</td>
<td>AP domain - special initialization for programs</td>
<td>0S 03</td>
</tr>
<tr>
<td>DFHAPIQ</td>
<td>CSECT (OCO)</td>
<td>AP domain - user exit data access service</td>
<td>03</td>
</tr>
<tr>
<td>DFHAPIQT</td>
<td>CSECT (OCO)</td>
<td>APIQ trace interpretation data</td>
<td>03</td>
</tr>
<tr>
<td>DFHAPIQX</td>
<td>Macro</td>
<td>APIQ request</td>
<td>11</td>
</tr>
<tr>
<td>DFHAPIQY</td>
<td>DSECT</td>
<td>APIQ parameter list</td>
<td>11</td>
</tr>
</tbody>
</table>
Table 123. CICS modules directory (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHAPJC</td>
<td>CSECT</td>
<td>AP domain - journal interface gate service</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHAPLH1</td>
<td>CSECT</td>
<td>-</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHAPLH3</td>
<td>CSECT</td>
<td>-</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHAPLIA</td>
<td>CSECT</td>
<td>AP domain - language interface program</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHAPLIT</td>
<td>CSECT</td>
<td>(OCO) AP domain - language interface service</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHAPL11</td>
<td>CSECT</td>
<td>(OCO) AP domain - language interface functions 1</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHAPL12</td>
<td>CSECT</td>
<td>(OCO) AP domain - language interface functions 2</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHAPL13</td>
<td>CSECT</td>
<td>(OCO) AP domain - language interface functions 3</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHAPL14</td>
<td>CSECT</td>
<td>-</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHAPL15</td>
<td>CSECT</td>
<td>-</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHAPL16</td>
<td>CSECT</td>
<td>-</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHAPL17</td>
<td>CSECT</td>
<td>-</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHAPL1J</td>
<td>CSECT</td>
<td>-</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHAPL3</td>
<td>CSECT</td>
<td>-</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHAPNT</td>
<td>CSECT</td>
<td>AP domain - MXT notify gate</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHAPPG</td>
<td>CSECT</td>
<td>AP domain - optimize initial_link for</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHMIR</td>
<td>CSECT</td>
<td>-</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHAPPIS</td>
<td>CSECT</td>
<td>Java hotpooling PIPI service routines</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHAPPIV</td>
<td>CSECT</td>
<td>Java hotpooling PIPI service routines</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHAPRC</td>
<td>CSECT</td>
<td>User log record recovery module</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHAPRDA</td>
<td>CSECT</td>
<td>APROD interface parameter area</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHAPRDR</td>
<td>CSECT</td>
<td>Resource definition recovery module</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHAPRDT</td>
<td>CSECT</td>
<td>APROD translate table</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHAPRT</td>
<td>CSECT</td>
<td>AP Domain - route transaction gate</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHAPRTA</td>
<td>DSECT</td>
<td>APRT parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHAPRTM</td>
<td>Macro</td>
<td>APRT request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHAPRTT</td>
<td>CSECT</td>
<td>APROM trace interpretation data</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHAPSDF</td>
<td>CSECT</td>
<td>AP domain - formatted dump print module</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHAPS1</td>
<td>CSECT</td>
<td>AP domain - gate initialization</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHAPS1P</td>
<td>CSECT</td>
<td>AP domain - system initialization program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHAPS1M</td>
<td>CSECT</td>
<td>AP domain - storage notify gate</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHAPS1T</td>
<td>CSECT</td>
<td>AP domain - statistics collection</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHAPT1C</td>
<td>CSECT</td>
<td>AP domain - TC transport for Request streams</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHAPT1A</td>
<td>CSECT</td>
<td>APTC interface parameter area</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHAPT1C</td>
<td>CSECT</td>
<td>APTC interface macro</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHAPT1T</td>
<td>CSECT</td>
<td>-</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHAPT1C</td>
<td>CSECT</td>
<td>AP TC trace interpretation</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHAPT1I</td>
<td>CSECT</td>
<td>AP domain - timer notify gate</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHAPT1M</td>
<td>CSECT</td>
<td>AP domain - interval control midnight task</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHAPT1X</td>
<td>CSECT</td>
<td>AP domain - expiry analysis task</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHAPT1A</td>
<td>CSECT</td>
<td>IRC trace point ID aliases</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHAPT1A</td>
<td>CSECT</td>
<td>IRC trace interpreter</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHAPT1A</td>
<td>CSECT</td>
<td>XRF trace interpreter</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHAPT1A</td>
<td>CSECT</td>
<td>User exit trace interpreter</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHAPT1D</td>
<td>CSECT</td>
<td>DFHAPDM/DFHAPAP trace interpreter</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHAPT1A</td>
<td>CSECT</td>
<td>Data tables trace interpreter</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHAPT1F</td>
<td>CSECT</td>
<td>SAA communications and resource recovery</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHAPT1G</td>
<td>CSECT</td>
<td>interfaces trace interpreter</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHAPT1A</td>
<td>CSECT</td>
<td>ZC exception and VTAM exit trace interpreter</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHAPT1A</td>
<td>CSECT</td>
<td>DFHAPDM/DFHAPAP trace interpreter</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHAPT1A</td>
<td>CSECT</td>
<td>AP domain - resource definition interpretation</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHAPT1A</td>
<td>CSECT</td>
<td>CICS OS/2 LU2 mirror trace interpreter</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHAPT1A</td>
<td>CSECT</td>
<td>Autoinstall terminal model manager trace interpreter</td>
<td>OS 03</td>
</tr>
</tbody>
</table>
## CICS directory

### Table 123. CICS modules directory (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHAPTR0</td>
<td>CSECT</td>
<td>LU6.2 application request logic trace interpreter</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHAPTRP</td>
<td>CSECT</td>
<td>Program control trace interpreter</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHAPTRR</td>
<td>CSECT (OCO)</td>
<td>Partner resource manager trace interpreter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHAPTRS</td>
<td>CSECT (OCO)</td>
<td>AP domain - DFHEISR trace interpreter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHAPTRU</td>
<td>CSECT</td>
<td>ZC install trace interpretation</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHAPTRV</td>
<td>CSECT (OCO)</td>
<td>AP domain - DFHSRP trace interpreter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHAPTRW</td>
<td>CSECT (OCO)</td>
<td>AP domain - FEPI trace interpreter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHAPTRX</td>
<td>CSECT</td>
<td>ZC persistent sessions trace interpretation</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHAPTRY</td>
<td>CSECT</td>
<td>AP domain - trace formatting (APRM, APXM, IXML, and TDXM)</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHAPTR2</td>
<td>CSECT</td>
<td>AP domain - statistics trace interpreter</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHAPTR5</td>
<td>CSECT</td>
<td>File control trace interpreter</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHAPTR6</td>
<td>CSECT</td>
<td>DBCTL trace interpreter</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHAPTR7</td>
<td>CSECT</td>
<td>Transaction routing trace interpreter</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHAPTR8</td>
<td>CSECT</td>
<td>Security trace interpreter</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHAPTR9</td>
<td>CSECT</td>
<td>Interval control trace interpreter</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHAPUEA</td>
<td>DSECT</td>
<td>APUE parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHAPUEM</td>
<td>Macro</td>
<td>APUE request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHAPUET</td>
<td>CSECT</td>
<td>APUE trace interpretation data</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHAPXDD</td>
<td>CSECT</td>
<td>AP domain - transaction definition extension</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHAPXM</td>
<td>CSECT</td>
<td>AP domain - transaction initialization and termination services</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHAPXMA</td>
<td>DSECT</td>
<td>APXM parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHAPXME</td>
<td>CSECT</td>
<td>AP domain - XM exception handler</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHAPXMT</td>
<td>CSECT (OCO)</td>
<td>APXM trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHASMVS</td>
<td>Other</td>
<td>Cataloged procedure to assemble CICS programs and user-written macro-level programs</td>
<td>18 -</td>
</tr>
<tr>
<td>DFHASSUA</td>
<td>DSECT</td>
<td>ASSU parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHASSUM</td>
<td>Macro</td>
<td>ASSU request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHASSUT</td>
<td>CSECT</td>
<td>ASSU trace interpretation data</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHASV</td>
<td>CSECT</td>
<td>Authorized services interface</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHATUP</td>
<td>CSECT</td>
<td>Audit trail Utility Program</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHAUDUF</td>
<td>CSECT</td>
<td>-</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHAUPLE</td>
<td>Other</td>
<td>Cataloged procedure to assemble and link-edit CICS control tables, and provide information to SMP/E</td>
<td>02 -</td>
</tr>
<tr>
<td>DFHAUTH</td>
<td>Macro</td>
<td>Verify environment and activate CICS SVCs</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHAXI</td>
<td>Macro</td>
<td>XRF alternate subsystem identifier table</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHA03DS</td>
<td>DSECT</td>
<td>VTAM statistics</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHA03DS</td>
<td>DSECT</td>
<td>VTAM statistics - COBOL</td>
<td>C2 07</td>
</tr>
<tr>
<td>DFHA03DS</td>
<td>DSECT</td>
<td>VTAM statistics - PL/I</td>
<td>P2 -</td>
</tr>
<tr>
<td>DFHA04DS</td>
<td>DSECT</td>
<td>Autoinstall statistics</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHA04DS</td>
<td>DSECT</td>
<td>Autoinstall statistics - COBOL</td>
<td>C2 07</td>
</tr>
<tr>
<td>DFHA04DS</td>
<td>DSECT</td>
<td>Autoinstall statistics - PL/I</td>
<td>P2 -</td>
</tr>
<tr>
<td>DFHA06DS</td>
<td>DSECT</td>
<td>Terminal statistics</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHA06DS</td>
<td>DSECT</td>
<td>Terminal statistics - COBOL</td>
<td>C2 07</td>
</tr>
<tr>
<td>DFHA06DS</td>
<td>DSECT</td>
<td>Terminal statistics - PL/I</td>
<td>P2 -</td>
</tr>
<tr>
<td>DFHA08DS</td>
<td>DSECT</td>
<td>LSR pool statistics</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHA08DS</td>
<td>DSECT</td>
<td>LSR pool statistics - COBOL</td>
<td>C2 07</td>
</tr>
<tr>
<td>DFHA08DS</td>
<td>DSECT</td>
<td>LSR pool statistics - PL/I</td>
<td>P2 -</td>
</tr>
<tr>
<td>DFHA09DS</td>
<td>DSECT</td>
<td>LSR pool file-related statistics</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHA09DS</td>
<td>DSECT</td>
<td>LSR pool file-related statistics</td>
<td>C2 07</td>
</tr>
</tbody>
</table>
Table 123. CICS modules directory (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHA09DS</td>
<td>DSECT</td>
<td>LSR pool file-related statistics</td>
<td>P2 -</td>
</tr>
<tr>
<td>DFHA14DS</td>
<td>DSECT</td>
<td>ISC/IRC statistics for system entries</td>
<td>I1 -</td>
</tr>
<tr>
<td>DFHA14DS</td>
<td>DSECT</td>
<td>ISC/IRC statistics for system entries</td>
<td>C2 07</td>
</tr>
<tr>
<td>DFHA14DS</td>
<td>DSECT</td>
<td>ISC/IRC statistics for system entries</td>
<td>P2 -</td>
</tr>
<tr>
<td>DFHA16DS</td>
<td>DSECT</td>
<td>Table manager statistics</td>
<td>I1 -</td>
</tr>
<tr>
<td>DFHA16DS</td>
<td>DSECT</td>
<td>Table manager statistics</td>
<td>C2 07</td>
</tr>
<tr>
<td>DFHA16DS</td>
<td>DSECT</td>
<td>Table manager statistics</td>
<td>P2 -</td>
</tr>
<tr>
<td>DFHA17DS</td>
<td>DSECT</td>
<td>File control statistics</td>
<td>I1 -</td>
</tr>
<tr>
<td>DFHA17DS</td>
<td>DSECT</td>
<td>File control statistics</td>
<td>C2 07</td>
</tr>
<tr>
<td>DFHA17DS</td>
<td>DSECT</td>
<td>File control statistics</td>
<td>P2 -</td>
</tr>
<tr>
<td>DFHA20DS</td>
<td>DSECT</td>
<td>ISC/IRC statistics for mode entries</td>
<td>I1 -</td>
</tr>
<tr>
<td>DFHA20DS</td>
<td>DSECT</td>
<td>ISC/IRC statistics for mode entries</td>
<td>C2 07</td>
</tr>
<tr>
<td>DFHA20DS</td>
<td>DSECT</td>
<td>ISC/IRC statistics for mode entries</td>
<td>P2 -</td>
</tr>
<tr>
<td>DFHA21DS</td>
<td>DSECT</td>
<td>ISC/IRC attach-time statistics</td>
<td>I1 -</td>
</tr>
<tr>
<td>DFHA21DS</td>
<td>DSECT</td>
<td>ISC/IRC attach-time statistics</td>
<td>C2 07</td>
</tr>
<tr>
<td>DFHA21DS</td>
<td>DSECT</td>
<td>ISC/IRC attach-time statistics</td>
<td>P2 -</td>
</tr>
<tr>
<td>DFHA22DS</td>
<td>DSECT</td>
<td>FEPI pool statistics</td>
<td>I1 -</td>
</tr>
<tr>
<td>DFHA22DS</td>
<td>DSECT</td>
<td>FEPI pool statistics</td>
<td>C2 07</td>
</tr>
<tr>
<td>DFHA22DS</td>
<td>DSECT</td>
<td>FEPI pool statistics</td>
<td>P2 -</td>
</tr>
<tr>
<td>DFHA23DS</td>
<td>DSECT</td>
<td>FEPI connection statistics</td>
<td>I1 -</td>
</tr>
<tr>
<td>DFHA23DS</td>
<td>DSECT</td>
<td>FEPI connection statistics</td>
<td>C2 07</td>
</tr>
<tr>
<td>DFHA23DS</td>
<td>DSECT</td>
<td>FEPI connection statistics</td>
<td>P2 -</td>
</tr>
<tr>
<td>DFHA24DS</td>
<td>DSECT</td>
<td>FEPI target statistics</td>
<td>I1 -</td>
</tr>
<tr>
<td>DFHA24DS</td>
<td>DSECT</td>
<td>FEPI target statistics</td>
<td>C2 07</td>
</tr>
<tr>
<td>DFHA24DS</td>
<td>DSECT</td>
<td>FEPI target statistics</td>
<td>P2 -</td>
</tr>
<tr>
<td>DFHBAAC</td>
<td>CSECT</td>
<td>BAAC CDURUN and Gate module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHBAACT</td>
<td>CSECT</td>
<td>BAM Activity Class class declaration</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHBAAC0</td>
<td>CSECT</td>
<td>- 03</td>
<td></td>
</tr>
<tr>
<td>DFHBAAC1</td>
<td>CSECT</td>
<td>- 03</td>
<td></td>
</tr>
<tr>
<td>DFHBAAC2</td>
<td>CSECT</td>
<td>- 03</td>
<td></td>
</tr>
<tr>
<td>DFHBAAC3</td>
<td>CSECT</td>
<td>- 03</td>
<td></td>
</tr>
<tr>
<td>DFHBAAC4</td>
<td>CSECT</td>
<td>- 03</td>
<td></td>
</tr>
<tr>
<td>DFHBAAC5</td>
<td>CSECT</td>
<td>- 03</td>
<td></td>
</tr>
<tr>
<td>DFHBAAC6</td>
<td>CSECT</td>
<td>- 03</td>
<td></td>
</tr>
<tr>
<td>DFHBAAR1</td>
<td>CSECT</td>
<td>- 03</td>
<td></td>
</tr>
<tr>
<td>DFHBAAR2</td>
<td>CSECT</td>
<td>- 03</td>
<td></td>
</tr>
<tr>
<td>DFHBAAR10</td>
<td>CSECT</td>
<td>- 03</td>
<td></td>
</tr>
<tr>
<td>DFHBAAR11</td>
<td>CSECT</td>
<td>- 03</td>
<td></td>
</tr>
<tr>
<td>DFHBAAR12</td>
<td>CSECT</td>
<td>- 03</td>
<td></td>
</tr>
<tr>
<td>DFHBAABR</td>
<td>CSECT</td>
<td>BABR CDURUN and Gate Module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHBAABRA</td>
<td>CSECT</td>
<td>BABR interface parameter area</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHBAABRM</td>
<td>Macro</td>
<td>BABR interface macro</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHBAABRT</td>
<td>CSECT</td>
<td>- 03</td>
<td></td>
</tr>
<tr>
<td>DFHBABU1</td>
<td>CSECT</td>
<td>- 03</td>
<td></td>
</tr>
<tr>
<td>DFHBAEBR</td>
<td>CSECT</td>
<td>BACR CDURUN and Gate Module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHBAEBRT</td>
<td>CSECT</td>
<td>- 03</td>
<td></td>
</tr>
<tr>
<td>DFHBADM</td>
<td>CSECT</td>
<td>BA Domain Management</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHBADUF</td>
<td>CSECT</td>
<td>BA Domain Dump Formatting</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHBADU1</td>
<td>CSECT</td>
<td>- 03</td>
<td></td>
</tr>
<tr>
<td>DFHBAGDT</td>
<td>CSECT</td>
<td>- 03</td>
<td></td>
</tr>
<tr>
<td>DFHBALR2</td>
<td>CSECT</td>
<td>- 03</td>
<td></td>
</tr>
<tr>
<td>DFHBALR3</td>
<td>CSECT</td>
<td>- 03</td>
<td></td>
</tr>
<tr>
<td>DFHBALR4</td>
<td>CSECT</td>
<td>- 03</td>
<td></td>
</tr>
<tr>
<td>DFHBALR5</td>
<td>CSECT</td>
<td>- 03</td>
<td></td>
</tr>
<tr>
<td>DFHBALR6</td>
<td>CSECT</td>
<td>- 03</td>
<td></td>
</tr>
<tr>
<td>DFHBALR7</td>
<td>CSECT</td>
<td>- 03</td>
<td></td>
</tr>
<tr>
<td>DFHBALR8</td>
<td>CSECT</td>
<td>- 03</td>
<td></td>
</tr>
</tbody>
</table>
## CICS directory

**Table 123. CICS modules directory (continued)**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHBALR9</td>
<td>CSECT</td>
<td>CSDUP - SPI offline messages table (51xx)</td>
<td>03</td>
</tr>
<tr>
<td>DFHBAMS1</td>
<td>CSECT</td>
<td>CSDUP - SPI offline messages table (52xx)</td>
<td>03</td>
</tr>
<tr>
<td>DFHBAMS5</td>
<td>CSECT</td>
<td>CSDUP - SPI offline messages table (55xx)</td>
<td>03</td>
</tr>
<tr>
<td>DFHBAMS6</td>
<td>CSECT</td>
<td>CSDUP - SPI offline messages table (56xx)</td>
<td>03</td>
</tr>
<tr>
<td>DFHBAOF1</td>
<td>CSECT</td>
<td>-</td>
<td>03</td>
</tr>
<tr>
<td>DFHBAAPR</td>
<td>CSECT</td>
<td>BAPR CURDUN and Gate Module</td>
<td>03</td>
</tr>
<tr>
<td>DFHBAAPT</td>
<td>CSECT</td>
<td>-</td>
<td>03</td>
</tr>
<tr>
<td>DFHBAPO</td>
<td>CSECT</td>
<td>-</td>
<td>03</td>
</tr>
<tr>
<td>DFHBAT1</td>
<td>CSECT</td>
<td>-</td>
<td>03</td>
</tr>
<tr>
<td>DFHBAU</td>
<td>CSECT</td>
<td>-</td>
<td>03</td>
</tr>
<tr>
<td>DFHBARC</td>
<td>CSECT</td>
<td>-</td>
<td>03</td>
</tr>
<tr>
<td>DFHBARU</td>
<td>CSECT</td>
<td>CBTS Repository Utility Program</td>
<td>03</td>
</tr>
<tr>
<td>DFHBAUC</td>
<td>CSECT</td>
<td>BRDATA for CBTS Constants</td>
<td>08</td>
</tr>
<tr>
<td>DFHBAACL</td>
<td>CSECT</td>
<td>BRDATA for CBTS Constants</td>
<td>17</td>
</tr>
<tr>
<td>DFHBAASO</td>
<td>CSECT</td>
<td>BRDATA for CBTS Constants</td>
<td>07</td>
</tr>
<tr>
<td>DFHBAOSSD</td>
<td>CSECT</td>
<td>BRDATA for CBTS Bridge Exit</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHBAOSDH</td>
<td>CSECT</td>
<td>BRDATA for CBTS Bridge Exit</td>
<td>08 -</td>
</tr>
<tr>
<td>DFHBAOSDL</td>
<td>CSECT</td>
<td>BRDATA for CBTS Bridge Exit</td>
<td>17 -</td>
</tr>
<tr>
<td>DFHBAOSDO</td>
<td>CSECT</td>
<td>BRDATA for CBTS Bridge Exit</td>
<td>19 -</td>
</tr>
<tr>
<td>DFHBAOASP</td>
<td>CSECT</td>
<td>BASP Gate Module</td>
<td>03</td>
</tr>
<tr>
<td>DFHBATATI</td>
<td>CSECT</td>
<td>BAM Domain Trace Interpretation</td>
<td>03</td>
</tr>
<tr>
<td>DFHBATT</td>
<td>CSECT</td>
<td>BAS 03 DURUN and Gate Module</td>
<td>03</td>
</tr>
<tr>
<td>DFHBAUETT</td>
<td>CSECT</td>
<td>-</td>
<td>03</td>
</tr>
<tr>
<td>DFHBAUE</td>
<td>CSECT</td>
<td>BAUE Gate Module</td>
<td>03</td>
</tr>
<tr>
<td>DFHBAUV1</td>
<td>CSECT</td>
<td>-</td>
<td>03</td>
</tr>
<tr>
<td>DFHBAUXM</td>
<td>CSECT</td>
<td>BA XM Interfaces</td>
<td>03</td>
</tr>
<tr>
<td>DFHBAZMT</td>
<td>CSECT</td>
<td>-</td>
<td>03</td>
</tr>
<tr>
<td>DFHBEPB</td>
<td>CSECT</td>
<td>RDO batch error program</td>
<td>03</td>
</tr>
<tr>
<td>DFHBEPB</td>
<td>CSECT</td>
<td>RDO message formatting module</td>
<td>03</td>
</tr>
<tr>
<td>DFHBTFCA</td>
<td>Macro</td>
<td>Built-in functions TCA macro</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHBMPI</td>
<td>Macro</td>
<td>BMS picture analysis</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHBMMS</td>
<td>Macro</td>
<td>Basic mapping support request</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHBMSCA1</td>
<td>Symbolic</td>
<td>BMS attribute definitions</td>
<td>11 08</td>
</tr>
<tr>
<td>DFHBMSCA2</td>
<td>Symbolic</td>
<td>BMS attribute definitions</td>
<td>C2 07</td>
</tr>
<tr>
<td>DFHBMUS</td>
<td>Macro</td>
<td>-</td>
<td>18</td>
</tr>
<tr>
<td>DFHBMUSPA</td>
<td>Macro</td>
<td>-</td>
<td>03</td>
</tr>
<tr>
<td>DFHBMUTM</td>
<td>Macro</td>
<td>Trace BMS module generation options</td>
<td>08 -</td>
</tr>
<tr>
<td>DFHBPXPA</td>
<td>Sample</td>
<td>-</td>
<td>02</td>
</tr>
<tr>
<td>DFHBPXPA0</td>
<td>Sample</td>
<td>-</td>
<td>02</td>
</tr>
<tr>
<td>DFHBPXPI</td>
<td>Sample</td>
<td>-</td>
<td>02</td>
</tr>
<tr>
<td>DFHBRACD</td>
<td>Symbolic</td>
<td>Bridge copybook</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHBRACH</td>
<td>Symbolic</td>
<td>Bridge copybook</td>
<td>D2 -</td>
</tr>
<tr>
<td>DFHBRACL</td>
<td>Symbolic</td>
<td>Bridge copybook</td>
<td>P2 17</td>
</tr>
<tr>
<td>DFHBRACD</td>
<td>Symbolic</td>
<td>Bridge copybook</td>
<td>C2 -</td>
</tr>
<tr>
<td>DFHBRARD</td>
<td>Symbolic</td>
<td>Bridge copybook</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHBRARH</td>
<td>Symbolic</td>
<td>Bridge copybook</td>
<td>D2 -</td>
</tr>
<tr>
<td>DFHBRARL</td>
<td>Symbolic</td>
<td>Bridge copybook</td>
<td>P2 17</td>
</tr>
<tr>
<td>DFHBRARO</td>
<td>Symbolic</td>
<td>Bridge copybook</td>
<td>C2 -</td>
</tr>
<tr>
<td>DFHBRAT</td>
<td>CSECT</td>
<td>Design Bridge - BRAT Gate Functions</td>
<td>03</td>
</tr>
<tr>
<td>DFHBRATA</td>
<td>CSECT</td>
<td>BRAT interface parameter area</td>
<td>08 -</td>
</tr>
<tr>
<td>DFHBRATM</td>
<td>CSECT</td>
<td>DFHBRAO interface macro</td>
<td>08 -</td>
</tr>
<tr>
<td>DFHBRATT</td>
<td>CSECT</td>
<td>-</td>
<td>03</td>
</tr>
<tr>
<td>DFHBRBFB</td>
<td>CSECT</td>
<td>Bridge module</td>
<td>08 -</td>
</tr>
<tr>
<td>DFHBRDCD</td>
<td>CSECT</td>
<td>-</td>
<td>03</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
<td>Library</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>--------------------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>DFHBDRUF</td>
<td>CSECT</td>
<td>Bridge module</td>
<td>03</td>
</tr>
<tr>
<td>DFHBDRFM</td>
<td>CSECT</td>
<td>Bridge module</td>
<td>03</td>
</tr>
<tr>
<td>DFHBDRFMT</td>
<td>Symbolic</td>
<td>Trace interpretation data</td>
<td>03</td>
</tr>
<tr>
<td>DFHBDRIC</td>
<td>CSECT</td>
<td>Bridge module</td>
<td>03</td>
</tr>
<tr>
<td>DFHBRIQ</td>
<td>CSECT</td>
<td>Design Bridge - BRIQ Gate Functions</td>
<td>03</td>
</tr>
<tr>
<td>DFHBRIQA</td>
<td>CSECT</td>
<td>BRIQ interface parameter area</td>
<td>OS</td>
</tr>
<tr>
<td>DFHBRIQI</td>
<td>CSECT</td>
<td>OS</td>
<td></td>
</tr>
<tr>
<td>DFHBRIQM</td>
<td>CSECT</td>
<td>DFHBRIQ interface macro</td>
<td>OS</td>
</tr>
<tr>
<td>DFHBRIQT</td>
<td>CSECT</td>
<td>OS</td>
<td>03</td>
</tr>
<tr>
<td>DFHBRIQX</td>
<td>Macro</td>
<td>Bridge XPI macro</td>
<td>11</td>
</tr>
<tr>
<td>DFHBRIQY</td>
<td>Symbolic</td>
<td>Copybook</td>
<td>11</td>
</tr>
<tr>
<td>DFHBMCD</td>
<td>Symbolic</td>
<td>Bridge copybook</td>
<td>19</td>
</tr>
<tr>
<td>DFHBMCH</td>
<td>Symbolic</td>
<td>Bridge copybook</td>
<td>D3</td>
</tr>
<tr>
<td>DFHBMCL</td>
<td>Symbolic</td>
<td>Bridge copybook</td>
<td>P3</td>
</tr>
<tr>
<td>DFHBMCO</td>
<td>Symbolic</td>
<td>Bridge copybook</td>
<td>C3</td>
</tr>
<tr>
<td>DFHBME</td>
<td>CSECT</td>
<td>BR Exit Program</td>
<td>03</td>
</tr>
<tr>
<td>DFHBMF</td>
<td>CSECT</td>
<td>BR Formatter Program</td>
<td>03</td>
</tr>
<tr>
<td>DFHBMHD</td>
<td>Symbolic</td>
<td>Bridge copybook</td>
<td>19</td>
</tr>
<tr>
<td>DFHBMHH</td>
<td>Symbolic</td>
<td>Bridge copybook</td>
<td>D3</td>
</tr>
<tr>
<td>DFHBMHL</td>
<td>Symbolic</td>
<td>Bridge copybook</td>
<td>P3</td>
</tr>
<tr>
<td>DFHBMHQ</td>
<td>Symbolic</td>
<td>Bridge copybook</td>
<td>C3</td>
</tr>
<tr>
<td>DFHBMQL</td>
<td>Symbolic</td>
<td>Bridge copybook</td>
<td>19</td>
</tr>
<tr>
<td>DFHBMQO</td>
<td>Symbolic</td>
<td>Bridge copybook</td>
<td>D3</td>
</tr>
<tr>
<td>DFHBMQI</td>
<td>Symbolic</td>
<td>Bridge copybook</td>
<td>P3</td>
</tr>
<tr>
<td>DFHBMQY</td>
<td>Symbolic</td>
<td>Bridge copybook</td>
<td>C3</td>
</tr>
<tr>
<td>DFHBRMS</td>
<td>CSECT</td>
<td>Bridge module</td>
<td>03</td>
</tr>
<tr>
<td>DFHBRM</td>
<td>CSECT</td>
<td>DFHBRM Design Bridge - Recovery Manager</td>
<td>03</td>
</tr>
<tr>
<td>DFHBRSCD</td>
<td>Symbolic</td>
<td>Bridge copybook</td>
<td>19</td>
</tr>
<tr>
<td>DFHBRSCH</td>
<td>Symbolic</td>
<td>Bridge copybook</td>
<td>D3</td>
</tr>
<tr>
<td>DFHBRSDL</td>
<td>Symbolic</td>
<td>Bridge copybook</td>
<td>P3</td>
</tr>
<tr>
<td>DFHBRSDD</td>
<td>Symbolic</td>
<td>Bridge copybook</td>
<td>C3</td>
</tr>
<tr>
<td>DFHBRSDD</td>
<td>Symbolic</td>
<td>Bridge copybook</td>
<td>19</td>
</tr>
<tr>
<td>DFHBRSDE</td>
<td>Symbolic</td>
<td>Bridge copybook</td>
<td>03</td>
</tr>
<tr>
<td>DFHBRSDF</td>
<td>Symbolic</td>
<td>Bridge copybook</td>
<td>P3</td>
</tr>
<tr>
<td>DFHBRSDF</td>
<td>Symbolic</td>
<td>Bridge copybook</td>
<td>C3</td>
</tr>
<tr>
<td>DFHBRSP</td>
<td>CSECT</td>
<td>Bridge module</td>
<td>03</td>
</tr>
<tr>
<td>DFHBRSPA</td>
<td>Symbolic</td>
<td>Bridge copybook</td>
<td>05</td>
</tr>
<tr>
<td>DFHBRSPM</td>
<td>Symbolic</td>
<td>Bridge copybook</td>
<td>OS</td>
</tr>
<tr>
<td>DFHBRSPQ</td>
<td>Symbolic</td>
<td>Bridge copybook</td>
<td>03</td>
</tr>
<tr>
<td>DFHB临时</td>
<td>CSECT</td>
<td>Bridge Virtual Terminal Buffer</td>
<td>03</td>
</tr>
<tr>
<td>DFHB临时</td>
<td>CSECT</td>
<td>Bridge module</td>
<td>03</td>
</tr>
<tr>
<td>DFHB临时</td>
<td>CSECT</td>
<td>Bridge module</td>
<td>03</td>
</tr>
<tr>
<td>DFHB临时</td>
<td>Macro</td>
<td>Bridge module</td>
<td>03</td>
</tr>
<tr>
<td>DFHB临时</td>
<td>CSECT</td>
<td>BR XM Principal Client</td>
<td>03</td>
</tr>
<tr>
<td>DFHB临时</td>
<td>CSECT</td>
<td>Generate binary search code</td>
<td>11</td>
</tr>
<tr>
<td>DFHB临时</td>
<td>CSECT</td>
<td>Switch subspace request</td>
<td>OS</td>
</tr>
<tr>
<td>DFHB临时</td>
<td>CSECT</td>
<td>BMS 3270 builder</td>
<td>OS</td>
</tr>
<tr>
<td>DFHB临时</td>
<td>CSECT</td>
<td>Add SCS support</td>
<td>OS</td>
</tr>
<tr>
<td>DFHB临时</td>
<td>CSECT</td>
<td>Add DFHZCP 3270 support</td>
<td>OS</td>
</tr>
<tr>
<td>DFHB临时</td>
<td>CSECT</td>
<td>Build terminal session</td>
<td>OS</td>
</tr>
<tr>
<td>DFHB临时</td>
<td>CSECT</td>
<td>Build pipeline pool table entry</td>
<td>OS</td>
</tr>
<tr>
<td>DFHB临时</td>
<td>CSECT</td>
<td>Generate sessions for modegroup</td>
<td>OS</td>
</tr>
<tr>
<td>DFHB临时</td>
<td>CSECT</td>
<td>Build a modegroup</td>
<td>OS</td>
</tr>
<tr>
<td>DFHB临时</td>
<td>CSECT</td>
<td>Build a connection</td>
<td>OS</td>
</tr>
<tr>
<td>DFHB临时</td>
<td>CSECT</td>
<td>Build DFHKCP support in a system entry</td>
<td>OS</td>
</tr>
<tr>
<td>DFHB临时</td>
<td>CSECT</td>
<td>Build stats support in a system entry</td>
<td>OS</td>
</tr>
<tr>
<td>DFHB临时</td>
<td>CSECT</td>
<td>Build security support in a system entry</td>
<td>OS</td>
</tr>
</tbody>
</table>
## CICS directory

### Table 123. CICS modules directory (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHBSSZ</td>
<td>CSECT</td>
<td>Build VTAM support in a system entry</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHBSSZG</td>
<td>CSECT</td>
<td>Add an APPC single-session</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHBSSZI</td>
<td>CSECT</td>
<td>Add an indirect terminal system</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHBSSZL</td>
<td>CSECT</td>
<td>Add a local terminal system</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHBSSZM</td>
<td>CSECT</td>
<td>Introduce new system to ZCP</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHBSSZP</td>
<td>CSECT</td>
<td>Add an APPC parallel-session</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHBSSZR</td>
<td>CSECT</td>
<td>Add an MRO system</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHBSSZS</td>
<td>CSECT</td>
<td>Add an APPC</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHBSSZ6</td>
<td>CSECT</td>
<td>Add an LU6.1 connection</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHBST</td>
<td>CSECT</td>
<td>Common TCTTE builder</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHBSTB</td>
<td>CSECT</td>
<td>Add a resource for BMS</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHBSTBL</td>
<td>CSECT</td>
<td>Add logical device support</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHBSTC</td>
<td>CSECT</td>
<td>Add install-time options support</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHBSTD</td>
<td>CSECT</td>
<td>Add DFHDIP support</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHBSTE</td>
<td>CSECT</td>
<td>Add EDF support</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHBSTH</td>
<td>CSECT</td>
<td>EXEC interface builder</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHBSTI</td>
<td>CSECT</td>
<td>Add DFHICP support</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHBSTM</td>
<td>CSECT</td>
<td>Add DFHMGP support</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHBSTO</td>
<td>CSECT</td>
<td>Spooler terminal builder</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHBSTP3</td>
<td>CSECT</td>
<td>Add 3270-copy support</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHBSTS</td>
<td>CSECT</td>
<td>Add DFHIST support</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHBSTT</td>
<td>CSECT</td>
<td>Add DFHKCP support</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHBSTZ</td>
<td>CSECT</td>
<td>Build terminal or session resource</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHBSTZA</td>
<td>CSECT</td>
<td>Add DFHZCP support</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHBSTZB</td>
<td>CSECT</td>
<td>Add or delete bind-image</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHBSTZC</td>
<td>CSECT</td>
<td>Add single-session to APPC</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHBSTZE</td>
<td>CSECT</td>
<td>Set error message writer fields</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHBSTZL</td>
<td>CSECT</td>
<td>Add logical device code support</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHBSTZO</td>
<td>CSECT</td>
<td>Add an MVS console</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHBSTZP</td>
<td>CSECT</td>
<td>Pipeline terminal builder</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHBSTZR</td>
<td>CSECT</td>
<td>Add IRC session</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHBSTZS</td>
<td>CSECT</td>
<td>Add an APPC session</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHBSTZV</td>
<td>CSECT</td>
<td>Add VTAM and IRC information</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHBSTZ2</td>
<td>CSECT</td>
<td>Add non-APP CCP session</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHBSTZ1</td>
<td>CSECT</td>
<td>Add remote terminal support</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHBSTZ2</td>
<td>CSECT</td>
<td>Remote APPC builder</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHBSTZ3</td>
<td>CSECT</td>
<td>Add 3270 support</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHBSSZ</td>
<td>CSECT</td>
<td>Add terminal or session</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHBSSZS</td>
<td>CSECT</td>
<td>Add session to LU6.2 support</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHBSSZV</td>
<td>CSECT</td>
<td>Add VTAM terminal or session</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHBST</td>
<td>Macro</td>
<td>Parameter sublist translation</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHCALLA</td>
<td>CSECT</td>
<td>CZ Direct-to-CICS</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCAPB</td>
<td>CSECT</td>
<td>CSDUP - command analysis program (DFHCAP)</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCAPC</td>
<td>CSECT</td>
<td>RDO utility - RDL command locator</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCCC</td>
<td>CSECT (OCO)</td>
<td>GC/LC domains - functions</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCCC</td>
<td>DSECT</td>
<td>CCCC parameter list</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCCC</td>
<td>Macro</td>
<td>CCCC request</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCCCT</td>
<td>CSECT (OCO)</td>
<td>CCCC trace interpretation data</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCCDM</td>
<td>CSECT (OCO)</td>
<td>GC/LC domains - initialization/termination</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCCDUF</td>
<td>CSECT (OCO)</td>
<td>SDUMP formatter for GC/LC domains</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCN</td>
<td>CSECT</td>
<td>Data conversion for CICS OS/2 ISC users</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCNV2</td>
<td>CSECT</td>
<td>Convert characters in multi-byte representation</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCTR</td>
<td>CSECT (OCO)</td>
<td>Trace interpreter for GC/LC domains</td>
<td>OS 03</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
<td>Library</td>
</tr>
<tr>
<td>----------</td>
<td>-------</td>
<td>------------------------------------------------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>DFHCUTL</td>
<td>CSECT</td>
<td>CICS local catalog initialization program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDBLK</td>
<td>Symbolic</td>
<td>CONVDATA area</td>
<td>11 02</td>
</tr>
<tr>
<td>DFHDBMI</td>
<td>Other</td>
<td>CDBM group file definition JCL</td>
<td>- 02</td>
</tr>
<tr>
<td>DFHCDBTC</td>
<td>Macro</td>
<td>Domain call argument conversion</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHCDC</td>
<td>Macro</td>
<td>Syntax analysis and code generation for DFHxxyM/X domain call macros</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHCDCON</td>
<td>CSECT</td>
<td>Formatted parameter list translator</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDEDA</td>
<td>DSECT</td>
<td>CDED parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHDEDM</td>
<td>Macro</td>
<td>CDED request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHDEDT</td>
<td>CSECT</td>
<td>CDED trace interpretation data</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDKRN</td>
<td>CSECT</td>
<td>KE Java to CDURUN Interface</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDMMK</td>
<td>Macro</td>
<td>Domain call inner macro - generate assignments for IN keywords</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHDPMK</td>
<td>Macro</td>
<td>Domain call inner macro - generate assignments for OUT keywords</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHDSPL</td>
<td>Macro</td>
<td>Domain call inner macro - subvalues of character list</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHDSUB</td>
<td>Macro</td>
<td>Domain call inner macro - subvalues of sub-parameter list</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHDSYN</td>
<td>Macro</td>
<td>Syntax analysis on positional operands for DFHxxyM/X domain call macros</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHDTST</td>
<td>Macro</td>
<td>DFHTEST inner macro</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHDTYP</td>
<td>Macro</td>
<td>Determine domain call argument data type</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHCEGN</td>
<td>CSECT</td>
<td>Goodnight transaction stub</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHCEID</td>
<td>CSECT</td>
<td>DCE table clear routine</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHCESC</td>
<td>CSECT</td>
<td>Terminal, XRF, and enable timeout routines</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHCESD</td>
<td>CSECT</td>
<td>CICS shutdown assist program</td>
<td>19 -</td>
</tr>
<tr>
<td>DFHCESDP</td>
<td>CSECT</td>
<td>CICS shutdown assist program</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHCETRA</td>
<td>CSECT</td>
<td>Trace control transaction (CETR) - main program</td>
<td>03 03</td>
</tr>
<tr>
<td>DFHCETRB</td>
<td>CSECT</td>
<td>CETR - trace component flags inquire/set</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCETRC</td>
<td>CSECT</td>
<td>CETR - terminal/transaction trace control</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCETRD</td>
<td>CSECT</td>
<td>CETR - common subroutines</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCFCF</td>
<td>CSECT</td>
<td>CFDT Server CF Interface</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHCFCN</td>
<td>CSECT</td>
<td>CFDT Server Client Connect/Disconnect</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHCFD</td>
<td>CSECT</td>
<td>CFDT AXM Server Definitions</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHCFN</td>
<td>CSECT</td>
<td>CFDT ENF event interface</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHCFI</td>
<td>CSECT</td>
<td>CFDT Server Interface Module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHCFIQ</td>
<td>CSECT</td>
<td>CFDT Table Inquire Routines</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHCFLW</td>
<td>CSECT</td>
<td>CFDT Server Lock Wait Routines</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHCFMN</td>
<td>CSECT</td>
<td>CFDT Server Main Program</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHCMNS</td>
<td>CSECT</td>
<td>CFDT Server Messages</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHCFC</td>
<td>CSECT</td>
<td>CFDT Server Table Open/Close</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHCFO</td>
<td>CSECT</td>
<td>CFDT Server Operator Command Support</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHCFCR</td>
<td>CSECT</td>
<td>CFDT Server Parameter Processing</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHCFL</td>
<td>CSECT</td>
<td>CFDT Server Pool Reload routine</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHCFQ</td>
<td>CSECT</td>
<td>CFDT Server Record Request Routines</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHCFS</td>
<td>CSECT</td>
<td>CFDT ARM Restart Support</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHCFS</td>
<td>CSECT</td>
<td>CFDT Server Syncpoint and Restart</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHCFS</td>
<td>CSECT</td>
<td>CFDT Server Statistics Routines</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHCFS</td>
<td>CSECT</td>
<td>CFDT Statistics for list structure</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHCFS</td>
<td>CSECT</td>
<td>CFDT Statistics for table accesses</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHCFS</td>
<td>CSECT</td>
<td>CFDT Request statistics</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHCFS</td>
<td>CSECT</td>
<td>CFDT Statistics for server storage</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHCFS</td>
<td>CSECT</td>
<td>CFDT Server Pool Unload Routine</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHCFS</td>
<td>CSECT</td>
<td>CFDT Server External Security Support</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHCFS</td>
<td>CSECT</td>
<td>CICS mirror for CICS OS/2 and CICS/VM</td>
<td>05 03</td>
</tr>
</tbody>
</table>
### CICS directory

**Table 123. CICS modules directory (continued)**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHCJVMA</td>
<td>CSECT</td>
<td>JVM Interface assembler routines</td>
<td>03</td>
</tr>
<tr>
<td>DFHCICS</td>
<td>CSECT</td>
<td>CICS copyright information</td>
<td>03</td>
</tr>
<tr>
<td>DFHCLID</td>
<td>Macro</td>
<td>CICS service-level identifier</td>
<td>11</td>
</tr>
<tr>
<td>DFHCLS3</td>
<td>CSECT (OCO)</td>
<td>APPC signoff transaction program</td>
<td>03</td>
</tr>
<tr>
<td>DFHCLS4</td>
<td>CSECT (OCO)</td>
<td>APPC signon transaction program</td>
<td>03</td>
</tr>
<tr>
<td>DFHCLS5</td>
<td>CSECT (OCO)</td>
<td>Connection Quiesce Protocol</td>
<td>03</td>
</tr>
<tr>
<td>DFHCLLT</td>
<td>Macro</td>
<td>Command list table</td>
<td>11</td>
</tr>
<tr>
<td>DFHCLT1G</td>
<td>Sample</td>
<td>Command list table</td>
<td>19 03</td>
</tr>
<tr>
<td>DFHCMAC</td>
<td>CSECT (OCO)</td>
<td>ME domain - CICS messages and codes transaction (CMAC)</td>
<td>03</td>
</tr>
<tr>
<td>DFHCMACD</td>
<td>Other</td>
<td>Source data file for CMAC transaction</td>
<td>13</td>
</tr>
<tr>
<td>DFHCMAC1</td>
<td>Other</td>
<td>JCL to install the CICS messages data set</td>
<td>02</td>
</tr>
<tr>
<td>DFHCMACU</td>
<td>Other</td>
<td>JCL to update the CICS messages data set</td>
<td>02</td>
</tr>
<tr>
<td>DFHCMA5M</td>
<td>Macro</td>
<td>CPI pseudonym file for assembler</td>
<td>11</td>
</tr>
<tr>
<td>DFHCMC</td>
<td>CSECT (OCO)</td>
<td>CMAC transaction map set (C/370)</td>
<td>02</td>
</tr>
<tr>
<td>DFHCMMCM</td>
<td>CSECT (OCO)</td>
<td>CMAC transaction map set</td>
<td>03</td>
</tr>
<tr>
<td>DFHCMMCOB</td>
<td>CSECT (OCO)</td>
<td>CMAC transaction map set (COBOL)</td>
<td>03</td>
</tr>
<tr>
<td>DFHCMPI</td>
<td>CSECT</td>
<td>CICS monitoring compatibility interface</td>
<td>03</td>
</tr>
<tr>
<td>DFHCMPL1</td>
<td>CSECT (OCO)</td>
<td>CMAC transaction map set (PL/I)</td>
<td>03</td>
</tr>
<tr>
<td>DFHCMNDS</td>
<td>Macro</td>
<td>TCT console control element</td>
<td>11</td>
</tr>
<tr>
<td>DFHCNV</td>
<td>Macro</td>
<td>ISC template definition</td>
<td>11</td>
</tr>
<tr>
<td>DFHCNVCA</td>
<td>DSECT</td>
<td>DFHCNV commarea layout</td>
<td>03</td>
</tr>
<tr>
<td>DFHCNVNE</td>
<td>Macro</td>
<td>DFHCNV data conversion tables</td>
<td>03</td>
</tr>
<tr>
<td>DFHCNVNH</td>
<td>Macro</td>
<td>DFHCNV data conversion tables</td>
<td>03</td>
</tr>
<tr>
<td>DFHCNVXX</td>
<td>Macro</td>
<td>DFHCNV data conversion related</td>
<td>03</td>
</tr>
<tr>
<td>DFHCNV00</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>03</td>
</tr>
<tr>
<td>DFHCNV01</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>03</td>
</tr>
<tr>
<td>DFHCNV02</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>03</td>
</tr>
<tr>
<td>DFHCNV03</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>03</td>
</tr>
<tr>
<td>DFHCNV04</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>03</td>
</tr>
<tr>
<td>DFHCNV05</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>03</td>
</tr>
<tr>
<td>DFHCNV06</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>03</td>
</tr>
<tr>
<td>DFHCNV07</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>03</td>
</tr>
<tr>
<td>DFHCNV08</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>03</td>
</tr>
<tr>
<td>DFHCNV09</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>03</td>
</tr>
<tr>
<td>DFHCNV10</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>03</td>
</tr>
<tr>
<td>DFHCNV11</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>03</td>
</tr>
<tr>
<td>DFHCNV12</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>03</td>
</tr>
<tr>
<td>DFHCNV13</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>03</td>
</tr>
<tr>
<td>DFHCNV14</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>03</td>
</tr>
<tr>
<td>DFHCNV15</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>03</td>
</tr>
<tr>
<td>DFHCNV16</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>03</td>
</tr>
<tr>
<td>DFHCNV17</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>03</td>
</tr>
<tr>
<td>DFHCNV18</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>03</td>
</tr>
<tr>
<td>DFHCNV19</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>03</td>
</tr>
<tr>
<td>DFHCNV20</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>03</td>
</tr>
<tr>
<td>DFHCNV21</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>03</td>
</tr>
<tr>
<td>DFHCNV22</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>03</td>
</tr>
<tr>
<td>DFHCNV23</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>03</td>
</tr>
<tr>
<td>DFHCNV24</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>03</td>
</tr>
<tr>
<td>DFHCNV25</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>03</td>
</tr>
<tr>
<td>DFHCNV26</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>03</td>
</tr>
<tr>
<td>DFHCNV27</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>03</td>
</tr>
<tr>
<td>DFHCNV28</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>03</td>
</tr>
<tr>
<td>DFHCNV29</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>03</td>
</tr>
<tr>
<td>DFHCNV30</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>03</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
<td>Library</td>
</tr>
<tr>
<td>----------</td>
<td>-------</td>
<td>----------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>DFHCNV31</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCNV32</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCNV33</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCNV34</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCNV35</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCNV36</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCNV37</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCNV38</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCNV39</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCNV40</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCNV41</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCNV42</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCNV43</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCNV44</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCNV45</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCNV46</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCNV47</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCNV48</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCNV49</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCNV50</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCNV51</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCNV52</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCNV53</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCNV54</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCNV55</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCNV56</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCNV57</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCNV58</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCNV59</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCNV60</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCNV61</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCNV62</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCNV63</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCNV64</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCNV65</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCNV66</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCNV67</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCNV68</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCNV69</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCNV70</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCNV71</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCNV72</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCNV73</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCNV74</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCNV75</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCNV76</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCNV77</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCNV78</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCNV79</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCNV80</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCNV81</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCNV82</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCNV83</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCNV84</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCNV85</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCNV86</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCNV87</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCNV88</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCNV89</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCNV90</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>OS 03</td>
</tr>
</tbody>
</table>
### CICS directory

**Table 123. CICS modules directory (continued)**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHCN45E</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHCN45F</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHCN46A</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHCN46B</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHCN46E</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHCN46F</td>
<td>CSECT</td>
<td>DFHCNV data conversion tables</td>
<td>OS -</td>
</tr>
<tr>
<td>DHFCOAP</td>
<td>Other</td>
<td>JCL to delete and recreate CICS system data sets common to all regions</td>
<td>OS -</td>
</tr>
<tr>
<td>DHFCOMP</td>
<td>Macro</td>
<td>Generate compare equate values</td>
<td>OS -</td>
</tr>
<tr>
<td>DHFCOVER</td>
<td>Macro</td>
<td>Cover page generator</td>
<td>11 -</td>
</tr>
<tr>
<td>DHCPCARH</td>
<td>CSECT (OCO)</td>
<td>CPIC - Oxxxxx application request handler</td>
<td>03</td>
</tr>
<tr>
<td>DHCPCAC</td>
<td>CSECT (OCO)</td>
<td>CPIC - Accept_Conversation</td>
<td>03</td>
</tr>
<tr>
<td>DHCPCAL</td>
<td>CSECT (OCO)</td>
<td>CPIC - Allocate</td>
<td>03</td>
</tr>
<tr>
<td>DHCPCBA</td>
<td>CSECT (OCO)</td>
<td>CPIC - Create_CPC (Accept)</td>
<td>03</td>
</tr>
<tr>
<td>DHCPCBB</td>
<td>CSECT (OCO)</td>
<td>CPIC - Increment_Last_Convid</td>
<td>03</td>
</tr>
<tr>
<td>DHCPCBD</td>
<td>CSECT (OCO)</td>
<td>CPIC - Delete_Conversation</td>
<td>03</td>
</tr>
<tr>
<td>DHCPCBE</td>
<td>CSECT (OCO)</td>
<td>CPIC - Extract_Syncpoint_rc</td>
<td>03</td>
</tr>
<tr>
<td>DHCPCBG</td>
<td>CSECT (OCO)</td>
<td>CPIC - Initialize_CPC</td>
<td>03</td>
</tr>
<tr>
<td>DHCPCBI</td>
<td>CSECT (OCO)</td>
<td>CPIC - Create_CPC (Initialize)</td>
<td>03</td>
</tr>
<tr>
<td>DHCPCBL</td>
<td>CSECT (OCO)</td>
<td>CPIC - Locate_CPC</td>
<td>03</td>
</tr>
<tr>
<td>DHCPCBS</td>
<td>CSECT (OCO)</td>
<td>CPIC - Set_CPC_Log_Data</td>
<td>03</td>
</tr>
<tr>
<td>DHCPCBT</td>
<td>CSECT (OCO)</td>
<td>CPIC - Load module branch table</td>
<td>03</td>
</tr>
<tr>
<td>DHCPCCA</td>
<td>DSECT</td>
<td>CPCC parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DHCPCCD</td>
<td>CSECT (OCO)</td>
<td>CPIC - Confirmed</td>
<td>03</td>
</tr>
<tr>
<td>DHCPCCF</td>
<td>CSECT (OCO)</td>
<td>CPIC - Confirm</td>
<td>03</td>
</tr>
<tr>
<td>DHCPCCM</td>
<td>Macro</td>
<td>CPCC request</td>
<td>OS -</td>
</tr>
<tr>
<td>DHCPCCT</td>
<td>CSECT (OCO)</td>
<td>CPCC trace interpretation data</td>
<td>03</td>
</tr>
<tr>
<td>DHCPCDE</td>
<td>CSECT (OCO)</td>
<td>CPIC - Deallocate</td>
<td>03</td>
</tr>
<tr>
<td>DHCPCEA</td>
<td>CSECT (OCO)</td>
<td>CPIC - Extract_Conversation_Type</td>
<td>03</td>
</tr>
<tr>
<td>DHCPCEB</td>
<td>CSECT (OCO)</td>
<td>CPIC - Extract_Mode_Name</td>
<td>03</td>
</tr>
<tr>
<td>DHCPCEC</td>
<td>CSECT (OCO)</td>
<td>CPIC - Extract_Partner_LU_Name</td>
<td>03</td>
</tr>
<tr>
<td>DHCPCED</td>
<td>CSECT (OCO)</td>
<td>CPIC - Extract_Sync_Level</td>
<td>03</td>
</tr>
<tr>
<td>DHCPCCE</td>
<td>CSECT (OCO)</td>
<td>CPIC - Extract_Conversation_State</td>
<td>03</td>
</tr>
<tr>
<td>DHCPCFL</td>
<td>CSECT (OCO)</td>
<td>CPIC - Flush</td>
<td>03</td>
</tr>
<tr>
<td>DHCPCFS</td>
<td>CSECT (OCO)</td>
<td>CPIC - finite state machine</td>
<td>03</td>
</tr>
<tr>
<td>DHCPCIC</td>
<td>CSECT (OCO)</td>
<td>CPIC - Initialize_Conversation</td>
<td>03</td>
</tr>
<tr>
<td>DHCPCICL</td>
<td>CSECT (OCO)</td>
<td>CPIC - interface to DFHLUC</td>
<td>03</td>
</tr>
<tr>
<td>DHCPCIM</td>
<td>CSECT (OCO)</td>
<td>CPIC - build send list</td>
<td>03</td>
</tr>
<tr>
<td>DHCPCICL</td>
<td>CSECT (OCO)</td>
<td>DFHLUC to CPIC return code conversion</td>
<td>03</td>
</tr>
<tr>
<td>DHCPCND</td>
<td>CSECT (OCO)</td>
<td>CPIC - Send_Data</td>
<td>03</td>
</tr>
<tr>
<td>DHCPCNE</td>
<td>CSECT (OCO)</td>
<td>CPIC - End_Error</td>
<td>03</td>
</tr>
<tr>
<td>DHCPCN1</td>
<td>CSECT (OCO)</td>
<td>CPIC - Send_and_Buffer</td>
<td>03</td>
</tr>
<tr>
<td>DHCPCN2</td>
<td>CSECT (OCO)</td>
<td>CPIC - Send_and_Flush</td>
<td>03</td>
</tr>
<tr>
<td>DHCPCN3</td>
<td>CSECT (OCO)</td>
<td>CPIC - Send_and_Prep_To_Receive</td>
<td>03</td>
</tr>
<tr>
<td>DHCPCN4</td>
<td>CSECT (OCO)</td>
<td>CPIC - Send_and_Confirm</td>
<td>03</td>
</tr>
<tr>
<td>DHCPCN5</td>
<td>CSECT (OCO)</td>
<td>CPIC - Send_and_Deallocate</td>
<td>03</td>
</tr>
<tr>
<td>DHCPCOJ</td>
<td>CSECT (OCO)</td>
<td>CPIC - Output_Journaling</td>
<td>03</td>
</tr>
<tr>
<td>DHCPCPR</td>
<td>CSECT (OCO)</td>
<td>CPIC - Prepare_To_Receive</td>
<td>03</td>
</tr>
<tr>
<td>DHCPCRA</td>
<td>CSECT (OCO)</td>
<td>CPIC - Receive mapped data</td>
<td>03</td>
</tr>
<tr>
<td>DHCPCRB</td>
<td>CSECT (OCO)</td>
<td>CPIC - Receive GDS header</td>
<td>03</td>
</tr>
<tr>
<td>DHCPCRC</td>
<td>CSECT (OCO)</td>
<td>CPIC - Receive basic data</td>
<td>03</td>
</tr>
<tr>
<td>DHCPCRI</td>
<td>CSECT (OCO)</td>
<td>CPIC - Receive_Immediate</td>
<td>03</td>
</tr>
<tr>
<td>DHCPCS</td>
<td>CSECT (OCO)</td>
<td>CPIC - Request_To_Send</td>
<td>03</td>
</tr>
<tr>
<td>DHCPCRV</td>
<td>CSECT (OCO)</td>
<td>CPIC - Receive</td>
<td>03</td>
</tr>
<tr>
<td>DHCPCRW</td>
<td>CSECT (OCO)</td>
<td>CPIC - Receive_and_Wait</td>
<td>03</td>
</tr>
<tr>
<td>DHCPCS</td>
<td>CSECT (OCO)</td>
<td>CPIC - Set_Conversation_Type</td>
<td>03</td>
</tr>
</tbody>
</table>
Table 123. CICS modules directory (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHPCSB</td>
<td>CSECT (OCO)</td>
<td>CPIC - Set_Deallocate_Type</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHPCSC</td>
<td>CSECT (OCO)</td>
<td>CPIC - Set_Error_Direction</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHPCSD</td>
<td>CSECT (OCO)</td>
<td>CPIC - Set_Fill</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHPCSE</td>
<td>CSECT (OCO)</td>
<td>CPIC - Set_Log_Data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHPCSF</td>
<td>CSECT (OCO)</td>
<td>CPIC - Set_Mode_Name</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHPCSG</td>
<td>CSECT (OCO)</td>
<td>CPIC - Set_Partner_LU_Name</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHPCSH</td>
<td>CSECT (OCO)</td>
<td>CPIC - Set_Prepare_To_Receive</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHPCSI</td>
<td>CSECT (OCO)</td>
<td>CPIC - Set_Receive_Type</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHPCSJ</td>
<td>CSECT (OCO)</td>
<td>CPIC - Set_Return_Control</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHPCSK</td>
<td>CSECT (OCO)</td>
<td>CPIC - Set_Send_Type</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHPCSL</td>
<td>CSECT (OCO)</td>
<td>CPIC - Set_Partner_LU_Name</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHPCSM</td>
<td>CSECT (OCO)</td>
<td>CPIC - Set_TP_Name</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHPCST</td>
<td>CSECT (OCO)</td>
<td>CPIC - Test_Request_To_Send_Received</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHCPCSDF</td>
<td>CSECT (OCO)</td>
<td>SDUMP formatter for CP keyword</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHCPI</td>
<td>CSECT (OCO)</td>
<td>Common programming interface (CPI) program</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHCPIA</td>
<td>DSECT</td>
<td>CPIN parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHCPIAM</td>
<td>Macro</td>
<td>CPIN request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHCPIANT</td>
<td>CSECT (OCO)</td>
<td>CPIN trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHCPIIN1</td>
<td>CSECT (OCO)</td>
<td>CPI initialization management program</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHCPIIN2</td>
<td>CSECT (OCO)</td>
<td>CPI initialization subtask program</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHCPIR</td>
<td>CSECT (OCO)</td>
<td>SRxxxxx application request processor</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHCPLC</td>
<td>CSECT (OCO)</td>
<td>Link-edit stub for application programs using SAA communications interface</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHCPLR</td>
<td>CSECT (OCO)</td>
<td>Link-edit stub for application programs using SAA resource recovery interface</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHCPOST</td>
<td>Macro</td>
<td>POST macro for extended ECBs</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHCPSDS</td>
<td>DSECT</td>
<td>CPI static storage</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHCPSPA</td>
<td>DSECT</td>
<td>CPSP parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHCPSPM</td>
<td>Macro</td>
<td>CPSP request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHCPSPT</td>
<td>CSECT (OCO)</td>
<td>CPSP trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHCPSRH</td>
<td>CSECT (OCO)</td>
<td>CPIC - syncpoint request handler</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHCPTY</td>
<td>CSECT</td>
<td>3270 hard copy support</td>
<td>03 OS</td>
</tr>
<tr>
<td>DFHCRRD</td>
<td>DSECT</td>
<td>CICS region control block</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHCRRU</td>
<td>DSECT</td>
<td>UOW back-to-front processor module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHCRC</td>
<td>DSECT</td>
<td>Interregion abnormal exit module</td>
<td>03 OS</td>
</tr>
<tr>
<td>DFHCRD</td>
<td>DSECT</td>
<td>Communications recovery services declares</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHCRCRI</td>
<td>DSECT</td>
<td>AP domain - Communications recovery management - resync</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHCRCRP</td>
<td>DSECT</td>
<td>Perform unshunt invoked by RM</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHCRESI</td>
<td>DSECT</td>
<td>Session failure during syncpoint</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHCRESI</td>
<td>DSECT</td>
<td>AP domain - communication recovery management</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHCRIU</td>
<td>CSECT</td>
<td>IRC RMC syncpoint event processor</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHCR</td>
<td>CSECT</td>
<td>RMC logging back-to-front processor</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHCR</td>
<td>CSECT</td>
<td>RMC bind time logging for old MRO/LU6.2</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHCR</td>
<td>CSECT</td>
<td>CRLB parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHCRB</td>
<td>CSECT</td>
<td>CRLB translate tables</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHCR</td>
<td>CSECT</td>
<td>Interregion connection manager</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCR</td>
<td>CSECT</td>
<td>ATI purge program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCR</td>
<td>CSECT</td>
<td>Interregion session recovery program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCR</td>
<td>CSECT</td>
<td>Communications resynchronization</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHCR</td>
<td>CSECT</td>
<td>Remote scheduler program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCR</td>
<td>CSECT</td>
<td>CICS IRC startup module</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCR</td>
<td>CSECT</td>
<td>Transaction routing relay program for APPC devices</td>
<td>OS 03</td>
</tr>
</tbody>
</table>
## CICS directory

### Table 123. CICS modules directory (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHCRTTRI</td>
<td>CSECT</td>
<td>Offline trace formatting - interpretation - routine parameter list</td>
<td>03</td>
</tr>
<tr>
<td>DFHCRTU</td>
<td>CSECT</td>
<td>IRC LU61 syncpoint event processor</td>
<td>03</td>
</tr>
<tr>
<td>DFHCT2U</td>
<td>CSECT</td>
<td>IRC LU62 RMC syncpoint event processor</td>
<td>03</td>
</tr>
<tr>
<td>DFHCSCA</td>
<td>CSECT</td>
<td>Common system area</td>
<td>OS</td>
</tr>
<tr>
<td>DFHCSDAD</td>
<td>Macro</td>
<td>Common system area definition</td>
<td>11</td>
</tr>
<tr>
<td>DFHCSDDS</td>
<td>DSECT</td>
<td>Common system area definition</td>
<td>11</td>
</tr>
<tr>
<td>DFHCSCDS</td>
<td>Symbolic</td>
<td>CICS SVC startup return codes</td>
<td>OS</td>
</tr>
<tr>
<td>DFHCSCDF</td>
<td>CSECT</td>
<td>SDUMP formatter for CSA and CSA optional features list</td>
<td>OS</td>
</tr>
<tr>
<td>DFHCSCVC</td>
<td>CSECT</td>
<td>CICS SVC startup</td>
<td>OS</td>
</tr>
<tr>
<td>DFHCCTHF</td>
<td>CSECT</td>
<td>CETR transaction help screens map set</td>
<td>OS</td>
</tr>
<tr>
<td>DFHCCTRM</td>
<td>CSECT</td>
<td>CETR transaction main screens map set</td>
<td>OS</td>
</tr>
<tr>
<td>DFHCCTRMU</td>
<td>Sample</td>
<td></td>
<td>-19</td>
</tr>
<tr>
<td>DFHCUCAD</td>
<td>CSECT</td>
<td>CSDUP - add command</td>
<td>OS</td>
</tr>
<tr>
<td>DFHGUALG</td>
<td>CSECT</td>
<td>RDO off-line generic alter utility program</td>
<td>OS</td>
</tr>
<tr>
<td>DFHUGAL</td>
<td>CSECT</td>
<td>CSDUP - alter command</td>
<td>OS</td>
</tr>
<tr>
<td>DFHUGPP</td>
<td>CSECT</td>
<td>CSDUP - append command</td>
<td>OS</td>
</tr>
<tr>
<td>DFHUCAB</td>
<td>CSECT</td>
<td>CSDUP - command analyzer (DFHUCUA)</td>
<td>OS</td>
</tr>
<tr>
<td>DFHUCAC</td>
<td>CSECT</td>
<td>CSD manager - return and reason codes</td>
<td>OS</td>
</tr>
<tr>
<td>DFHUCBD</td>
<td>CSECT</td>
<td>CSDUP - command builder</td>
<td>OS</td>
</tr>
<tr>
<td>DFHUCBC</td>
<td>CSECT</td>
<td>CSDUP - RDL command locator (DFHUCCB)</td>
<td>OS</td>
</tr>
<tr>
<td>DFHUCD</td>
<td>CSECT</td>
<td>CSDUP - default values (DFHUCDC)</td>
<td>OS</td>
</tr>
<tr>
<td>DFHUCDC</td>
<td>CSECT</td>
<td>CSD manager - return and reason codes</td>
<td>OS</td>
</tr>
<tr>
<td>DFHUCOG</td>
<td>CSECT</td>
<td>CSDUP - generic copy command</td>
<td>OS</td>
</tr>
<tr>
<td>DFHUCOM</td>
<td>CSECT</td>
<td></td>
<td>OS</td>
</tr>
<tr>
<td>DFHUCOP</td>
<td>CSECT</td>
<td>CSDUP - copy command</td>
<td>OS</td>
</tr>
<tr>
<td>DFHUCP</td>
<td>CSECT</td>
<td>CSDUP - command processor</td>
<td>OS</td>
</tr>
<tr>
<td>DFHUCS</td>
<td>CSECT</td>
<td>CSDUP - CSD open and close</td>
<td>OS</td>
</tr>
<tr>
<td>DFHUCSE</td>
<td>CSECT</td>
<td>CSDUP - CSD error check routine</td>
<td>OS</td>
</tr>
<tr>
<td>DFHUCVC</td>
<td>CSECT</td>
<td>CSDUP - command validation</td>
<td>OS</td>
</tr>
<tr>
<td>DFHUCDEF</td>
<td>CSECT</td>
<td>CSDUP - define command</td>
<td>OS</td>
</tr>
<tr>
<td>DFHUCERAD</td>
<td>CSECT</td>
<td>CSDUP - delete/erase command</td>
<td>OS</td>
</tr>
<tr>
<td>DFHUCFA</td>
<td>CSECT</td>
<td>Offline utilities - free automatic storage</td>
<td>OS</td>
</tr>
<tr>
<td>DFHUCFAM</td>
<td>Macro</td>
<td>Offline DFHPROC - free automatic storage</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHUCGA</td>
<td>CSECT</td>
<td>Offline utilities - get automatic storage</td>
<td>OS</td>
</tr>
<tr>
<td>DFHUCGAM</td>
<td>Macro</td>
<td>Offline DFHPROC - get automatic storage</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHCUNI</td>
<td>CSECT</td>
<td>CSDUP - initialize command</td>
<td>OS</td>
</tr>
<tr>
<td>DFHCULIS</td>
<td>CSECT</td>
<td>CSDUP - extract and list commands</td>
<td>OS</td>
</tr>
<tr>
<td>DFHCULOC</td>
<td>CSECT</td>
<td>CSDUP - lock/unlock routine</td>
<td>OS</td>
</tr>
<tr>
<td>DFHCUMDO2</td>
<td>CSECT</td>
<td></td>
<td>OS</td>
</tr>
<tr>
<td>DFHCUMIG</td>
<td>CSECT</td>
<td>CSDUP - migrate command</td>
<td>OS</td>
</tr>
<tr>
<td>DFHCUMT</td>
<td>CSECT</td>
<td>CSDUP - TCT migration</td>
<td>OS</td>
</tr>
<tr>
<td>DFHCUMTD</td>
<td>CSECT</td>
<td>RDO migration utility program for the DCT</td>
<td>OS</td>
</tr>
<tr>
<td>DFHCUMTS</td>
<td>CSECT</td>
<td>RDO migration utility program for the TST</td>
<td>OS</td>
</tr>
<tr>
<td>DFHCUMWR</td>
<td>CSECT</td>
<td>CSDUP - CSD record write routine</td>
<td>OS</td>
</tr>
<tr>
<td>DFHCUMXI</td>
<td>CSECT</td>
<td>SPI offline utility for handling cross reference of IBM groups</td>
<td>OS</td>
</tr>
<tr>
<td>DFHCPURRC</td>
<td>CSECT</td>
<td>RDO off line utility</td>
<td>OS</td>
</tr>
<tr>
<td>DFHCUPROC</td>
<td>CSECT</td>
<td>CSDUP - CSD upgrade routine</td>
<td>OS</td>
</tr>
<tr>
<td>DFHCUROD</td>
<td>CSECT</td>
<td>CSD utilities - delete all existing CICS-supplied groups from previous releases</td>
<td>OS</td>
</tr>
<tr>
<td>DFHCUROI</td>
<td>CSECT</td>
<td>CSD utilities - RDL for basic initialize</td>
<td>OS</td>
</tr>
<tr>
<td>DFHCUROM</td>
<td>CSECT</td>
<td>CSD utilities - RDL for maintenance</td>
<td>OS</td>
</tr>
<tr>
<td>DFHCUROD</td>
<td>CSECT</td>
<td>CSD utilities - RDL for sample definitions</td>
<td>OS</td>
</tr>
<tr>
<td>DFHCUROX</td>
<td>CSECT</td>
<td>CSD utilities - RDL for compatibility gp</td>
<td>OS</td>
</tr>
<tr>
<td>DFHCUREM</td>
<td>CSECT</td>
<td>CSDUP - remove command</td>
<td>OS</td>
</tr>
</tbody>
</table>

1436 CICS Transaction Server: CICS Diagnosis Reference
Table 123. CICS modules directory (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHCURUG</td>
<td>CSECT</td>
<td>CSDUP - upgrade command</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCUSER</td>
<td>CSECT</td>
<td>CSDUP - service command</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCUSHL</td>
<td>CSECT</td>
<td>CSDUP - short lock/unlock routine</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCUSL</td>
<td>CSECT</td>
<td>CSD utilities - sample service request</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCUSR</td>
<td>CSECT</td>
<td></td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCUSER</td>
<td>CSECT</td>
<td>CSDUP - verify command</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCUXR</td>
<td>CSECT</td>
<td>RDO offline utility for building</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCVDAA</td>
<td>Symbolic</td>
<td>System programming command cvda names</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHVCTR1</td>
<td>CSECT</td>
<td>CCNV Gate trace interpretation</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHCZTRI</td>
<td>CSECT</td>
<td>CICS Foundation Classes trace interpretation</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHCZTRT</td>
<td>CSECT</td>
<td>Foundation classes trace interpreter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHCWTO</td>
<td>CSECT</td>
<td>Write to console operator program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCXCU</td>
<td>CSECT</td>
<td>XRF catch-up transaction</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHCSTR1</td>
<td>CSECT</td>
<td>Trace interpreter for DFHCLS3 trace points</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHCSTR1</td>
<td>CSECT</td>
<td>- 03</td>
<td></td>
</tr>
<tr>
<td>DFhhDATE</td>
<td>Macro</td>
<td>Date formatting</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHDBAT</td>
<td>CSECT</td>
<td>CICS-DBCTL adapter/transformer</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDBCON</td>
<td>CSECT</td>
<td>CICS-DBCTL connection program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDBCR</td>
<td>CSECT</td>
<td>CICS-DBCTL XRF tracking program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDBCT</td>
<td>CSECT</td>
<td>CICS-DBCTL control program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDBCTX</td>
<td>CSECT</td>
<td>CICS-DBCTL control exit</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDBOE</td>
<td>CSECT</td>
<td>CICS-DBCTL operator transaction map set</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHDBDI</td>
<td>CSECT</td>
<td>CICS-DBCTL disable program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDBDP</td>
<td>CSECT</td>
<td>CICS-DBCTL disconnection program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDBSF</td>
<td>CSECT</td>
<td>CICS-DBCTL disconnection program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDBSFU</td>
<td>CSECT</td>
<td>CICS-DBCTL SDUMP formatter</td>
<td>OS - 03</td>
</tr>
<tr>
<td>DFHDDBI</td>
<td>CSECT</td>
<td>CICS-DBCTL inquiry screens map set</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDBI</td>
<td>CSECT</td>
<td>CICS-DBCTL inquiry screens map set</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHDBI</td>
<td>CSECT</td>
<td>CICS-DBCTL inquiry program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDBI</td>
<td>CSECT</td>
<td>CICS-DBCTL menu program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDBI</td>
<td>CSECT</td>
<td>CICS-DBCTL monitoring exit</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDBI</td>
<td>CSECT</td>
<td>EDF browse map set</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHDBI</td>
<td>CSECT</td>
<td>EDF browse map set</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDBI</td>
<td>CSECT</td>
<td>CICS-DBCTL menu screens map set</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDBI</td>
<td>CSECT</td>
<td>CICS-DBCTL menu screens map set</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHDBI</td>
<td>CSECT</td>
<td>Dynamic backout program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDBI</td>
<td>CSECT</td>
<td>CICS-DBCTL resume exit</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDBI</td>
<td>CSECT</td>
<td>CICS-DBCTL suspend exit</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDBI</td>
<td>CSECT</td>
<td>CICS-DBCTL status exit</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDBI</td>
<td>CSECT</td>
<td>CICS-DBCTL statistics exit</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDBI</td>
<td>CSECT</td>
<td>EXEC DLI LD table</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDBI</td>
<td>CSECT</td>
<td>CICS-DBCTL token exit</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDBI</td>
<td>CSECT</td>
<td>COMAREA passed to DFHDBUEX</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHDBI</td>
<td>CSECT</td>
<td>DBCTL unsolicited statistics</td>
<td>11 07</td>
</tr>
<tr>
<td>DFHDBI</td>
<td>CSECT</td>
<td>DBCTL unsolicited statistics</td>
<td>C2 -</td>
</tr>
<tr>
<td>DFHDBI</td>
<td>CSECT</td>
<td>User-replaceable CICS-DBCTL exit</td>
<td>19 03</td>
</tr>
<tr>
<td>DFHDCI</td>
<td>Macro</td>
<td>Macro</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHDCI</td>
<td>CSECT</td>
<td>Transaction dump macro-compatibility program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDCI</td>
<td>CSECT</td>
<td>Transaction dump control record format</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHDCI</td>
<td>CSECT</td>
<td>Destination control table</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHDCI</td>
<td>CSECT</td>
<td>Destination control table</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHDCI</td>
<td>CSECT</td>
<td>Destination control table</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHDCI</td>
<td>CSECT</td>
<td>DD domain - browse Services</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHDCI</td>
<td>CSECT</td>
<td>DDBR trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHDCI</td>
<td>CSECT</td>
<td>DD domain - directory services</td>
<td>- 03</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
<td>Library</td>
</tr>
<tr>
<td>------------</td>
<td>---------</td>
<td>--------------------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>DFHDDDIA</td>
<td>CSECT (OCO)</td>
<td>DDDI parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHDDDIM</td>
<td>CSECT (OCO)</td>
<td>DDDI parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHDDDIT</td>
<td>CSECT (OCO)</td>
<td>DDDI trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHDDDMM</td>
<td>CSECT (OCO)</td>
<td>DD domain - domain services</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHDDDUX</td>
<td>CSECT (OCO)</td>
<td>DD domain - dump browse services</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHDDDUF</td>
<td>CSECT (OCO)</td>
<td>DD domain - dump formatting</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHDDUX</td>
<td>CSECT (OCO)</td>
<td>DD domain - locate service</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHDDLOA</td>
<td>CSECT (OCO)</td>
<td>DDLO parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHDDLOM</td>
<td>CSECT (OCO)</td>
<td>DDLO parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHDDLOT</td>
<td>CSECT (OCO)</td>
<td>DDLO trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHDTRI</td>
<td>CSECT (OCO)</td>
<td>DD domain - trace interpretation</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHDTRIX</td>
<td>CSECT (OCO)</td>
<td>DCE services domain - communications exit</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHDEOM</td>
<td>CSECT (OCO)</td>
<td>DCE services domain - services</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHDEDUF</td>
<td>CSECT (OCO)</td>
<td>DCE service domain - dump formatting</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHDEFDS</td>
<td>Other</td>
<td>JCL to delete and recreate CICS system</td>
<td>02 -</td>
</tr>
<tr>
<td>DFHDEIS</td>
<td>CSECT (OCO)</td>
<td>DCE services domain - inquire/set gate</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHDEIST</td>
<td>CSECT</td>
<td>DESIS trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHDEREX</td>
<td>CSECT</td>
<td>DCE services domain - resume exit</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHDESST</td>
<td>DSECT</td>
<td>DCE services domain - system services trace</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHDEST</td>
<td>DSECT</td>
<td>DCE services domain - stub</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHDESV</td>
<td>CSECT</td>
<td>DCE services domain - general services gate</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHDEST</td>
<td>DSECT</td>
<td>DESV trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHDETRI</td>
<td>DSECT</td>
<td>DCE services domain - trace interpreter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHDFST</td>
<td>CSECT</td>
<td>Document Handler Domain</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHDDH</td>
<td>CSECT</td>
<td>Document Handler Domain</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHDDDM</td>
<td>CSECT</td>
<td>Document Handler Domain</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHDDUF</td>
<td>CSECT</td>
<td>Document System Dump Formatter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHDEI</td>
<td>CSECT</td>
<td>DH Document Template EXEC resources</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHDHPB</td>
<td>CSECT</td>
<td>Document Handler Domain</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHDHPD</td>
<td>CSECT</td>
<td>Document Handler Domain</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHDHPM</td>
<td>CSECT</td>
<td>Document Handler Domain</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHDHPR</td>
<td>CSECT</td>
<td>DH Document Handler Read PDS routine</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHDHPS</td>
<td>CSECT</td>
<td>Document Handler Domain</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHDHPT</td>
<td>CSECT</td>
<td>Document Handler Domain</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHDHPU</td>
<td>CSECT</td>
<td>Document Handler Domain</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHDHPX</td>
<td>CSECT</td>
<td>Document Handler Domain</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHDHRM</td>
<td>CSECT</td>
<td>DHARM CDURUN and Gate module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHDHRP</td>
<td>CSECT</td>
<td>Document Handler Recovery Program</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHDHRPT</td>
<td>CSECT</td>
<td>Document Handler Domain</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHDHSL</td>
<td>CSECT</td>
<td>Document Handler Domain</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHDHSLT</td>
<td>CSECT</td>
<td>Document Handler Domain</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHDHTM</td>
<td>CSECT</td>
<td>DH Document Handler Template Manager</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHDHTMT</td>
<td>CSECT</td>
<td>DH Document Handler Template Manager</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHDHTRI</td>
<td>CSECT</td>
<td>DH Domain Trace Formatter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHDHTXD</td>
<td>CSECT</td>
<td>Document Domain (DH) user Exit Services</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHDHTXH</td>
<td>CSECT</td>
<td>Document Domain (DH) user Exit Services</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHDHTXL</td>
<td>CSECT</td>
<td>Document Domain (DH) user Exit Services</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHDHTXO</td>
<td>CSECT</td>
<td>Document Domain (DH) user Exit Services</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHDHUE</td>
<td>CSECT</td>
<td>Data interchange request</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHDIBDS</td>
<td>Macro</td>
<td>Data interchange</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHDIBS</td>
<td>Macro</td>
<td>Data interchange</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHDIPD</td>
<td>CSECT</td>
<td>Data interchange program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDTIODS</td>
<td>CSECT</td>
<td>Data interchange program (dummy)</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDITOP</td>
<td>Macro</td>
<td>Data interchange internal macro</td>
<td>OS -</td>
</tr>
</tbody>
</table>
### Table 123. CICS modules directory (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHDKCR</td>
<td>CSECT</td>
<td>DCE services domain - APPC DES-based authentication</td>
<td>03</td>
</tr>
<tr>
<td>DFHDKDUF</td>
<td>CSECT</td>
<td>DCE service domain - table management dump formatting</td>
<td>03</td>
</tr>
<tr>
<td>DFHDKMR</td>
<td>CSECT</td>
<td>DCE services domain - table manager</td>
<td>03</td>
</tr>
<tr>
<td>DFHDKMRA</td>
<td>DSECT</td>
<td>DKMR parameter list</td>
<td>OS</td>
</tr>
<tr>
<td>DFHDKMRRM</td>
<td>Macro</td>
<td>DKMR request</td>
<td>OS</td>
</tr>
<tr>
<td>DFHDKMRT</td>
<td>CSECT</td>
<td>DKMR trace interpretation data</td>
<td>03</td>
</tr>
<tr>
<td>DFHDKTRI</td>
<td>CSECT</td>
<td>DD domain - trace interpreter</td>
<td>03</td>
</tr>
<tr>
<td>DFHDLI</td>
<td>CSECT</td>
<td>DL/I call router</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDLIAI</td>
<td>CSECT</td>
<td>Application interface for DL/I</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDLIDP</td>
<td>CSECT</td>
<td>DBCTL call processor</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDLRIP</td>
<td>CSECT</td>
<td>DL/I remote call processor</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDL00</td>
<td>CSECT</td>
<td></td>
<td>03</td>
</tr>
<tr>
<td>DFHDLT</td>
<td>Macro</td>
<td>CICS-DL/I interface</td>
<td>11</td>
</tr>
<tr>
<td>DFHDLPSB</td>
<td>Macro</td>
<td>Generate DL/I PSB directory list</td>
<td>11</td>
</tr>
<tr>
<td>DFHDLXDF</td>
<td>CSECT</td>
<td>DU domain - transaction dump formatter for DL/I related areas</td>
<td>03</td>
</tr>
<tr>
<td>DFHDMDM</td>
<td>CSECT</td>
<td>DM domain - domain initialization/quiesce</td>
<td>03</td>
</tr>
<tr>
<td>DFHDMDMIA</td>
<td>DSECT</td>
<td>DMDM parameter list</td>
<td>OS</td>
</tr>
<tr>
<td>DFHDMT</td>
<td>Macro</td>
<td>DMDM request</td>
<td>OS</td>
</tr>
<tr>
<td>DFHDMTDM</td>
<td>CSECT</td>
<td>DMDM trace interpretation data</td>
<td>03</td>
</tr>
<tr>
<td>DFHDMDS</td>
<td>CSECT</td>
<td>DM domain - task reply handler</td>
<td>03</td>
</tr>
<tr>
<td>DFHDMDFU</td>
<td>CSECT</td>
<td>SDUMP formatter for DM domain</td>
<td>03</td>
</tr>
<tr>
<td>DFHDMEN</td>
<td>CSECT</td>
<td>Domain manager ENF support</td>
<td>03</td>
</tr>
<tr>
<td>DFHDMENF</td>
<td>CSECT</td>
<td>Domain manager event notification routine</td>
<td>03</td>
</tr>
<tr>
<td>DFHDMEMES</td>
<td>CSECT</td>
<td>CICS ENF SRBEXIT</td>
<td>03</td>
</tr>
<tr>
<td>DFHDMENT</td>
<td>CSECT</td>
<td>DMEN translation tables</td>
<td>03</td>
</tr>
<tr>
<td>DFHDMIQ</td>
<td>CSECT</td>
<td>DM domain - browse and inquiry</td>
<td>03</td>
</tr>
<tr>
<td>DFHDMIOQA</td>
<td>DSECT</td>
<td>DMIQ parameter list</td>
<td>OS</td>
</tr>
<tr>
<td>DFHDMIOMQ</td>
<td>Macro</td>
<td>DMIQ request</td>
<td>OS</td>
</tr>
<tr>
<td>DFHDMIQT</td>
<td>CSECT</td>
<td>DMIQ trace interpretation data</td>
<td>03</td>
</tr>
<tr>
<td>DFHDMDB</td>
<td>CSECT</td>
<td>CSDUP - definition file (CSD) manager, batch environment router (DFHDMDB)</td>
<td>03</td>
</tr>
<tr>
<td>DFHDMDBA</td>
<td>CSECT</td>
<td>CSDUP - batch environment adapter</td>
<td>03</td>
</tr>
<tr>
<td>DFHDMPC</td>
<td>CSECT</td>
<td>CSD manager - CICS environment router</td>
<td>03</td>
</tr>
<tr>
<td>DFHDMPCA</td>
<td>CSECT</td>
<td>CSD manager - CICS environment adapter</td>
<td>03</td>
</tr>
<tr>
<td>DFHDMPH</td>
<td>Symbolic</td>
<td>DM domain - phase definitions</td>
<td>OS</td>
</tr>
<tr>
<td>DFHDMR</td>
<td>CSECT</td>
<td>CSD manager - CSD close routine</td>
<td>03</td>
</tr>
<tr>
<td>DFHDSVC</td>
<td>CSECT</td>
<td>DM domain - SVC processing routine</td>
<td>03</td>
</tr>
<tr>
<td>DFHDMTRI</td>
<td>CSECT</td>
<td>DM domain - trace interpreter</td>
<td>03</td>
</tr>
<tr>
<td>DFHDMWQ</td>
<td>CSECT</td>
<td>DM domain - wait queue subroutine</td>
<td>03</td>
</tr>
<tr>
<td>DFHDMWQA</td>
<td>DSECT</td>
<td>DMWQ parameter list</td>
<td>OS</td>
</tr>
<tr>
<td>DFHDMWQM</td>
<td>Macro</td>
<td>DMWQ request</td>
<td>OS</td>
</tr>
<tr>
<td>DFHDMWQT</td>
<td>CSECT</td>
<td>DMWQ trace interpretation data</td>
<td>03</td>
</tr>
<tr>
<td>DFHDM01B</td>
<td>CSECT</td>
<td>CSDUP - connect (DFHDM01 batch)</td>
<td>03</td>
</tr>
<tr>
<td>DFHDM01C</td>
<td>CSECT</td>
<td>CSD manager - connect (DFHDM01 CICS)</td>
<td>03</td>
</tr>
<tr>
<td>DFHDM02B</td>
<td>CSECT</td>
<td>CSDUP - disconnect (DFHDM02 batch)</td>
<td>03</td>
</tr>
<tr>
<td>DFHDM02C</td>
<td>CSECT</td>
<td>CSD manager - disconnect (DFHDM02 CICS)</td>
<td>03</td>
</tr>
<tr>
<td>DFHDM03B</td>
<td>CSECT</td>
<td>CSDUP - write (DFHDM03 batch)</td>
<td>03</td>
</tr>
<tr>
<td>DFHDM03C</td>
<td>CSECT</td>
<td>CSD manager - write (DFHDM03 CICS)</td>
<td>03</td>
</tr>
<tr>
<td>DFHDM04B</td>
<td>CSECT</td>
<td>CSDUP - read (DFHDM04 batch)</td>
<td>03</td>
</tr>
<tr>
<td>DFHDM04C</td>
<td>CSECT</td>
<td>CSD manager - read (DFHDM04 CICS)</td>
<td>03</td>
</tr>
<tr>
<td>DFHDM05B</td>
<td>CSECT</td>
<td>CSDUP - delete (DFHDM05 batch)</td>
<td>03</td>
</tr>
<tr>
<td>DFHDM05C</td>
<td>CSECT</td>
<td>CSD manager - delete (DFHDM05 CICS)</td>
<td>03</td>
</tr>
<tr>
<td>DFHDM06B</td>
<td>CSECT</td>
<td>CSDUP - lock/unlock (DFHDM06 batch)</td>
<td>03</td>
</tr>
</tbody>
</table>
## CICS directory

### Table 123. CICS modules directory (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHDM06C</td>
<td>CSECT</td>
<td>CSD manager - lock/unlock (DFHDM06 CICS)</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDM08B</td>
<td>CSECT</td>
<td>CSUP - setbrowse (DFHDM08 batch)</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDM08C</td>
<td>CSECT</td>
<td>CSD manager - setbrowse (DFHDM08 CICS)</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDM09B</td>
<td>CSECT</td>
<td>CSUP - getnext (DFHDM09 batch)</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDM09C</td>
<td>CSECT</td>
<td>CSD manager - getnext (DFHDM09 CICS)</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDM10B</td>
<td>CSECT</td>
<td>CSUP - endbrowse (DFHDM10 batch)</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDM10C</td>
<td>CSECT</td>
<td>CSD manager - endbrowse (DFHDM10 CICS)</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDM11B</td>
<td>CSECT</td>
<td>CSUP - createset (DFHDM11 batch)</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDM11C</td>
<td>CSECT</td>
<td>CSD manager - createset (DFHDM11 CICS)</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDM12B</td>
<td>CSECT</td>
<td>CSUP - eraseset (DFHDM12 batch only)</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDM13B</td>
<td>CSECT</td>
<td>CSUP - queryset (DFHDM13 batch)</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDM13C</td>
<td>CSECT</td>
<td>CSD manager - queryset (DFHDM13 CICS)</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDM15B</td>
<td>CSECT</td>
<td>CSUP - read/write control records (DFHDM15 batch)</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDM15C</td>
<td>CSECT</td>
<td>CSD manager - read/write control records (DFHDM15 CICS)</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDM16B</td>
<td>CSECT</td>
<td>CSUP - buildkey (DFHDM16 batch)</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDM16C</td>
<td>CSECT</td>
<td>CSD manager - buildkey (DFHDM16 CICS)</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDM17B</td>
<td>CSECT</td>
<td>CSUP - relsekwa (DFHDM17 batch)</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDM17C</td>
<td>CSECT</td>
<td>CSD manager - relsekwa (DFHDM17 CICS)</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDM18B</td>
<td>CSECT</td>
<td>CSUP - tokenize utilities (DFHDM18 batch)</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDM18C</td>
<td>CSECT</td>
<td>CSD manager - tokenize utilities (DFHDM18 CICS)</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDM19B</td>
<td>CSECT</td>
<td>CSUP - free generic tokens chain (DFHDM19 batch)</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDM19C</td>
<td>CSECT</td>
<td>CSD manager - free generic tokens chain (DFHDM19 CICS)</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDM21B</td>
<td>CSECT</td>
<td>CSUP - generic qualification (DFHDM21 batch)</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDM21C</td>
<td>CSECT</td>
<td>CSD manager - generic qualification (DFHDM21 CICS)</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDM22B</td>
<td>CSECT</td>
<td>CSUP - resequence utility (DFHDM22 batch)</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDM22C</td>
<td>CSECT</td>
<td>CSD manager - resequence utility (DFHDM22 CICS)</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDM23B</td>
<td>CSECT</td>
<td>CSUP - verify key work area (DFHDM23 batch)</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDM23C</td>
<td>CSECT</td>
<td>CSD manager - verify key work area (DFHDM23 CICS)</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDNSRT</td>
<td>Macro</td>
<td>Internal index sorting macro</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHDRX</td>
<td>Macro</td>
<td>DL/I resource table</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHDSAT</td>
<td>CSECT (OCO)</td>
<td>DS domain - attach, change mode, change/set priority, cancel task</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDSDTA</td>
<td>DSECT</td>
<td>DSAT parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHDSDATM</td>
<td>Macro</td>
<td>DSAT request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHDSATT</td>
<td>CSECT (OCO)</td>
<td>DSAT trace interpretation data</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDSATX</td>
<td>Macro</td>
<td>DSAT request (XPI)</td>
<td>OS 11</td>
</tr>
<tr>
<td>DFHDSDATY</td>
<td>DSECT</td>
<td>DSAT parameter list (XPI)</td>
<td>OS 11</td>
</tr>
<tr>
<td>DFHDSAUT</td>
<td>CSECT (OCO)</td>
<td>DS domain - authorized services</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDSD</td>
<td>CSECT</td>
<td>BMS data stream build</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHDSSA</td>
<td>CSECT</td>
<td>BMS data stream build (standard)</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDSCR</td>
<td>CSECT (OCO)</td>
<td>DS domain - browse, inquire task</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDSSRA</td>
<td>DSECT</td>
<td>DSBR parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHDSSRM</td>
<td>Macro</td>
<td>DSBR request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHDSSRT</td>
<td>CSECT (OCO)</td>
<td>DSBR trace interpretation data</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDSSU</td>
<td>CSECT</td>
<td>BMS data stream build (full)</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDSCPX</td>
<td>CSECT (OCO)</td>
<td>POST routine for DS WAIT_MVS requests</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDSCSA</td>
<td>CSECT (OCO)</td>
<td>DS domain - update CSA on task dispatch</td>
<td>OS 03</td>
</tr>
</tbody>
</table>
Table 123. CICS modules directory (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHDSRM</td>
<td>Macro</td>
<td>DS domain - storage notify handler</td>
<td>03</td>
</tr>
<tr>
<td>DFHDSST</td>
<td>CSECT</td>
<td>DS domain - statistics collection</td>
<td>03</td>
</tr>
<tr>
<td>DFHDSSTAX</td>
<td>CSECT</td>
<td>DS domain - STIMERM exit</td>
<td>03</td>
</tr>
<tr>
<td>DFHDSSTCB</td>
<td>CSECT</td>
<td>DS domain - KEDS TCB_REPLY handler</td>
<td>03</td>
</tr>
<tr>
<td>DFHDSSTIQ</td>
<td>Macro</td>
<td>DS domain - obtain domain index of task</td>
<td>03</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CSECT</td>
<td>DS domain - Trace interpreter</td>
<td>03</td>
</tr>
<tr>
<td></td>
<td>DSECT</td>
<td>DS domain - Task Area</td>
<td>03</td>
</tr>
<tr>
<td></td>
<td>CSECT</td>
<td>DS domain - enable/disable user exits</td>
<td>03</td>
</tr>
<tr>
<td></td>
<td>CSECT</td>
<td>Shared data tables connect file PC</td>
<td>03</td>
</tr>
<tr>
<td></td>
<td>CSECT</td>
<td>Shared data tables cell pool management</td>
<td>03</td>
</tr>
<tr>
<td></td>
<td>CSECT</td>
<td>Shared data tables connection validation</td>
<td>03</td>
</tr>
<tr>
<td></td>
<td>CSECT</td>
<td>Shared data tables data space and ALET code</td>
<td>03</td>
</tr>
<tr>
<td></td>
<td>CSECT</td>
<td>Shared data tables data management</td>
<td>03</td>
</tr>
<tr>
<td></td>
<td>CSECT</td>
<td>Shared data tables initialization</td>
<td>03</td>
</tr>
<tr>
<td></td>
<td>CSECT</td>
<td>Shared data tables index management</td>
<td>03</td>
</tr>
<tr>
<td></td>
<td>CSECT</td>
<td>Shared data tables load attach</td>
<td>03</td>
</tr>
<tr>
<td></td>
<td>CSECT</td>
<td>Shared data tables local initialization</td>
<td>03</td>
</tr>
<tr>
<td></td>
<td>CSECT</td>
<td>Shared data tables load transaction</td>
<td>03</td>
</tr>
<tr>
<td></td>
<td>DSECT</td>
<td>Data tables - services interface block</td>
<td>03</td>
</tr>
<tr>
<td></td>
<td>CSECT</td>
<td>Shared data tables program call stub</td>
<td>03</td>
</tr>
<tr>
<td></td>
<td>CSECT</td>
<td>Shared data tables remote file connection and disconnection</td>
<td>03</td>
</tr>
<tr>
<td></td>
<td>CSECT</td>
<td>Shared data tables remote file connection and disconnection</td>
<td>03</td>
</tr>
</tbody>
</table>
### CICS directory

**Table 123. CICS modules directory (continued)**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHDTRI</td>
<td>CSECT (OCO)</td>
<td>Shared data tables remote environment initialization</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHDTRM</td>
<td>CSECT (OCO)</td>
<td>Shared data tables record management</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHDTRR</td>
<td>CSECT (OCO)</td>
<td>Shared data tables remote retrieval</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHDTSR</td>
<td>CSECT (OCO)</td>
<td>Shared data tables shared retrieval</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHDTS</td>
<td>CSECT (OCO)</td>
<td>Shared data table server status</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHDTSV</td>
<td>CSECT (OCO)</td>
<td>Shared data tables SVC services</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHDTSV</td>
<td>CSECT (OCO)</td>
<td>Shared data tables update and syncpoint services</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHDTSX</td>
<td>CSECT (OCO)</td>
<td>Shared data tables connection security</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHDUDDA</td>
<td>DSECT</td>
<td>DUDD parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHDUDDM</td>
<td>Macro</td>
<td>DUDD request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHDUDDT</td>
<td>CSECT</td>
<td>DUDD trace interpretation data</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDUDM</td>
<td>CSECT</td>
<td>DU domain - initialization/termination</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDUDT</td>
<td>CSECT</td>
<td>DU domain - dump table services</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDUDTA</td>
<td>DSECT</td>
<td>DUDT parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHDUDTM</td>
<td>Macro</td>
<td>DUDT request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHDUDTT</td>
<td>CSECT</td>
<td>DUOT trace interpretation data</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDUDU</td>
<td>CSECT</td>
<td>DU domain - take system/transaction dump</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDUDUA</td>
<td>DSECT</td>
<td>DUUD parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHDUDUF</td>
<td>CSECT (OCO)</td>
<td>SDUMP formatter for DU domain</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHDUDUM</td>
<td>Macro</td>
<td>DUU request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHDUDUT</td>
<td>CSECT</td>
<td>DUOT trace interpretation data</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDUDUX</td>
<td>Macro</td>
<td>DUU request (XPI)</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHDUDUY</td>
<td>DSECT</td>
<td>DUUD parameter list (XPI)</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHDUF</td>
<td>CSECT (OCO)</td>
<td>SDUMP formatting router</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHDUFET</td>
<td>CSECT (OCO)</td>
<td>PRDUMP formatter - service functions</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDUFET</td>
<td>CSECT (OCO)</td>
<td>Dump domain services</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDUFTA</td>
<td>DSECT</td>
<td>DUTF parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHDUFTD</td>
<td>DSECT</td>
<td>DUTF formatting routines parameter declares</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHDUFTM</td>
<td>Macro</td>
<td>DUTF macro</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHDUFTT</td>
<td>DSECT (OCO)</td>
<td>DUTF translate tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDUFTX</td>
<td>Macro</td>
<td>DUTF macro</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHDUFTY</td>
<td>DSECT</td>
<td>DUTF call structured parameter list</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHDUFUT</td>
<td>DSECT (OCO)</td>
<td>SDUMP formatting - service functions</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHDUO10</td>
<td>CSECT</td>
<td>DU domain - open/close/switch/write</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDUO10</td>
<td>DSECT</td>
<td>DUO10 parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHDUO10</td>
<td>Macro</td>
<td>DUO10 request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHDUO10</td>
<td>DSECT</td>
<td>DUO10 trace interpretation data</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDUMPX</td>
<td>CSECT</td>
<td>DU domain - SDUMPX IEASDUMP.QUERY exit</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDUPH</td>
<td>CSECT</td>
<td>Dump utility program - dump index summary</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDUPM</td>
<td>CSECT</td>
<td>Dump utility program - module index</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDUPMC</td>
<td>DSECT</td>
<td>Dump utility program - parameter block for module index routine</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHDUPP</td>
<td>CSECT</td>
<td>Dump utility program - I/O routines</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDUPPC</td>
<td>DSECT</td>
<td>Dump utility program - parameter block for print routine</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHDUPR</td>
<td>CSECT</td>
<td>Dump utility program - main component</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDUPS</td>
<td>CSECT</td>
<td>Dump utility program - dump selection</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDUPSC</td>
<td>DSECT</td>
<td>Dump utility program - parameter block for dump selection</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHDUSR</td>
<td>CSECT</td>
<td>DU domain - dump services</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDUSR</td>
<td>DSECT</td>
<td>DUSR parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHDUSR</td>
<td>Macro</td>
<td>DUSR request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHDUSR</td>
<td>CSECT</td>
<td>DU domain - subroutines</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDUSU</td>
<td>CSECT</td>
<td>DU domain - subroutines</td>
<td>OS 03</td>
</tr>
</tbody>
</table>
### Table 123. CICS modules directory (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHDUSUA</td>
<td>DSECT</td>
<td>DUSU parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHDUSUM</td>
<td>Macro</td>
<td>DUSU request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHDUSUT</td>
<td>CSECT</td>
<td>DUSU trace interpretation data</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDUSVC</td>
<td>CSECT</td>
<td>DU domain - SVC processing routine</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDUTFM</td>
<td>CSECT</td>
<td>DU domain - dump table manager</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDUTRI</td>
<td>CSECT</td>
<td>Trace interpreter for DU domain</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDUXD</td>
<td>CSECT</td>
<td>DU domain - transaction dump control</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDUXFA</td>
<td>DSECT</td>
<td>DUXF parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHDUXFM</td>
<td>Macro</td>
<td>DUXF request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHDUXFT</td>
<td>CSECT</td>
<td>DUXF trace interpretation data</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDUXW</td>
<td>CSECT</td>
<td>DU domain - transaction dump buffer control</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDUXWA</td>
<td>DSECT</td>
<td>DUXW parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHDUXWM</td>
<td>Macro</td>
<td>DUXW request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHDWE</td>
<td>CSECT</td>
<td>Deferred work element</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHWDWS</td>
<td>DSECT</td>
<td>Deferred work element</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHDXACH</td>
<td>CSECT</td>
<td>CICS-DBCTL XRF subtask router</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDXAX</td>
<td>CSECT</td>
<td>CICS-DBCTL XRF connection handling</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDXCU</td>
<td>CSECT</td>
<td>CICS-DBCTL XRF catch-up transaction</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDXSTM</td>
<td>CSECT</td>
<td>CICS-DBCTL XRF subtask manager</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHDUXJEP</td>
<td>DSECT</td>
<td>CICS-DBCTL XRF plist to global user exits</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHDYP</td>
<td>Sample</td>
<td>Dynamic routing program</td>
<td>C2 07</td>
</tr>
<tr>
<td>DFHDYP</td>
<td>Sample</td>
<td>Dynamic routing program</td>
<td>D2 -</td>
</tr>
<tr>
<td>DFHDYP</td>
<td>CSECT</td>
<td>User-replaceable dynamic routing program</td>
<td>19 03</td>
</tr>
<tr>
<td>DFHDYPS</td>
<td>DSECT</td>
<td>COMAREA passed to DFHDYP</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHDYPS</td>
<td>DSECT</td>
<td>COMAREA passed to DFHDYP</td>
<td>C2 07</td>
</tr>
<tr>
<td>DFHDYPS</td>
<td>DSECT</td>
<td>COMAREA passed to DFHDYP</td>
<td>D2 -</td>
</tr>
<tr>
<td>DFHD2CC</td>
<td>CSECT</td>
<td>DB2 module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHD2CCT</td>
<td>CSECT</td>
<td>DB2 module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHD2CMP</td>
<td>CSECT</td>
<td>DB2 module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHD2CM0</td>
<td>CSECT</td>
<td>DB2 module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHD2CM1</td>
<td>CSECT</td>
<td>DB2 module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHD2CM2</td>
<td>CSECT</td>
<td>DB2 module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHD2CM3</td>
<td>CSECT</td>
<td>DB2 module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHD2COT</td>
<td>CSECT</td>
<td>DB2 module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHD2OF</td>
<td>CSECT</td>
<td>DB2 module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHD2OFT</td>
<td>CSECT</td>
<td>DB2 module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHD2EDF</td>
<td>CSECT</td>
<td>DB2 module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHD2EXS</td>
<td>CSECT</td>
<td>DB2 module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHD2EX1</td>
<td>CSECT</td>
<td>DB2 module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHD2EX2</td>
<td>CSECT</td>
<td>DB2 module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHD2EX3</td>
<td>CSECT</td>
<td>DB2 module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHD2GDS</td>
<td>CSECT</td>
<td>DB2 module</td>
<td>11 07</td>
</tr>
<tr>
<td>DFHD2IN1</td>
<td>CSECT</td>
<td>DB2 module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHD2IN2</td>
<td>CSECT</td>
<td>DB2 module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHD2II</td>
<td>CSECT</td>
<td>DB2 module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHD2II</td>
<td>CSECT</td>
<td>CICS-DB2 stub (Language interface module)</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHD2M5B</td>
<td>CSECT</td>
<td>DB2 module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHD2RDS</td>
<td>CSECT</td>
<td>DB2 module</td>
<td>11 07</td>
</tr>
<tr>
<td>DFHD2RP</td>
<td>CSECT</td>
<td>DB2 module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHD2SSD</td>
<td>CSECT</td>
<td>DB2 module</td>
<td>0S -</td>
</tr>
<tr>
<td>DFHD2ST</td>
<td>CSECT</td>
<td>DB2 module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHD2STP</td>
<td>CSECT</td>
<td>DB2 module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHD2STR</td>
<td>CSECT</td>
<td>DB2 module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHD2TM</td>
<td>CSECT</td>
<td>DB2 module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHD2TM</td>
<td>CSECT</td>
<td>DB2 module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHD2MTM</td>
<td>CSECT</td>
<td>DB2 module</td>
<td>- 03</td>
</tr>
</tbody>
</table>
## CICS directory

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFH02TRI</td>
<td>CSECT</td>
<td>code generation functions</td>
<td>-03</td>
</tr>
<tr>
<td>DFHEA1</td>
<td>CSECT</td>
<td>EXEC interface link-edit stub for EXEC calls in assembler language programs</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEA10</td>
<td>CSECT</td>
<td>EXEC interface link-edit stub for prolog and epilog calls in assembler language programs</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEAMA2</td>
<td>CSECT</td>
<td>Assembler-language translator - advanced</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEAMEE</td>
<td>CSECT</td>
<td>Assembler-language translator - error editor</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEAMPA</td>
<td>CSECT</td>
<td>Assembler-language translator - primary code generation functions</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEAMS2</td>
<td>CSECT</td>
<td>Assembler-language translator - source scanner</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEAMO2</td>
<td>CSECT</td>
<td>Assembler-language translator - options card</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEAMO8</td>
<td>CSECT</td>
<td>Assembler-language translator - check options</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEAM11</td>
<td>CSECT</td>
<td>Assembler-language translator - atomization</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEBBND</td>
<td>Sample</td>
<td>Part of the CICS EJB sample</td>
<td>- 19</td>
</tr>
<tr>
<td>DFHEBCBJ</td>
<td>Sample</td>
<td>Part of the CICS EJB sample</td>
<td>- 19</td>
</tr>
<tr>
<td>DFHEBCB1</td>
<td>Sample</td>
<td>COBOL source for V2ACTDB program</td>
<td>- 19</td>
</tr>
<tr>
<td>DFHEBCB2</td>
<td>Sample</td>
<td>COBOL source for V2CSTDB program</td>
<td>- 19</td>
</tr>
<tr>
<td>DFHEBCNV</td>
<td>Sample</td>
<td>EJB Sample COMMAREA Conversion Table</td>
<td>- 19</td>
</tr>
<tr>
<td>DFHEBDAT</td>
<td>Sample</td>
<td>Part of the CICS EJB sample</td>
<td>- 19</td>
</tr>
<tr>
<td>DFHEBDEF</td>
<td>Sample</td>
<td>CICS EJB Sample Resource Definitions</td>
<td>- 19</td>
</tr>
<tr>
<td>DFHEBF</td>
<td>CSECT</td>
<td>EXEC interface for BIF DEEDIT command</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEBGRT</td>
<td>Sample</td>
<td>Part of the CICS EJB sample</td>
<td>- 19</td>
</tr>
<tr>
<td>DFHEBCT</td>
<td>CSECT</td>
<td>CBRC LD table</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEBREB</td>
<td>Sample</td>
<td>Part of the CICS EJB sample</td>
<td>- 19</td>
</tr>
<tr>
<td>DFHEBTAB</td>
<td>Sample</td>
<td>Part of the CICS EJB sample</td>
<td>- 19</td>
</tr>
<tr>
<td>DFHEBTAL</td>
<td>Other</td>
<td>Cataloged procedure to translate, assemble and link-edit assembler-language application programs that use EXEC DLI and will run in a batch or CICS shared database region</td>
<td>- 18</td>
</tr>
<tr>
<td>DFHEBTPL</td>
<td>Other</td>
<td>Cataloged procedure to translate, compile and link-edit PL/I application programs that use EXEC DLI and will run in a batch or CICS shared database region</td>
<td>- 18</td>
</tr>
<tr>
<td>DFHEBTVL</td>
<td>Other</td>
<td>Cataloged procedure to translate, compile and link-edit VS COBOL II application programs that use EXEC DLI and will run in a batch or CICS shared database region</td>
<td>- 18</td>
</tr>
<tr>
<td>DFHEBU</td>
<td>CSECT</td>
<td>EXEC FMW construction</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHECADS</td>
<td>DSECT</td>
<td>Event control area for interval control elements</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHECALL</td>
<td>Macro</td>
<td>EXEC interface call macro for assembler-language</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHECB</td>
<td>Macro</td>
<td>CICS posting and testing of operating system ECBs</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHECBAM</td>
<td>CSECT</td>
<td>EXEC interface stub for EXEC calls (COBOL)</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEC1</td>
<td>CSECT</td>
<td>COBOL translator - advanced code generation functions</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHECMAC</td>
<td>CSECT</td>
<td>COBOL translator - error editor</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHECMEE</td>
<td>CSECT</td>
<td>COBOL translator - primary code generation functions</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHECMPC</td>
<td>CSECT</td>
<td>COBOL translator - input scanner</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHECM02</td>
<td>CSECT</td>
<td>COBOL translator - initialization</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHECM07</td>
<td>CSECT</td>
<td>COBOL translator - options card</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHECM08</td>
<td>CSECT</td>
<td>COBOL translator - check options</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHECM10</td>
<td>CSECT</td>
<td>COBOL translator - analyze program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHECM11</td>
<td>CSECT</td>
<td>COBOL translator - atomization</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHECM17</td>
<td>CSECT</td>
<td>COBOL translator - generate output</td>
<td>OS 03</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
<td>Library</td>
</tr>
<tr>
<td>------------</td>
<td>------</td>
<td>--------------------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>DFHEDC</td>
<td>CSECT</td>
<td>EXEC interface for dump control</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEDCP</td>
<td>CSECT</td>
<td>EXEC interface for dump system/transaction</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEDFBR</td>
<td>CSECT</td>
<td>Temporary-storage browse transaction, CEBR</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEDFCB</td>
<td>CSECT</td>
<td>Build one page</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEDFCC</td>
<td>CSECT</td>
<td>Parameter copy program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEDFCE</td>
<td>CSECT</td>
<td>Extract from one page</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEDFCR</td>
<td>CSECT</td>
<td>LD table utilities</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEDFCS</td>
<td>CSECT</td>
<td>CICS special cases</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEDFCX</td>
<td>CSECT</td>
<td>Display unformatted arguments</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEDFD</td>
<td>CSECT</td>
<td>EDF display program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEDFDL</td>
<td>CSECT</td>
<td>DL/I special cases</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEDDFS</td>
<td>DSECT</td>
<td>EDF communication area</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHEDFE</td>
<td>CSECT</td>
<td>EDF attach error handler</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEDFM</td>
<td>CSECT</td>
<td>EDF map set</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEDFP</td>
<td>CSECT</td>
<td>EDF control program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEDFR</td>
<td>CSECT</td>
<td>EDF response table</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEDFS</td>
<td>CSECT</td>
<td>EDF display handling routines</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEDFU</td>
<td>CSECT</td>
<td>Data utilities</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEDFW</td>
<td>CSECT</td>
<td>Display working storage</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEDFX</td>
<td>CSECT</td>
<td>EDF task switch program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEDI</td>
<td>CSECT</td>
<td>EXEC interface for data interchange</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEDMAD</td>
<td>CSECT</td>
<td>C/370 translator - advanced code generation</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEDMEE</td>
<td>CSECT</td>
<td>C/370 translator - error editor</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEDMPO</td>
<td>CSECT</td>
<td>C/370 translator - primary code generation</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEDMSD</td>
<td>CSECT</td>
<td>C/370 translator - input scanner</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEDM02</td>
<td>CSECT</td>
<td>C/370 translator - initialization</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEDM07</td>
<td>CSECT</td>
<td>C/370 translator - options card</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEDM08</td>
<td>CSECT</td>
<td>C/370 translator - check options</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEDM10</td>
<td>CSECT</td>
<td>C/370 translator - analyze program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEDM11</td>
<td>CSECT</td>
<td>C/370 translator - atomization</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEDM14</td>
<td>CSECT</td>
<td>C/370 translator - read input</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEDM17</td>
<td>CSECT</td>
<td>C/370 translator - generate output</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEDP</td>
<td>CSECT</td>
<td>EXEC DLI command stub</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEEI</td>
<td>CSECT</td>
<td>EXEC interface for HANDLE, ADDRESS, ASSIGN</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEEX</td>
<td>CSECT</td>
<td>EXEC FMH extraction</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEGL</td>
<td>CSECT</td>
<td>EXEC interface for unmapped LU6.2 commands</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEIAQ</td>
<td>CSECT</td>
<td>EXEC ACQUIRE TERMINAL</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEIAR</td>
<td>Macro</td>
<td>EIP arguments macro</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHEIBAM</td>
<td>CSECT</td>
<td>- 03</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHEIBLC</td>
<td>DSECT</td>
<td>EXEC interface block</td>
<td>C2 07</td>
</tr>
<tr>
<td>DFHEIBLK</td>
<td>DSECT</td>
<td>EXEC interface block</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHEIBLKM</td>
<td>DSECT</td>
<td>EXEC interface block</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHEICDS</td>
<td>DSECT</td>
<td>EXEC interface COMMAREA</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHEICRE</td>
<td>DSECT</td>
<td>EXEC CICS CREATE command</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEIDDS</td>
<td>Macro</td>
<td>EXEC interface argument 0 descriptor</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHEIDH</td>
<td>CSECT</td>
<td>Document Language Table and EI Layer</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEIDI</td>
<td>CSECT</td>
<td>Address set for COBOL</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHEIDTI</td>
<td>CSECT</td>
<td>EXEC ask-time, format-time program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEIEIA</td>
<td>DSECT</td>
<td>EIEI parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHEIEIEM</td>
<td>Macro</td>
<td>EIEI request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHEIEIT</td>
<td>CSECT</td>
<td>EIEI trace interpretation data</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEIEM</td>
<td>CSECT</td>
<td>DFHEIEM Design Exec EM request handler</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEIEND</td>
<td>Macro</td>
<td>EXEC interface storage end macro</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHEIEN</td>
<td>Macro</td>
<td>EXEC interface prolog macro</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHEIFC</td>
<td>Macro</td>
<td>File control exec interface module</td>
<td>- 03</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
<td>Library</td>
</tr>
<tr>
<td>------------</td>
<td>------------</td>
<td>--------------------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>DFHEIFSP</td>
<td>Macro</td>
<td>Free space</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHEIGBL</td>
<td>Macro</td>
<td>EXEC interface globals definition macro</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHEIGDS</td>
<td>CSECT</td>
<td>Translator table (GDS commands)</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEIGSP</td>
<td>Macro</td>
<td>Get space</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHEIIIC</td>
<td>CSECT</td>
<td>EXEC interface IC module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEIIIF</td>
<td>Macro</td>
<td>EXEC interface IF macro</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHEILIA</td>
<td>Other</td>
<td>Used by DFHEITAL cataloged procedure</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHEILIC</td>
<td>Other</td>
<td>Used by DFHEITCL cataloged procedure</td>
<td>C2 -</td>
</tr>
<tr>
<td>DFHEILID</td>
<td>Other</td>
<td>Used by DFHEITDL cataloged procedure</td>
<td>D2 -</td>
</tr>
<tr>
<td>DFHEILIPI</td>
<td>Other</td>
<td>Used by DFHEITPL cataloged procedure</td>
<td>P2 -</td>
</tr>
<tr>
<td>DFHEIMDS</td>
<td>Macro</td>
<td>Master terminal return codes</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHEIMOP</td>
<td>CSECT</td>
<td>Translator options</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEIMSG</td>
<td>Macro</td>
<td>EXEC interface message macro</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHEIMV</td>
<td>Macro</td>
<td>EXEC interface move macro</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHEIN00</td>
<td>CSECT</td>
<td>Interpreter - CECI/GECS program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEIN01</td>
<td>CSECT</td>
<td>Interpreter - control module</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEIN02</td>
<td>CSECT</td>
<td>Interpreter - initialization</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEIN03</td>
<td>CSECT</td>
<td>CBRC/CECI/CEAS/CEMT - storage manager</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEIN11</td>
<td>CSECT</td>
<td>CBRC/CECI - atomization</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEIN12</td>
<td>CSECT</td>
<td>Interpreter - argument analysis</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEIN13</td>
<td>CSECT</td>
<td>CECI/CEAS/CEMT - diagnosis</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEIN16</td>
<td>CSECT</td>
<td>CECI/CEAS/CEMT - binary conversion</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEIN19</td>
<td>CSECT</td>
<td>Interpreter - command analysis</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEIN20</td>
<td>CSECT</td>
<td>Interpreter - table analysis</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEIN21</td>
<td>CSECT</td>
<td>Interpreter - keyword analysis</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEIN22</td>
<td>CSECT</td>
<td>Interpreter - special case code</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEIN23</td>
<td>CSECT</td>
<td>Interpreter - plist generation</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEIN26</td>
<td>CSECT</td>
<td>CECI/CEMT - message editor</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEIN27</td>
<td>CSECT</td>
<td>Interpreter - spelling correction</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEIN28</td>
<td>CSECT</td>
<td>Interpreter - basic messages</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEIN50</td>
<td>CSECT</td>
<td>Interpreter - special displays</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEIN51</td>
<td>CSECT</td>
<td>Interpreter - display extraction</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEIN52</td>
<td>CSECT</td>
<td>Interpreter - syntax display</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEIN53</td>
<td>CSECT</td>
<td>Interpreter - utilities</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEIN54</td>
<td>CSECT</td>
<td>Interpreter - further utilities</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEIP</td>
<td>CSECT</td>
<td>EXEC (command-level) interface program</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEIPA</td>
<td>CSECT</td>
<td>EXEC interface prolog and epilog code</td>
<td>OS 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Macro</td>
<td>EXEC interface intermodule addressing</td>
<td>OS -</td>
</tr>
<tr>
<td></td>
<td>DSECT</td>
<td>EXEC interface control blocks</td>
<td>11 -</td>
</tr>
<tr>
<td></td>
<td>Source</td>
<td>EXEC interface layer epilog code</td>
<td>OS -</td>
</tr>
<tr>
<td></td>
<td>Symbolic</td>
<td>EXEC interface EQU statements</td>
<td>OS -</td>
</tr>
<tr>
<td></td>
<td>Source</td>
<td>EXEC interface error handling data</td>
<td>OS -</td>
</tr>
<tr>
<td></td>
<td>Macro</td>
<td>EXEC interface epilog code</td>
<td>OS -</td>
</tr>
<tr>
<td></td>
<td>Macro</td>
<td>EXEC interface prolog code</td>
<td>OS -</td>
</tr>
<tr>
<td></td>
<td>Source</td>
<td>EXEC interface layer prolog code</td>
<td>OS -</td>
</tr>
<tr>
<td></td>
<td>CSECT</td>
<td>EXEC interface for perform resettime</td>
<td>- 03</td>
</tr>
<tr>
<td></td>
<td>CSECT</td>
<td>EXEC interface for perform security</td>
<td>- 03</td>
</tr>
<tr>
<td></td>
<td>CSECT</td>
<td>EXEC interface for perform shutdown</td>
<td>- 03</td>
</tr>
<tr>
<td></td>
<td>EXEC</td>
<td>EXEC inquire reqid</td>
<td>- 03</td>
</tr>
<tr>
<td></td>
<td>EXEC</td>
<td>EXEC inquire cfdtpool</td>
<td>- 03</td>
</tr>
<tr>
<td></td>
<td>CSECT</td>
<td>EXEC inquire/set for DCE services domain</td>
<td>- 03</td>
</tr>
<tr>
<td></td>
<td>CSECT</td>
<td>EXEC inquire doctemplate</td>
<td>- 03</td>
</tr>
<tr>
<td></td>
<td>CSECT</td>
<td>EXEC inquire/set for external data sets</td>
<td>- 03</td>
</tr>
<tr>
<td></td>
<td>CSECT</td>
<td>EXEC inquire/set/discard for files</td>
<td>- 03</td>
</tr>
<tr>
<td></td>
<td>CSECT</td>
<td>EXEC inquire/set for dump data sets and dump codes</td>
<td>- 03</td>
</tr>
</tbody>
</table>
Table 123. CICS modules directory (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHEIQD2</td>
<td>CSECT</td>
<td>EXEC inquire/set for IRC</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEIQEJ</td>
<td>CSECT</td>
<td>EXEC inquire/set for monitor and stats</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEIOMS</td>
<td>EXEC</td>
<td>EXEC inquire/set for CEMT-only commands</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEIQMT</td>
<td>EXEC</td>
<td>EXEC inquire/set for journals</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEIQOP</td>
<td>EXEC</td>
<td>EXEC inquire requestmodel</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEIQPF</td>
<td>EXEC</td>
<td>EXEC inquire/discard for profiles</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEIQPN</td>
<td>EXEC</td>
<td>EXEC inquire/discard for partners</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEIQRO</td>
<td>EXEC</td>
<td>EXEC inquire for queued requests (REQIDs)</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEIQRQ</td>
<td>EXEC</td>
<td>EXEC inquire for system attributes</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEIQSA</td>
<td>EXEC</td>
<td>EXEC inquire for connections</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEIQSB</td>
<td>EXEC</td>
<td>EXEC inquire for journals</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEIQSC</td>
<td>EXEC</td>
<td>EXEC inquire for system attributes</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEIQSD</td>
<td>EXEC</td>
<td>EXEC inquire for journalmodel or streamname</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEIQSE</td>
<td>EXEC</td>
<td>EXEC inquire for journalmodel or streamname</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEIQSF</td>
<td>EXEC</td>
<td>EXEC inquire for journals</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEIQSG</td>
<td>EXEC</td>
<td>EXEC inquire for journalmodel or streamname</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEIQSH</td>
<td>EXEC</td>
<td>EXEC inquire for journals</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEIQSI</td>
<td>EXEC</td>
<td>EXEC inquire for journals</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEIQSJ</td>
<td>EXEC</td>
<td>EXEC inquire for journals</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEIQSK</td>
<td>EXEC</td>
<td>EXEC inquire for journals</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEIQSL</td>
<td>EXEC</td>
<td>EXEC inquire for journalmodel or streamname</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEIQSO</td>
<td>EXEC</td>
<td>EXEC inquire for journals</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEIQSP</td>
<td>EXEC</td>
<td>EXEC inquire for journals</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEIQST</td>
<td>EXEC</td>
<td>EXEC inquire for terminals</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEIQSV</td>
<td>EXEC</td>
<td>EXEC inquire for volumes</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEIQSW</td>
<td>EXEC</td>
<td>EXEC inquire for transactions</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEIQSZ</td>
<td>EXEC</td>
<td>EXEC inquire for transactions</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEIQTM</td>
<td>EXEC</td>
<td>EXEC inquire for autinstallmodel</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEIQTR</td>
<td>EXEC</td>
<td>EXEC inquire for trace</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEIQTS</td>
<td>EXEC</td>
<td>EXEC inquire for TS queues</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEIQVT</td>
<td>EXEC</td>
<td>EXEC inquire for exit programs</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEIRET</td>
<td>EXEC</td>
<td>EXEC inquire for VTAM and autoinstall</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEIS</td>
<td>EXEC</td>
<td>EXEC interface epilog macro</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHEISD</td>
<td>EXEC</td>
<td>EXEC interface storage definition</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHEISEI</td>
<td>EXEC</td>
<td>EXEC interface structure entry I/F</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHEISO</td>
<td>EXEC</td>
<td>Sockets Domain API</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEISIP</td>
<td>EXEC</td>
<td>EXEC interface syncpoint processor</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEISIR</td>
<td>EXEC</td>
<td>EXEC interface service routines</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEISRA</td>
<td>DSECT</td>
<td>EISR parameter list</td>
<td>0S -</td>
</tr>
<tr>
<td>DFHEISRM</td>
<td>Macro</td>
<td>EISR request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHEISRT</td>
<td>EXEC</td>
<td>EISR trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEISTG</td>
<td>Macro</td>
<td>EXEC interface storage start macro</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHEITAB</td>
<td>CSECT</td>
<td>Translator table (basic commands)</td>
<td>0S 03</td>
</tr>
<tr>
<td>DFHEITAL</td>
<td>Other</td>
<td>Cataloged procedure to translate, assemble, 18 -</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>and link-edit assembler-language application programs</td>
<td></td>
</tr>
<tr>
<td>DFHEITBS</td>
<td>CSECT</td>
<td>Translator table (special commands)</td>
<td>0S 03</td>
</tr>
<tr>
<td>DFHEITCU</td>
<td>CSECT</td>
<td>RDO offline LD table</td>
<td>0S 03</td>
</tr>
<tr>
<td>DFHEITDL</td>
<td>Other</td>
<td>Cataloged procedure to translate, 18 - compile, and link-edit C/370 application programs</td>
<td></td>
</tr>
<tr>
<td>DFHEITHG</td>
<td>CSECT</td>
<td>EXEC interface hired gun lookup table</td>
<td>0S 03</td>
</tr>
<tr>
<td>DFHEITMT</td>
<td>CSECT</td>
<td>Command language table for CEMT</td>
<td>0S 03</td>
</tr>
<tr>
<td>DFHEITOT</td>
<td>CSECT</td>
<td>Command language table for CEOT</td>
<td>0S 03</td>
</tr>
<tr>
<td>DFHEITPL</td>
<td>Other</td>
<td>Cataloged procedure to translate, 18 - compile, and link-edit PL/I application programs</td>
<td></td>
</tr>
</tbody>
</table>
### CICS directory

**Table 123. CICS modules directory (continued)**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHEIITS</td>
<td>CSECT</td>
<td>Temporary storage exec layer</td>
<td>03</td>
</tr>
<tr>
<td>DFHEIITSP</td>
<td>CSECT</td>
<td>Language definition table</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEIITRD</td>
<td>DSECT</td>
<td>Trace point IDs for DFHETC</td>
<td>OS</td>
</tr>
<tr>
<td>DFHEIINST</td>
<td>CSECT</td>
<td>CEST language definition table</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEITSZ</td>
<td>CSECT (OCO)</td>
<td>EXEC CICS language definition table</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEITTR</td>
<td>CSECT</td>
<td>EXEC interface lookup table</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEITT2</td>
<td>CSECT</td>
<td>EXEC interface level 2 lookup table</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEITUT</td>
<td>Source</td>
<td>Definition of EIP trace entries</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHEITVL</td>
<td>Other</td>
<td>Cataloged procedure to translate, compile, and link-edit VS COBOL II</td>
<td>18 -</td>
</tr>
<tr>
<td>DFHEIUOW</td>
<td>DSECT</td>
<td>EXEC inquire/set uow, or inquire uqenq uowlk and uowdsn fail</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEIUS</td>
<td>DSECT</td>
<td>EXEC interface storage - USER part</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHEIVAR</td>
<td>CSECT</td>
<td>COBOL working storage</td>
<td>C2 -</td>
</tr>
<tr>
<td>DFHEIWB</td>
<td>CSECT</td>
<td>CWI Language Table and EXEC Layer</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEJB</td>
<td>CSECT</td>
<td>SJ Bean Browse</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEJBGT</td>
<td>CSECT</td>
<td>SJ Bean General</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEJCG</td>
<td>CSECT</td>
<td>SJ CorbaServer General</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEJCGT</td>
<td>CSECT</td>
<td>SJ CorbaServer General</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEJCPT</td>
<td>CSECT</td>
<td>SJ JDBC</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEJDB</td>
<td>CSECT</td>
<td>SJ DJar Browse</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEJDBT</td>
<td>CSECT</td>
<td>SJ DJar General</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEJG</td>
<td>CSECT</td>
<td>SJ DJar General</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEJGT</td>
<td>CSECT</td>
<td>SJ DJar General</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEJDI</td>
<td>CSECT</td>
<td>DJE Design EJ Domain EJDI gate functions</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEJDIIT</td>
<td>CSECT</td>
<td>EJ Domain Functions</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEJDM</td>
<td>CSECT</td>
<td>EJ Domain Functions</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEJDND</td>
<td>Sample</td>
<td>Distinguished name URM</td>
<td>19</td>
</tr>
<tr>
<td>DFHEJDNH</td>
<td>Sample</td>
<td>Distinguished name URM</td>
<td>19</td>
</tr>
<tr>
<td>DFHEJDNL</td>
<td>Sample</td>
<td>Distinguished name URM</td>
<td>19</td>
</tr>
<tr>
<td>DFHEJDNO</td>
<td>Sample</td>
<td>Distinguished name URM</td>
<td>19</td>
</tr>
<tr>
<td>DFHEJDNX</td>
<td>Sample</td>
<td>Distinguished name URM</td>
<td>03</td>
</tr>
<tr>
<td>DFHEJDNP</td>
<td>Sample</td>
<td>Distinguished name URM</td>
<td>19</td>
</tr>
<tr>
<td>DFHEJDN2</td>
<td>Sample</td>
<td>CICS-supplied C-language version of DJE/JDI</td>
<td>19</td>
</tr>
<tr>
<td>DFHEJDJ</td>
<td>CSECT</td>
<td>EJ domain EJDU gate functions</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEJDJF</td>
<td>CSECT</td>
<td>Dump interpretation for EJ Domain</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEJDJFT</td>
<td>CSECT</td>
<td>Dump interpretation for EJ Domain</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEJECT</td>
<td>Macro</td>
<td>Page eject/space option</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHEJGE</td>
<td>CSECT</td>
<td>EJ General Operations</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEJGET</td>
<td>CSECT</td>
<td>EJ Domain Functions</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEJJOT</td>
<td>CSECT</td>
<td>EJ Domain Functions</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEJIT</td>
<td>CSECT</td>
<td>EJ Transaction Functions</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEJJO</td>
<td>CSECT</td>
<td>EJ Domain Functions</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEJJOT</td>
<td>CSECT</td>
<td>EJ Domain Functions</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEJMI</td>
<td>CSECT</td>
<td>EJMI CDMRUN and Gate Module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEJMD</td>
<td>CSECT</td>
<td>Message Numbers for the EJ Domain</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHEJMID</td>
<td>CSECT</td>
<td>Message Numbers for the EJ Domain</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEJMB</td>
<td>CSECT</td>
<td>Object Store Browse</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEJMBT</td>
<td>CSECT</td>
<td>Object Store Program</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEJOS</td>
<td>CSECT</td>
<td>Object Store Program</td>
<td>- 03</td>
</tr>
</tbody>
</table>
## Table 123. CICS modules directory (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHEJOST</td>
<td>CSECT</td>
<td>-03</td>
<td>-03</td>
</tr>
<tr>
<td>DFHEJRDS</td>
<td>CSECT</td>
<td>11 07</td>
<td></td>
</tr>
<tr>
<td>DFHEJST</td>
<td>CSECT</td>
<td>EJ Domain - Statistics (STST) gate</td>
<td>-03</td>
</tr>
<tr>
<td>DFHEJTBB</td>
<td>CSECT</td>
<td>-03</td>
<td></td>
</tr>
<tr>
<td>DFHEJTBG</td>
<td>CSECT</td>
<td>-03</td>
<td></td>
</tr>
<tr>
<td>DFHEJTB1</td>
<td>CSECT</td>
<td>-03</td>
<td></td>
</tr>
<tr>
<td>DFHEJTCB</td>
<td>CSECT</td>
<td>-03</td>
<td></td>
</tr>
<tr>
<td>DFHEJTCG</td>
<td>CSECT</td>
<td>-03</td>
<td></td>
</tr>
<tr>
<td>DFHEJTC1</td>
<td>CSECT</td>
<td>-03</td>
<td></td>
</tr>
<tr>
<td>DFHEJTD</td>
<td>CSECT</td>
<td>-03</td>
<td></td>
</tr>
<tr>
<td>DFHEJTDG</td>
<td>CSECT</td>
<td>-03</td>
<td></td>
</tr>
<tr>
<td>DFHEJTM</td>
<td>CSECT</td>
<td>-03</td>
<td></td>
</tr>
<tr>
<td>DFHEJTD</td>
<td>CSECT</td>
<td>-03</td>
<td></td>
</tr>
<tr>
<td>DFHEJTGE</td>
<td>CSECT</td>
<td>-03</td>
<td></td>
</tr>
<tr>
<td>DFHEJTED</td>
<td>Macro</td>
<td>Trace Points for the EJ Domain</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHEJTO</td>
<td>CSECT</td>
<td>-03</td>
<td></td>
</tr>
<tr>
<td>DFHEJTI</td>
<td>CSECT</td>
<td>-03</td>
<td></td>
</tr>
<tr>
<td>DFHEJTO</td>
<td>CSECT</td>
<td>-03</td>
<td></td>
</tr>
<tr>
<td>DFHEJTR</td>
<td>CSECT</td>
<td>-03</td>
<td></td>
</tr>
<tr>
<td>DFHEJTR1</td>
<td>CSECT</td>
<td>EJ Trace Domain interpretation</td>
<td>-03</td>
</tr>
<tr>
<td>DFHEJUPA</td>
<td>Macro</td>
<td>EJ XRSINDI Overlay</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHEJXDF</td>
<td>CSECT</td>
<td>Transaction Dump - JRAS dump info</td>
<td>-03</td>
</tr>
<tr>
<td>DFHEK</td>
<td>CSECT</td>
<td>EXEC interface for task control</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEL</td>
<td>CSECT</td>
<td>EXEC interface link-edit stub for C/370 application programs</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEMBA</td>
<td>CSECT</td>
<td>EM Domain - EMBA gate functions</td>
<td>-03</td>
</tr>
<tr>
<td>DFHEMBR</td>
<td>CSECT</td>
<td>EM Domain - EMBR gate functions</td>
<td>-03</td>
</tr>
<tr>
<td>DFHEMBRT</td>
<td>CSECT</td>
<td>-03</td>
<td></td>
</tr>
<tr>
<td>DFHEMDM</td>
<td>CSECT</td>
<td>EM Domain - DMDM gate functions</td>
<td>-03</td>
</tr>
<tr>
<td>DFHEMDUF</td>
<td>CSECT</td>
<td>DFHEMDUF Design</td>
<td>-03</td>
</tr>
<tr>
<td>DFHEMEM</td>
<td>CSECT</td>
<td>EM Domain - EMEM gate functions</td>
<td>-03</td>
</tr>
<tr>
<td>DFHEMEXT</td>
<td>CSECT</td>
<td>-03</td>
<td></td>
</tr>
<tr>
<td>DFHEMPID</td>
<td>CSECT</td>
<td>EXEC interface for ME domain</td>
<td>-03</td>
</tr>
<tr>
<td>DFHEMS</td>
<td>CSECT</td>
<td>EXEC interface for BMS</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEMT00</td>
<td>CSECT</td>
<td>Master terminal - CEMT/CEOT/CEST program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEMT01</td>
<td>CSECT</td>
<td>Master terminal - control module</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEMT02</td>
<td>CSECT</td>
<td>Master terminal - initialization</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEMT11</td>
<td>CSECT</td>
<td>Master terminal - atomization</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEMT12</td>
<td>CSECT</td>
<td>Master terminal - argument analysis</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEMT19</td>
<td>CSECT</td>
<td>Master terminal - command analysis</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEMT20</td>
<td>CSECT</td>
<td>Master terminal - table analysis</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEMT21</td>
<td>CSECT</td>
<td>Master terminal - keyword analysis</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEMT22</td>
<td>CSECT</td>
<td>Master terminal - special case code</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEMT23</td>
<td>CSECT</td>
<td>Master terminal - plist generation</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEMT27</td>
<td>CSECT</td>
<td>Master terminal - spelling correction</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEMT50</td>
<td>CSECT</td>
<td>Master terminal - special displays</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEMT51</td>
<td>CSECT</td>
<td>Master terminal - display extraction</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEMT52</td>
<td>CSECT</td>
<td>Master terminal - syntax display</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEMT53</td>
<td>CSECT</td>
<td>Master terminal - utilities</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEMT54</td>
<td>CSECT</td>
<td>Master terminal - further utilities</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEMT55</td>
<td>CSECT</td>
<td>Master terminal - fulists</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEMT56</td>
<td>CSECT</td>
<td>Master terminal - execution interface</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEMT1</td>
<td>CSECT</td>
<td>DFHEMT1 Design</td>
<td>-03</td>
</tr>
<tr>
<td>DFHEND</td>
<td>Macro</td>
<td>Generate END statement</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHENV</td>
<td>Macro</td>
<td>CICS environment service request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHEOP</td>
<td>CSECT (OCO)</td>
<td>EXEC interface for write operator</td>
<td>-03</td>
</tr>
<tr>
<td>DFHEPC</td>
<td>CSECT</td>
<td>EXEC interface for program control</td>
<td>-03</td>
</tr>
</tbody>
</table>
### Table 123. CICS modules directory (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHEPILO</td>
<td>Macro</td>
<td>Free automatic storage application epilog</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHEPMAPI</td>
<td>CSECT</td>
<td>PL/I translator - advanced code generation functions</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEPMEE</td>
<td>CSECT</td>
<td>PL/I translator - error editor</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEPMPP</td>
<td>CSECT</td>
<td>PL/I translator - primary code generation functions</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEPMSP</td>
<td>CSECT</td>
<td>PL/I translator - input scanner</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEPM02</td>
<td>CSECT</td>
<td>PL/I translator - initialization</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEPM07</td>
<td>CSECT</td>
<td>PL/I translator - options card</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEPM08</td>
<td>CSECT</td>
<td>PL/I translator - check options</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEPM10</td>
<td>CSECT</td>
<td>PL/I translator - analyze program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEPM11</td>
<td>CSECT</td>
<td>PL/I translator - atomization</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEPM14</td>
<td>CSECT</td>
<td>PL/I translator - read input</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEPM17</td>
<td>CSECT</td>
<td>PL/I translator - generate output</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEPS</td>
<td>CSECT</td>
<td>System spooling interface stub</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHERDUF</td>
<td>CSECT (OCO)</td>
<td>SDUMP error message index processor</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHERM</td>
<td>CSECT</td>
<td>Resource manager interface (RMI) module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHERMRSP</td>
<td>CSECT</td>
<td>ERM resync processor</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHERMSP</td>
<td>CSECT</td>
<td>ERM syncpoint processor</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHESC</td>
<td>CSECT</td>
<td>EXEC interface for storage control</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHESN</td>
<td>CSECT (OCO)</td>
<td>EXEC interface for signon and sign-off</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHESP00</td>
<td>CSECT</td>
<td>RDO - CEDA/CEDB/CEDC program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHESP01</td>
<td>CSECT</td>
<td>RDO - CEDA control module</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHESP02</td>
<td>CSECT</td>
<td>RDO - CEDA initialization</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHESP11</td>
<td>CSECT</td>
<td>RDO - CEDA atomization</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHESP12</td>
<td>CSECT</td>
<td>RDO - CEDA argument analysis</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHESP19</td>
<td>CSECT</td>
<td>RDO - CEDA command analysis</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHESP20</td>
<td>CSECT</td>
<td>RDO - CEDA table analysis</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHESP21</td>
<td>CSECT</td>
<td>RDO - CEDA keyword analysis</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHESP22</td>
<td>CSECT</td>
<td>RDO - CEDA special case code</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHESP23</td>
<td>CSECT</td>
<td>RDO - CEDA plist generation</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHESP26</td>
<td>CSECT</td>
<td>RDO - CEDA message editor</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHESP27</td>
<td>CSECT</td>
<td>RDO - CEDA spelling correction</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHESP50</td>
<td>CSECT</td>
<td>RDO - CEDA special displays</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHESP51</td>
<td>CSECT</td>
<td>RDO - CEDA display extraction</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHESP52</td>
<td>CSECT</td>
<td>RDO - CEDA syntax display</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHESP53</td>
<td>CSECT</td>
<td>RDO - CEDA utilities</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHESP54</td>
<td>CSECT</td>
<td>RDO - CEDA further utilities</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHESP55</td>
<td>CSECT</td>
<td>RDO - CEDA fulists</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHESZ</td>
<td>CSECT (OCO)</td>
<td>EXEC CICS API commands for FEPI</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHETC</td>
<td>CSECT</td>
<td>EXEC interface for terminal control</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHETCB</td>
<td>Macro</td>
<td>EXEC terminal control block macro</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHETD</td>
<td>CSECT</td>
<td>EXEC interface for transient data</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHETL</td>
<td>CSECT</td>
<td>LUB.Z EXEC interface stub</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHETR</td>
<td>CSECT</td>
<td>EXEC interface for trace control</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHETRX</td>
<td>CSECT (OCO)</td>
<td>EXEC interface for enter tracenum, monitor</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEXAI</td>
<td>CSECT</td>
<td>Link-edit stub for assembler-language programs using CSD offline extract</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEXCI</td>
<td>CSECT</td>
<td>Link-edit stub for COBOL programs using CSD offline extract function</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEXDF</td>
<td>CSECT (OCO)</td>
<td>EXCI dump formatting routine</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHEX1</td>
<td>CSECT</td>
<td>Terminal exceptional input program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEXLE</td>
<td>CSECT</td>
<td></td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEXLI</td>
<td>CSECT</td>
<td>EXCI stub</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHEXMB</td>
<td>CSECT</td>
<td>Translators - default argument text build</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEXMAN</td>
<td>CSECT</td>
<td>Translators - statement syntax analysis</td>
<td>OS 03</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
<td>Library</td>
</tr>
<tr>
<td>--------</td>
<td>-------</td>
<td>-----------------------------------------------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>DFHEXMG1</td>
<td>CSECT</td>
<td>Translators - EXEC DLI code generator</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEXMG2</td>
<td>CSECT</td>
<td>Translators - EXEC CICS code generator</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEXMG3</td>
<td>CSECT</td>
<td>Translators - EXEC CICS GDS code generator</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEXMG4</td>
<td>CSECT</td>
<td>Translators - EXEC EXCI code generator</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEXMG5</td>
<td>CSECT</td>
<td>Translators - CICSPlex SM EXEC CICS command code generator</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEXMGW</td>
<td>CSECT</td>
<td>Translators - keyword analysis</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEXMP</td>
<td>CSECT</td>
<td>Translators - fatal error handler</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEXMS1</td>
<td>CSECT</td>
<td>Translators - DL/I WHERE operand code generator</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEXMS2</td>
<td>CSECT</td>
<td>Translators - EXEC CICS special case code generator</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEXMS3</td>
<td>CSECT</td>
<td>Translators - EXEC CICS GDS special case code generator</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEXMS4</td>
<td>CSECT</td>
<td>Translators - EXEC EXCI special case code generator</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEXMS5</td>
<td>CSECT</td>
<td>Translators - EXEC EXCI special case code generator for CICSPlex SM</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEXMD</td>
<td>CSECT</td>
<td>Translators - temporaries declaration</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEXMT</td>
<td>CSECT</td>
<td>Translators - EXEC trigger detection</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEXMK</td>
<td>CSECT</td>
<td>Translators - syntax checker</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEXMS</td>
<td>CSECT</td>
<td>Translators - syntax check control module</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEXMO</td>
<td>CSECT</td>
<td>Translators - control module</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEXMO1</td>
<td>CSECT</td>
<td>Translators - control module</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEXMO5</td>
<td>CSECT</td>
<td>Translators - PARM analysis</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEXMO6</td>
<td>CSECT</td>
<td>Translators - process single option</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEXMO9</td>
<td>CSECT</td>
<td>Translators - print options</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEXM12</td>
<td>CSECT</td>
<td>Translators - match brackets</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEXM13</td>
<td>CSECT</td>
<td>Translators - diagnosis</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEXM15</td>
<td>CSECT</td>
<td>Translators - I/O module</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEXM16</td>
<td>CSECT</td>
<td>Translators - conversions</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEXM18</td>
<td>CSECT</td>
<td>Translators - insert in I/O buffer</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEXM25</td>
<td>CSECT</td>
<td>Translators - print xref</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEXM27</td>
<td>CSECT</td>
<td>Translators - spelling correction</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEXPI</td>
<td>CSECT</td>
<td>Link-edit stub for PL/I programs</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEXTAL</td>
<td>Other</td>
<td>Cataloged procedure to translate, assemble, and link-edit Assembler-language application programs (EXCI)</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEXTDL</td>
<td>Other</td>
<td>Cataloged procedure to translate, compile, and link-edit C/370 application programs (EXCI)</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEXTM</td>
<td>Macro</td>
<td>Dummy macro for DOS compatibility</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHEXTPL</td>
<td>Other</td>
<td>Cataloged procedure to translate, compile, and link-edit PL/I application programs (EXCI)</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEXTRL</td>
<td>Macro</td>
<td>EXCI trace interpretation routine</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHEXTVL</td>
<td>Other</td>
<td>Cataloged procedure to translate, compile, and link-edit VS COBOL II application programs (EXCI)</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHFAUED</td>
<td>DSECT</td>
<td></td>
<td>- 11</td>
</tr>
<tr>
<td>DFHFBDPS</td>
<td>DSECT</td>
<td>File buffer pool control block</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHFCAT</td>
<td>CSECT</td>
<td>File control catalog manager</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHFCATL</td>
<td>DSECT</td>
<td>FCAT parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHFCATM</td>
<td>Macro</td>
<td>FCAT request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHFCATT</td>
<td>CSECT</td>
<td>FCAT translate tables</td>
<td>OS 03</td>
</tr>
</tbody>
</table>
### CICS directory

Table 123. CICS modules directory (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHFCBD</td>
<td>CSECT</td>
<td>File control BDAM request processor</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHFCCA</td>
<td>CSECT (OCO)</td>
<td>File control RLS control ACB manager</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHFCCAT</td>
<td>CSECT (OCO)</td>
<td>FCCA translate tables</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHFCCRT</td>
<td>CSECT</td>
<td>-</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHFCCTT</td>
<td>CSECT</td>
<td>-</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHFCCUT</td>
<td>CSECT</td>
<td>-</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHFCDL</td>
<td>CSECT</td>
<td>File control CFDT Load</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHFCDN</td>
<td>CSECT (OCO)</td>
<td>File control DSN block manager</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHFCDNA</td>
<td>DSECT</td>
<td>FCDS parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHFCDNM</td>
<td>Macro</td>
<td>FCDS request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHFCONT</td>
<td>CSECT (OCO)</td>
<td>FCDS translate tables</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHFCDO</td>
<td>CSECT</td>
<td>File control CFDT Open/Close</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHFCDR</td>
<td>CSECT</td>
<td>FC CF data table request handler</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHFCDSI</td>
<td>CSECT</td>
<td>-</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHFCDSJ</td>
<td>CSECT (OCO)</td>
<td>Shared data table request program</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHFCDTX</td>
<td>CSECT (OCO)</td>
<td>File control shared data table function</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHFCDU</td>
<td>CSECT</td>
<td>File control CFDT Recovery Control</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHFCDFU</td>
<td>CSECT (OCO)</td>
<td>File control SDUMP formatter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHFCDUT</td>
<td>CSECT</td>
<td>File control SDUMP formatter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHFCDW</td>
<td>CSECT</td>
<td>File control CFDT Recovery Control</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHFCDY</td>
<td>CSECT</td>
<td>File control CFDT Recovery Resynchronization</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHFCDTY</td>
<td>CSECT</td>
<td>-</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHFCEDS</td>
<td>DSECT</td>
<td>File control EXEC argument list</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHFCES</td>
<td>CSECT (OCO)</td>
<td>File control ENF servicer</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHFCFL</td>
<td>CSECT (OCO)</td>
<td>File control FRAB/FLAB processor</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHFCFLA</td>
<td>DSECT</td>
<td>FCFL parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHFCFLM</td>
<td>Macro</td>
<td>FCFL request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHFCFLT</td>
<td>CSECT</td>
<td>FCFL translate tables</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHFCFR</td>
<td>CSECT</td>
<td>File control file request handler</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHFCFRA</td>
<td>DSECT</td>
<td>FCFR parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHFCFRM</td>
<td>Macro</td>
<td>FCFR request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHFCFRT</td>
<td>CSECT</td>
<td>FCFR trace interpretation data</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHFCFS</td>
<td>CSECT</td>
<td>File control file state program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHFCFSM</td>
<td>Macro</td>
<td>FCFS request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHFCFST</td>
<td>CSECT</td>
<td>FCFS translate tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHFCFSA</td>
<td>DSECT</td>
<td>FCFS parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHFCFSM</td>
<td>Macro</td>
<td>FCFS request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHFCFST</td>
<td>CSECT</td>
<td>FCFS translate tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHFCFINA</td>
<td>DSECT</td>
<td>FCFIN parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHFCINM</td>
<td>Macro</td>
<td>FCFIN request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHFCINT</td>
<td>CSECT</td>
<td>FCFIN translate tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHFCIN1</td>
<td>CSECT</td>
<td>File control initialization program 1</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHFCIN2</td>
<td>CSECT</td>
<td>File control initialization program 2</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHFCIR</td>
<td>CSECT (OCO)</td>
<td>File control initialize recovery module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHFCIL</td>
<td>CSECT</td>
<td>File control VSAM LSR pool processor</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHFCILF</td>
<td>CSECT (OCO)</td>
<td>File control logger failures</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHFCILGF</td>
<td>CSECT</td>
<td>File control part of log record</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHFCILJ</td>
<td>CSECT (OCO)</td>
<td>File control logging and journaling</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHFCILJA</td>
<td>DSECT</td>
<td>FCLJ parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHFCILM</td>
<td>Macro</td>
<td>FCLJ request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHFCILM</td>
<td>CSECT</td>
<td>FCLJ translate tables</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHFCLTD</td>
<td>DSECT</td>
<td>File control logger user token</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHFCMC</td>
<td>CSECT</td>
<td>File control VSAM KSDS base open/close</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHFCMT</td>
<td>CSECT (OCO)</td>
<td>File control table manager</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHFCMTA</td>
<td>DSECT</td>
<td>FCMT parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHFCMTM</td>
<td>Macro</td>
<td>FCMT request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHFCMTT</td>
<td>CSECT (OCO)</td>
<td>FCMT translate tables</td>
<td>- 03</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
<td>Library</td>
</tr>
<tr>
<td>--------------</td>
<td>--------</td>
<td>---------------------------------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>DFHFCN</td>
<td>CSECT</td>
<td>File control open/close program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHFCNC</td>
<td>Source</td>
<td>File control - close request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHFCNO</td>
<td>Source</td>
<td>File control - open request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHFCNQ</td>
<td>CSECT (OCO)</td>
<td>File control non-RLS lock handler completion</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHFCOR</td>
<td>CSECT (OCO)</td>
<td>File control RLS offsite recovery</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHFCQI</td>
<td>CSECT</td>
<td>File control - VSAM RLS quiesce initiation module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHFCQIT</td>
<td>DSECT</td>
<td>FCQI translate tables</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHFCQR</td>
<td>CSECT (OCO)</td>
<td>File control - VSAM RLS quiesce receive module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHFCQRT</td>
<td>DSECT</td>
<td>FCQR translate tables</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHFCQS</td>
<td>CSECT (OCO)</td>
<td>File control - VSAM RLS quiesce send module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHFCQST</td>
<td>DSECT</td>
<td>FCQS translate tables</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHFCQT</td>
<td>CSECT (OCO)</td>
<td>File control - VSAM RLS quiesce process module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHFCQU</td>
<td>CSECT (OCO)</td>
<td>File control - VSAM RLS quiesce send module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHFCQUT</td>
<td>DSECT</td>
<td>FCQU translate tables</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHFCQX</td>
<td>CSECT (OCO)</td>
<td>File control - VSAM RLS quiesce exit module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHFCRC</td>
<td>CSECT (OCO)</td>
<td>File control recovery control</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHFCRD</td>
<td>CSECT (OCO)</td>
<td>File control VSAM RLS post server-failure recovery</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHFCRF</td>
<td>CSECT</td>
<td>File control Remote Interface</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHFCRFM</td>
<td>Macro</td>
<td>DFHFCRF interface macro</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHFCRFIT</td>
<td>CSECT</td>
<td>File control VSAM SHRCTL block manager</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHFCRLA</td>
<td>DSECT</td>
<td>FCRL parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHFCRMLM</td>
<td>Macro</td>
<td>FCRL request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHFCRCLT</td>
<td>CSECT (OCO)</td>
<td>FCRL translate tables</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHFCRCRO</td>
<td>CSECT (OCO)</td>
<td>File control VSAM RLS open/close processor</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHFCRCP</td>
<td>CSECT</td>
<td>File control restart program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHFCRPA</td>
<td>DSECT</td>
<td>FCRP parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHFCRPM</td>
<td>Macro</td>
<td>FCRP request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHFCRPT</td>
<td>CSECT</td>
<td>FCRP translate tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHFCCR</td>
<td>CSECT (OCO)</td>
<td>File control RLS restart program</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHFCRRT</td>
<td>CSECT</td>
<td>FCRR translate tables</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHFCRSA</td>
<td>CSECT (OCO)</td>
<td>File control RLS record management program</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHFCRCV</td>
<td>CSECT (OCO)</td>
<td>File control RLS VSAM interface program</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHFCSD</td>
<td>CSECT</td>
<td>File control shutdown program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHFCSDA</td>
<td>DSECT</td>
<td>FCSD parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHFCSDM</td>
<td>Macro</td>
<td>FCSD request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHFCSDS</td>
<td>DSECT</td>
<td>File control static storage</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHFCSDT</td>
<td>CSECT</td>
<td>FCSD translate tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHFCST</td>
<td>CSECT</td>
<td>File control statistics program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHFCSTA</td>
<td>DSECT</td>
<td>FCST parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHFCSTM</td>
<td>Macro</td>
<td>FCST request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHFCSTT</td>
<td>CSECT</td>
<td>FCST translate tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHFCCT</td>
<td>Macro</td>
<td>File control table</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHFCRTDS</td>
<td>DSECT</td>
<td>File control table entry</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHFACTRN</td>
<td>Symbolic</td>
<td>File control trace, message, and catalog constants for assembler modules</td>
<td>OS -</td>
</tr>
</tbody>
</table>
### CICS directory

#### Table 123. CICS modules directory (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHFCU</td>
<td>CSECT</td>
<td>File open utility program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHFCVR</td>
<td>CSECT</td>
<td>File control VSAM interface program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHFCVS</td>
<td>CSECT</td>
<td>File access VSAM request processor</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHFCWS</td>
<td>Macro</td>
<td>File control work areas</td>
<td>OS</td>
</tr>
<tr>
<td>DFHFCXDF</td>
<td>CSECT</td>
<td>DU domain - transaction dump formatter for file-related areas</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHFEP</td>
<td>CSECT</td>
<td>Field engineering program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHFIOA</td>
<td>DSECT</td>
<td>File input/output area</td>
<td>OS</td>
</tr>
<tr>
<td>DFHFLABD</td>
<td>DSECT</td>
<td>File lasting access block</td>
<td>OS</td>
</tr>
<tr>
<td>DFHFMH</td>
<td>Macro</td>
<td>Function management header</td>
<td>OS</td>
</tr>
<tr>
<td>DFHFMIDS</td>
<td>DSECT</td>
<td>Function management header</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHFORMS</td>
<td>CSECT</td>
<td></td>
<td>03</td>
</tr>
<tr>
<td>DFHFRABD</td>
<td>DSECT</td>
<td>File request anchor block</td>
<td>OS</td>
</tr>
<tr>
<td>DFHFRDUF</td>
<td>CSECT</td>
<td>File control recoverable work elements SDUMP formatter for anchor area</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHFRTED</td>
<td>DSECT</td>
<td>File request thread element</td>
<td>OS</td>
</tr>
<tr>
<td>DFHTDUF</td>
<td>CSECT</td>
<td>(OCO) Print feature 'FT' keyword processor</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTTRI</td>
<td>CSECT</td>
<td>(OCO) Offline TR entries trace interpretation</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHGCAA</td>
<td>CSECT</td>
<td>Language Environment/370 - get common anchor area</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHGDEFS</td>
<td>Symbolic</td>
<td>CICS global symbol definitions</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHGMN</td>
<td>CSECT</td>
<td>VTAM LU startup message</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHHASH</td>
<td>Macro</td>
<td>Locate TCTTE entries</td>
<td>OS</td>
</tr>
<tr>
<td>DFHHLPDS</td>
<td>DSECT</td>
<td>DL/I interface block</td>
<td>D3</td>
</tr>
<tr>
<td>DFHHMDCD</td>
<td>DSECT</td>
<td>CICS-IMS HLPI control blocks</td>
<td>OS 08</td>
</tr>
<tr>
<td>DFHHPSVC</td>
<td>CSECT</td>
<td>Handle manager table block</td>
<td>OS</td>
</tr>
<tr>
<td>DFHIC</td>
<td>Macro</td>
<td>Time service request</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHICEDS</td>
<td>DSECT</td>
<td>Interval control SDUMP formatter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHICIP</td>
<td>CSECT</td>
<td>Interval control program</td>
<td>03</td>
</tr>
<tr>
<td>DFHICRC</td>
<td>CSECT</td>
<td>Interval control recovery module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHICUED</td>
<td>CSECT</td>
<td>EXEC argument list for Interval Control</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHICXM</td>
<td>CSECT</td>
<td>AP domain - bind, inquire, and release facility IC functions</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHICXMA</td>
<td>DSECT</td>
<td>ICMX parameter list</td>
<td>OS</td>
</tr>
<tr>
<td>DFHICXMM</td>
<td>Macro</td>
<td>ICMX request</td>
<td>OS</td>
</tr>
<tr>
<td>DFHICXMT</td>
<td>CSECT</td>
<td>ICMX translate tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHIEDM</td>
<td>CSECT</td>
<td>IE Domain Initialization/Termination</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHIEDUF</td>
<td>CSECT</td>
<td>IE Domain System Dump Formatting</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHIIEIE</td>
<td>CSECT</td>
<td>IP ECI Listener</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHIIEIEA</td>
<td>CSECT</td>
<td>IEIE interface parameter area</td>
<td>OS</td>
</tr>
<tr>
<td>DFHIIEIEM</td>
<td>Macro</td>
<td>DFHIIEIE interface macro</td>
<td>OS</td>
</tr>
<tr>
<td>DFHIIEIET</td>
<td>CSECT</td>
<td></td>
<td>03</td>
</tr>
<tr>
<td>DFHIIEP</td>
<td>CSECT</td>
<td></td>
<td>03</td>
</tr>
<tr>
<td>DFHIERI</td>
<td>CSECT</td>
<td>IP ECI Domain Trace Interpretation</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHIEXM</td>
<td>CSECT</td>
<td></td>
<td>03</td>
</tr>
<tr>
<td>DFHIHFSB</td>
<td>CSECT</td>
<td></td>
<td>02</td>
</tr>
<tr>
<td>DFHIHFSI</td>
<td>CSECT</td>
<td></td>
<td>02</td>
</tr>
<tr>
<td>DFHIISC</td>
<td>CSECT</td>
<td>IIOP Command Processor</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHIIDM</td>
<td>CSECT</td>
<td>II Domain Initialization/Termination</td>
<td>- 03</td>
</tr>
</tbody>
</table>
### Table 123. CICS modules directory (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHIIDUF</td>
<td>CSECT</td>
<td>II Domain System Dump Formatting</td>
<td>03</td>
</tr>
<tr>
<td>DFHIILST</td>
<td>CSECT</td>
<td></td>
<td>03</td>
</tr>
<tr>
<td>DFHIIMM</td>
<td>CSECT</td>
<td>DFHIIMM Design II domain - IIMM gate functs.</td>
<td>03</td>
</tr>
<tr>
<td>DFHIIMT</td>
<td>CSECT</td>
<td></td>
<td>03</td>
</tr>
<tr>
<td>DFHIIP</td>
<td>CSECT</td>
<td>BMS non-3270 input mapping</td>
<td>05 -</td>
</tr>
<tr>
<td>DFHIIPAG</td>
<td>CSECT</td>
<td>BMS non-3270 input mapping (standard)</td>
<td>05 03</td>
</tr>
<tr>
<td>DFHIIP1G</td>
<td>CSECT</td>
<td>BMS non-3270 input mapping (full)</td>
<td>05 03</td>
</tr>
<tr>
<td>DFHIIRDS</td>
<td>DSECT</td>
<td></td>
<td>11 07</td>
</tr>
<tr>
<td>DFHIIRH</td>
<td>DSECT</td>
<td>IIOP Request Handler</td>
<td>03</td>
</tr>
<tr>
<td>DFHIIRHT</td>
<td>DSECT</td>
<td></td>
<td>03</td>
</tr>
<tr>
<td>DFHIIRP</td>
<td>CSECT</td>
<td>IIOP Request Processor</td>
<td>03</td>
</tr>
<tr>
<td>DFHIIRPT</td>
<td>DSECT</td>
<td></td>
<td>03</td>
</tr>
<tr>
<td>DFHIIRQ</td>
<td>CSECT</td>
<td>DFHIIRQ Design</td>
<td>03</td>
</tr>
<tr>
<td>DFHIIRQT</td>
<td>CSECT</td>
<td></td>
<td>03</td>
</tr>
<tr>
<td>DFHIIRR</td>
<td>CSECT</td>
<td>IIOP Request Receiver</td>
<td>03</td>
</tr>
<tr>
<td>DFHIIRRS</td>
<td>CSECT</td>
<td></td>
<td>03</td>
</tr>
<tr>
<td>DFHIIRRT</td>
<td>CSECT</td>
<td></td>
<td>03</td>
</tr>
<tr>
<td>DFHIIST</td>
<td>CSECT</td>
<td>II Domain - Statistics (STST) gate</td>
<td>03</td>
</tr>
<tr>
<td>DFHIITRI</td>
<td>CSECT</td>
<td>IIOP Domain Trace Interpretation</td>
<td>03</td>
</tr>
<tr>
<td>DFHIIRUH</td>
<td>CSECT</td>
<td></td>
<td>08</td>
</tr>
<tr>
<td>DFHIIXM</td>
<td>CSECT</td>
<td>IIOP Attach Client</td>
<td>03</td>
</tr>
<tr>
<td>DFHIQVME</td>
<td>Other</td>
<td>Customize a member of the SDFHENV library</td>
<td>02 -</td>
</tr>
<tr>
<td>DFHIQVMJ</td>
<td>Other</td>
<td></td>
<td>02 -</td>
</tr>
<tr>
<td>DFHI1LG1</td>
<td>Other</td>
<td>Define logstream CF structures to MVS logger</td>
<td>02 -</td>
</tr>
<tr>
<td>DFHI1LG2</td>
<td>Other</td>
<td>Define logstream models for system log</td>
<td>02 -</td>
</tr>
<tr>
<td></td>
<td></td>
<td>streams</td>
<td></td>
</tr>
<tr>
<td>DFHI1LG3</td>
<td>Other</td>
<td>Define logstream models for individual CICS</td>
<td>02 -</td>
</tr>
<tr>
<td></td>
<td></td>
<td>region</td>
<td></td>
</tr>
<tr>
<td>DFHI1LG4</td>
<td>Other</td>
<td>Define specific logstream for log of logs</td>
<td>02 -</td>
</tr>
<tr>
<td>DFHI1LG5</td>
<td>Other</td>
<td></td>
<td>02 -</td>
</tr>
<tr>
<td>DFHI1LG6</td>
<td>Other</td>
<td></td>
<td>02 -</td>
</tr>
<tr>
<td>DFHI1LG7</td>
<td>Other</td>
<td></td>
<td>02 -</td>
</tr>
<tr>
<td>DFHI1MSDS</td>
<td>DSECT</td>
<td>ISC message inserts</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHI1NDAP</td>
<td>CSECT</td>
<td>Indoubt tool</td>
<td>03</td>
</tr>
<tr>
<td>DFHI1NDS</td>
<td>CSECT</td>
<td>Indoubt tool syncpoint processor</td>
<td>03</td>
</tr>
<tr>
<td>DFHI1NT</td>
<td>CSECT</td>
<td>Indoubt tool</td>
<td>03</td>
</tr>
<tr>
<td>DFHI1NSTA</td>
<td>Other</td>
<td>TSO CLIST to generate installation jobs</td>
<td>02 -</td>
</tr>
<tr>
<td>DFHI1NSTJ</td>
<td>Other</td>
<td>JCL to create an additional target zone, CSI, and set of target libraries</td>
<td>02 -</td>
</tr>
<tr>
<td>DFHI1NST1</td>
<td>Other</td>
<td>JCL to RECEIVE, APPLY, and ACCEPT the Japanese language feature</td>
<td>02 -</td>
</tr>
<tr>
<td>DFHI1NST2</td>
<td>Other</td>
<td>JCL to allocate and catalog CICS target and distribution libraries</td>
<td>02 -</td>
</tr>
<tr>
<td>DFHI1NST3</td>
<td>Other</td>
<td>JCL to allocate and catalog CICS RELFILE data sets</td>
<td>02 -</td>
</tr>
<tr>
<td>DFHI1NST4</td>
<td>Other</td>
<td>JCL to initialize CICS SMP/E data sets</td>
<td>02 -</td>
</tr>
<tr>
<td>DFHI1NST5</td>
<td>Other</td>
<td>JCL to RECEIVE the CICS base-level function SYSMOD</td>
<td>02 -</td>
</tr>
<tr>
<td>DFHI1NST6</td>
<td>Other</td>
<td>JCL to APPLY and ACCEPT the CICS base-level function SYSMOD</td>
<td>02 -</td>
</tr>
<tr>
<td>DFHI1NTRU</td>
<td>CSECT</td>
<td>Indoubt tool task related user exit</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHI1ONCD</td>
<td>Other</td>
<td>Replace DDDEFS for LE/370 or TCP/IP libraries in SMP/E target zone</td>
<td>02 -</td>
</tr>
<tr>
<td>DFHI1ONCL</td>
<td>Other</td>
<td>Relink-edit DFHRPRP load module outside SMP/E</td>
<td>02 -</td>
</tr>
</tbody>
</table>
### CICS directory

Table 123. CICS modules directory (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHIPCSP</td>
<td>Other</td>
<td>IPCS parmlib imbed member for DFHPDxxxx</td>
<td>-15</td>
</tr>
<tr>
<td>DFHIPDFS</td>
<td>CSECT</td>
<td>SDUMP formatter for kernel stack internal procedures</td>
<td>-03</td>
</tr>
<tr>
<td>DFHIPBS</td>
<td>Other</td>
<td>Interregion request</td>
<td>-11</td>
</tr>
<tr>
<td>DFHISR</td>
<td>CSECT</td>
<td>Interregion communication program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHIRPAD</td>
<td>Source</td>
<td>IRC dynamic add of connections routines</td>
<td>OS</td>
</tr>
<tr>
<td>DFHIRPC</td>
<td>Source</td>
<td>IRC connect and disconnect routines</td>
<td>OS</td>
</tr>
<tr>
<td>DFHIRPLC</td>
<td>Source</td>
<td>IRC clear and logoff routines</td>
<td>OS</td>
</tr>
<tr>
<td>DFHIRPD</td>
<td>Macro</td>
<td>IRC program internal control blocks</td>
<td>11</td>
</tr>
<tr>
<td>DFHIRPL</td>
<td>Source</td>
<td>IRC logon routines</td>
<td>OS</td>
</tr>
<tr>
<td>DFHIRPM</td>
<td>Source</td>
<td>IRC subroutines</td>
<td>OS</td>
</tr>
<tr>
<td>DFHIRPRQ</td>
<td>Source</td>
<td>IRC in-service and quiesce routines</td>
<td>OS</td>
</tr>
<tr>
<td>DFHIRPS</td>
<td>Source</td>
<td>IRC subroutines</td>
<td>OS</td>
</tr>
<tr>
<td>DFHISP</td>
<td>Source</td>
<td>IRC SRB processor</td>
<td>OS</td>
</tr>
<tr>
<td>DFHIRPSW</td>
<td>Source</td>
<td>IRC switch and pull routines</td>
<td>OS</td>
</tr>
<tr>
<td>DFHIRRDS</td>
<td>Source</td>
<td>Interregion session recovery data stream</td>
<td>11</td>
</tr>
<tr>
<td>DFHIRRDX</td>
<td>Sample</td>
<td>IRC XCF retry DIE subroutine</td>
<td>OS</td>
</tr>
<tr>
<td>DFHIRRXP</td>
<td>Sample</td>
<td>IRC XCF termination subroutine</td>
<td>OS</td>
</tr>
<tr>
<td>DFHIRRXS</td>
<td>Sample</td>
<td>IRC XCF SRB processor</td>
<td>OS</td>
</tr>
<tr>
<td>DFHIRSDS</td>
<td>DSECT</td>
<td>Interregion subsystem control blocks</td>
<td>11</td>
</tr>
<tr>
<td>DFHIRW10</td>
<td>CSECT</td>
<td>IRC work delivery exit program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHIS</td>
<td>Macro</td>
<td>ISC request</td>
<td>OS</td>
</tr>
<tr>
<td>DFHISCRQ</td>
<td>Macro</td>
<td>ISC request parameter list</td>
<td>11</td>
</tr>
<tr>
<td>DFHISMDK</td>
<td>Other</td>
<td>Intersystem communication program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHISP</td>
<td>CSECT</td>
<td>JCL to invoke DFHINST</td>
<td>OS</td>
</tr>
<tr>
<td>DFHIPVT</td>
<td>Other</td>
<td>IVP (batch) to verify CICS startup</td>
<td>OS</td>
</tr>
<tr>
<td>DFHIPVOL</td>
<td>Other</td>
<td>IVP to verify CICS running with DBCTL</td>
<td>OS</td>
</tr>
<tr>
<td>DFHJCP</td>
<td>CSECT</td>
<td>Journal control program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHJCR</td>
<td>Macro</td>
<td>Journal control area definition</td>
<td>11</td>
</tr>
<tr>
<td>DFHJCDLG</td>
<td>CSECT</td>
<td>Autocall SCEEOBJ</td>
<td>-03</td>
</tr>
<tr>
<td>DFHJCDLL</td>
<td>CSECT</td>
<td>Autocall SCEEOBJ</td>
<td>-03</td>
</tr>
<tr>
<td>DFHJCEMP</td>
<td>CSECT</td>
<td>-20</td>
<td></td>
</tr>
<tr>
<td>DFHJCEPA</td>
<td>DSECT</td>
<td>JGC parameter list</td>
<td>OS</td>
</tr>
<tr>
<td>DFHJCEPM</td>
<td>Macro</td>
<td>JGC request</td>
<td>OS</td>
</tr>
<tr>
<td>DFHJCEPT</td>
<td>CSECT</td>
<td>JGC trace interpretation data</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHJCEPX</td>
<td>Macro</td>
<td>JGC request (XPI)</td>
<td>11</td>
</tr>
<tr>
<td>DFHJCEPY</td>
<td>DSECT</td>
<td>JGC parameter list (XPI)</td>
<td>11</td>
</tr>
<tr>
<td>DFHJCEP</td>
<td>CSECT</td>
<td>Journal control program</td>
<td>-03</td>
</tr>
<tr>
<td>DFHJCR</td>
<td>Macro</td>
<td>Journal control record</td>
<td>11</td>
</tr>
<tr>
<td>DFHJCEST</td>
<td>CSECT</td>
<td>-03</td>
<td></td>
</tr>
<tr>
<td>DFHJIMP</td>
<td>CSECT</td>
<td>-03</td>
<td></td>
</tr>
<tr>
<td>DFHJHPAT</td>
<td>Sample</td>
<td>Java Hotpooling Pre-Call URM</td>
<td>-19</td>
</tr>
<tr>
<td>DFHJUCP</td>
<td>CSECT</td>
<td>Journal control print utility</td>
<td>03</td>
</tr>
<tr>
<td>DFHJVCV0</td>
<td>CSECT</td>
<td>-03</td>
<td></td>
</tr>
<tr>
<td>DFHJVMA</td>
<td>CSECT</td>
<td>Autocall SCEEOBJ</td>
<td>-03</td>
</tr>
<tr>
<td>DFHJVMAT</td>
<td>Sample</td>
<td>CICS JVM Interface user replaceable module</td>
<td>-19</td>
</tr>
<tr>
<td>DFHJVMPT</td>
<td>Other</td>
<td>-09</td>
<td></td>
</tr>
<tr>
<td>DFHJVMPS</td>
<td>Other</td>
<td>-09</td>
<td></td>
</tr>
<tr>
<td>DFHJVTETR</td>
<td>CSECT</td>
<td>-03</td>
<td></td>
</tr>
<tr>
<td>DFHKC</td>
<td>Macro</td>
<td>Task service request</td>
<td>11</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
<td>Library</td>
</tr>
<tr>
<td>--------------</td>
<td>--------</td>
<td>--------------------------------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>DFHKCQ</td>
<td>CSECT</td>
<td>Transaction manager - secondary requests</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHKCRP</td>
<td>CSECT</td>
<td>Task control restart program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHKSCS</td>
<td>CSECT</td>
<td>DFHKCQ chain scanning for discard</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHKSCSA</td>
<td>DSECT</td>
<td>KCSC parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHKSCSM</td>
<td>Macro</td>
<td>KCSC request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHKSCST</td>
<td>CSECT</td>
<td>KCSC trace interpretation data</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHKSCSP</td>
<td>CSECT</td>
<td>Task SRB control program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHKKEALI</td>
<td>Macro</td>
<td>KE domain - label alignment</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHKKEAR</td>
<td>CSECT (OCO)</td>
<td>KE domain - MVS ARM support services</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHKKEARA</td>
<td>DSECT</td>
<td>KEAR parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHKKEARM</td>
<td>Macro</td>
<td>KEAR request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHKKEART</td>
<td>CSECT (OCO)</td>
<td>KEAR trace interpretation data</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHKEDECL</td>
<td>CSECT (OCO)</td>
<td>KE domain - domain call request handler</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHKEEDD</td>
<td>CSECT (OCO)</td>
<td>KE domain - domain definition services</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHKEDDA</td>
<td>DSECT</td>
<td>KEDD parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHKEDDM</td>
<td>Macro</td>
<td>KEDD request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHKEDDT</td>
<td>CSECT (OCO)</td>
<td>KEDD trace interpretation data</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHKEDRT</td>
<td>CSECT (OCO)</td>
<td>KE domain - domain return request handler</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHKEDS</td>
<td>CSECT (OCO)</td>
<td>KE domain - dispatcher interfaces</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHKEDSA</td>
<td>DSECT</td>
<td>KEDS parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHKEDSI</td>
<td>Macro</td>
<td>KE domain - optimize kernel path lengths</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHKEDSM</td>
<td>Macro</td>
<td>KEDS request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHKEDST</td>
<td>CSECT (OCO)</td>
<td>KEDS trace interpretation data</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHKEDSX</td>
<td>Macro</td>
<td>KEDS request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHKEDSY</td>
<td>CSECT</td>
<td>KEDS parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHKEDUF</td>
<td>CSECT (OCO)</td>
<td>SDUMP formatter for KE domain</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHKEEDA</td>
<td>CSECT (OCO)</td>
<td>KE domain - execute deferred abend</td>
<td>OS -</td>
</tr>
<tr>
<td>DFKEEENV</td>
<td>Macro</td>
<td>KE domain - declare/switch environment</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHKEGD</td>
<td>CSECT (OCO)</td>
<td>KE domain - global data services</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHKEGDA</td>
<td>DSECT</td>
<td>KEGD parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHKEGDM</td>
<td>Macro</td>
<td>KEGD request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHKEGDT</td>
<td>CSECT (OCO)</td>
<td>KEGD trace interpretation data</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHKEGIN</td>
<td>CSECT (OCO)</td>
<td>KE domain - initialization</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHKEINA</td>
<td>DSECT</td>
<td>KEIN parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHKEINM</td>
<td>Macro</td>
<td>KEIN request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHKEINT</td>
<td>CSECT (OCO)</td>
<td>KEIN trace interpretation data</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHKEKCL</td>
<td>CSECT (OCO)</td>
<td>KE domain - LIFO push simulation</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHKELOC</td>
<td>CSECT (OCO)</td>
<td>KE domain - LIFO routine for locating domain anchors</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHKELRT</td>
<td>CSECT (OCO)</td>
<td>KE domain - LIFO return/pop simulation</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHKEMD</td>
<td>Macro</td>
<td>KE domain - domain/subroutine prolog code</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHKEPUB</td>
<td>DSECT</td>
<td>KE domain - some control blocks</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHKEKCD</td>
<td>CSECT (OCO)</td>
<td>KE domain - kernel error data construction</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHKERER</td>
<td>CSECT (OCO)</td>
<td>KE domain - record error routine</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHKESET</td>
<td>CSECT (OCO)</td>
<td>KE domain - reset address service</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHKEKKE</td>
<td>CSECT (OCO)</td>
<td>KE domain - KERNERROR response handler</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHKEKRN</td>
<td>Macro</td>
<td>KE domain - generate call to kernel</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHKERPC</td>
<td>CSECT (OCO)</td>
<td>KE domain - recovery percolation</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHKERRI</td>
<td>CSECT (OCO)</td>
<td>KE domain - recovery invocation</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHKERQ</td>
<td>CSECT (OCO)</td>
<td>KE domain - recovery request service</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHкерРУ</td>
<td>CSECT (OCO)</td>
<td>KE domain - runaway task error handler</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHKERX</td>
<td>CSECT (OCO)</td>
<td>KE domain - recovery exit service</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHKESEL</td>
<td>CSECT (OCO)</td>
<td>KE domain - subroutine call handler</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHKESESFM</td>
<td>CSECT (OCO)</td>
<td>KE domain - disposable segments freemain</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHKESES</td>
<td>CSECT (OCO)</td>
<td>KE domain - new stack segments getmain</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHKESEP</td>
<td>CSECT (OCO)</td>
<td>KE domain - system initialization program</td>
<td>OS -</td>
</tr>
</tbody>
</table>
## CICS directory

**Table 123. CICS modules directory (continued)**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHKESRT</td>
<td>CSECT</td>
<td>KE domain - subroutine return handler</td>
<td>03</td>
</tr>
<tr>
<td>DFHKESRTP</td>
<td>DSECT</td>
<td>KE domain - kernel stack structure</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHKESTX</td>
<td>CSECT</td>
<td>KE domain - kernel ESTAE exit</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHKESVC</td>
<td>CSECT</td>
<td>KE domain - authorized service routine</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHKETAB</td>
<td>CSECT</td>
<td>KE domain - list of domains requiring preinitialization on CICS run</td>
<td>- 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>preinitialization on DFHSTUP run</td>
<td></td>
</tr>
<tr>
<td>DFHKETB2</td>
<td>CSECT</td>
<td>KE domain - list of domains requiring preinitialization on DFHSTUP run</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHKETCB</td>
<td>CSECT</td>
<td>KE domain - kernel TCB startup routine</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHKETIA</td>
<td>DSECT</td>
<td>KE domain - timer services</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHKETIM</td>
<td>Macro</td>
<td>KETI request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHKETIT</td>
<td>CSECT</td>
<td>KETI trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHKETIX</td>
<td>CSECT</td>
<td>KE domain - STIMER exit</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHKETXR</td>
<td>CSECT</td>
<td>KE ETXR</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHKEXM</td>
<td>CSECT</td>
<td>KE domain - XM domain services</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHKEXMA</td>
<td>DSECT</td>
<td>KEXM parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHKEXMM</td>
<td>Macro</td>
<td>KEXM request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHKEXMT</td>
<td>CSECT</td>
<td>KEXM trace interpretation data</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHKETRI</td>
<td>CSECT</td>
<td>Trace interpreter for KE domain</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHLANG</td>
<td>Other</td>
<td>List of National Languages for CICS</td>
<td>10 -</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- alias for MEULANG</td>
<td></td>
</tr>
<tr>
<td>DFHLDDM</td>
<td>CSECT</td>
<td>LD domain - initialization/termination</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHLDDMI</td>
<td>CSECT</td>
<td>LD domain - secondary initialization</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHLDDUF</td>
<td>DSECT</td>
<td>SDUMP formatter for LD domain</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHLGDS</td>
<td>DSECT</td>
<td>LD domain - global statistics</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHLGDS</td>
<td>DSECT</td>
<td>LD domain - global statistics</td>
<td>C2 07</td>
</tr>
<tr>
<td>DFHLGDS</td>
<td>DSECT</td>
<td>LD domain - global statistics</td>
<td>P2 -</td>
</tr>
<tr>
<td>DFHLDDA</td>
<td>DSECT</td>
<td>LDLD parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHLDDM</td>
<td>Macro</td>
<td>LDLD request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHLDDT</td>
<td>CSECT</td>
<td>LDLD trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHLDDLX</td>
<td>Macro</td>
<td>LDLD request (XPI)</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHLDDY</td>
<td>DSECT</td>
<td>LDLD parameter list (XPI)</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHLDD1</td>
<td>CSECT</td>
<td>LD domain - acquire/release/refresh</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHLDD2</td>
<td>CSECT</td>
<td>LD domain - define/delete</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHLDD3</td>
<td>CSECT</td>
<td>LD domain - general functions</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHLNT</td>
<td>CSECT</td>
<td>LD domain - storage notify handler</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHLDRDS</td>
<td>DSECT</td>
<td>LD domain - program statistics</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHLDRDS</td>
<td>DSECT</td>
<td>LD domain - program statistics</td>
<td>C2 07</td>
</tr>
<tr>
<td>DFHLST</td>
<td>CSECT</td>
<td>LD domain - statistics collection</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHLSDUA</td>
<td>DSECT</td>
<td>LDSU parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHLSDUM</td>
<td>Macro</td>
<td>LDSU request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHLSDUT</td>
<td>CSECT</td>
<td>LDSU trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHLSDVC</td>
<td>CSECT</td>
<td>LD domain - authorized service routine</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHLSTRI</td>
<td>CSECT</td>
<td>Trace interpreter for LD domain</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHLAS</td>
<td>CSECT</td>
<td>ADD SUBPOOL service</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHLDS</td>
<td>CSECT</td>
<td>DELETE SUBPOOL service</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHLG</td>
<td>CSECT</td>
<td>Transaction Dump service</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHLF</td>
<td>CSECT</td>
<td>GETMAIN service</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHLFQ</td>
<td>CSECT</td>
<td>Quickcell freemain service</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHLG</td>
<td>CSECT</td>
<td>GETMAIN service</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHLGQ</td>
<td>CSECT</td>
<td>Quickcell getmain service</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHLE</td>
<td>CSECT</td>
<td>Runtime options service</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHLSRV</td>
<td>Macro</td>
<td>CICS Service routine vector</td>
<td>11 -</td>
</tr>
</tbody>
</table>
### Table 123. CICS modules directory (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHLETR</td>
<td>CSECT</td>
<td>Trace service routine</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHLETRM</td>
<td>Macro</td>
<td>LE Trace Service invocation macro</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHLFM</td>
<td>Macro</td>
<td>LIFO macro</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHLFT</td>
<td>Macro</td>
<td>LIFO trace macro</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHLFX</td>
<td>Macro</td>
<td>LIFO stack entry</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHLGBA</td>
<td>Macro</td>
<td>LGBA parameter list</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHLGAM</td>
<td>Macro</td>
<td>LGBA request</td>
<td>0S -</td>
</tr>
<tr>
<td>DFHLGABT</td>
<td>DSECT</td>
<td>LGBA translate tables</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHLGCBT</td>
<td>DSECT</td>
<td>LGCB translate tables</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHLGCCA</td>
<td>CSECT</td>
<td>LGCC parameter list</td>
<td>0S -</td>
</tr>
<tr>
<td>DFHLGCCM</td>
<td>Macro</td>
<td>LGCC request</td>
<td>0S -</td>
</tr>
<tr>
<td>DFHLGCT</td>
<td>DSECT</td>
<td>LGCC translate tables</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHLGDM</td>
<td>CSECT</td>
<td>Logger domain - domain initialization</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHLGDFU</td>
<td>CSECT</td>
<td>Log Manager domain dump formatting</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHLGFLD</td>
<td>DSECT</td>
<td>Log Manager log of log format</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHLGFD</td>
<td>DSECT</td>
<td>Log Manager general log format</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHLGGL</td>
<td>CSECT</td>
<td>Log Manager general log gate module</td>
<td>- 05</td>
</tr>
<tr>
<td>DFHLGGLA</td>
<td>DSECT</td>
<td>LGLA parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHLGGLI</td>
<td>CSECT</td>
<td>Journal number to name conversion</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHLGGLM</td>
<td>Macro</td>
<td>LGLM request</td>
<td>0S -</td>
</tr>
<tr>
<td>DFHLGGLT</td>
<td>DSECT</td>
<td>LGLT translate tables</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHLGICV</td>
<td>CSECT</td>
<td>LG SSI log record conversion to old format</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHLGIQT</td>
<td>DSECT</td>
<td>LG LOGR SSI dataset GET exit</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHLGILA</td>
<td>CSECT</td>
<td>LG Subsystem exit - lexical analyzer</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHLGIMS</td>
<td>CSECT</td>
<td>LG Subsystem exit - syntax message composer</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHLGIPA</td>
<td>DSECT</td>
<td>LG Subsystem exit - parser</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHLGIPI</td>
<td>DSECT</td>
<td>LG Subsystem exit - parse interface routine</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHLGISM</td>
<td>CSECT</td>
<td>LG Subsystem exit - parse message exits</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHLGJN</td>
<td>CSECT</td>
<td>Log Manager journal inventory gate module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHLGJNT</td>
<td>DSECT</td>
<td>LGJN translate tables</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHLGLBA</td>
<td>DSECT</td>
<td>LGLB parameter list</td>
<td>0S -</td>
</tr>
<tr>
<td>DFHLGLBM</td>
<td>Macro</td>
<td>LGLB request</td>
<td>0S -</td>
</tr>
<tr>
<td>DFHLGLBT</td>
<td>DSECT</td>
<td>LGLBT translate tables</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHLGLD</td>
<td>DSECT</td>
<td>Log Manager JournalModel gate</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHLGLDT</td>
<td>DSECT</td>
<td>LGDL translate tables</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHLGMSD</td>
<td>CSECT</td>
<td>Log Manager MVS SMF log format</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHLGMVA</td>
<td>CSECT</td>
<td>LGMV parameter list</td>
<td>0S -</td>
</tr>
<tr>
<td>DFHLGMVM</td>
<td>Macro</td>
<td>LGVM request</td>
<td>0S -</td>
</tr>
<tr>
<td>DFHLGVMT</td>
<td>DSECT</td>
<td>LGMV translate tables</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHLGPA</td>
<td>CSECT</td>
<td>Logger Domain - inquire/set parameters</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHLGPA</td>
<td>CSECT</td>
<td>LGPA parameter list</td>
<td>0S -</td>
</tr>
<tr>
<td>DFHLGPA</td>
<td>Macro</td>
<td>LGPA request</td>
<td>0S -</td>
</tr>
<tr>
<td>DFHLGPA</td>
<td>DSECT</td>
<td>LGPA translate tables</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHLGPA</td>
<td>Macro</td>
<td>Log Manager parameter parameter PLIST</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHLGPA</td>
<td>DSECT</td>
<td>Log Manager parameter manager manager PLIST</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHLGQC</td>
<td>CSECT</td>
<td>Log Manager RLS cleanup</td>
<td>- 05</td>
</tr>
<tr>
<td>DFHLGRDS</td>
<td>CSECT</td>
<td>Log Manager journal statistics</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHLGRDS</td>
<td>CSECT</td>
<td>Log Manager journal statistics</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHLGSC</td>
<td>CSECT</td>
<td>Log Manager statistics collection</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHLGS</td>
<td>CSECT</td>
<td>Log Manager logstream statistics</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHLGS</td>
<td>CSECT</td>
<td>Log Manager logstream statistics</td>
<td>C2 07</td>
</tr>
<tr>
<td>DFHLGSRA</td>
<td>CSECT</td>
<td>LGRS parameter list</td>
<td>0S -</td>
</tr>
<tr>
<td>DFHLGSR</td>
<td>DSECT</td>
<td>LGRS translate tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHLGSSI</td>
<td>CSECT</td>
<td>Log Manager LOGR SSI dataset exit</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHLGST</td>
<td>CSECT</td>
<td>Log Manager stream connection gate</td>
<td>- 03</td>
</tr>
</tbody>
</table>
### CICS directory

#### Table 123. CICS modules directory (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHLGSTT</td>
<td>DSECT (OCO)</td>
<td>LGST translate tables</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHLGTRI</td>
<td>CSECT (OCO)</td>
<td>Logger - trace interpretation</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHLGFMT</td>
<td>DSECT</td>
<td>LGHF translate tables</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHLFIO</td>
<td>DSECT</td>
<td>KE domain - LIFO control blocks</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHLILBD</td>
<td>Source</td>
<td>Language interface program language block</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHLILIA</td>
<td>Source</td>
<td>Language interface parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHLILII</td>
<td>Source</td>
<td>AP domain - Perform goto call to language interface</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHLILIM</td>
<td>Source</td>
<td>Language interface services</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHLILIT</td>
<td>CSECT (OCO)</td>
<td>Language interface trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHLIRET</td>
<td>CSECT (OCO)</td>
<td>Language interface return program</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHLITRI</td>
<td>CSECT (OCO)</td>
<td>Language interface trace interpreter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHLIWAD</td>
<td>Source</td>
<td>Language interface work area</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHLI000</td>
<td>Macro</td>
<td></td>
<td>11 -</td>
</tr>
<tr>
<td>DFHLI1DC</td>
<td>DSECT</td>
<td>Local logical device code table</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHLIDL1</td>
<td>DSECT</td>
<td>DL1 call level api macro (alias of CALDL1)</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHLMDM</td>
<td>CSECT (OCO)</td>
<td>LM domain - initialization/termination</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHLMDUF</td>
<td>CSECT (OCO)</td>
<td>SDUMP formatter for LM domain</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHLMIQ</td>
<td>CSECT (OCO)</td>
<td>LM domain - browse and inquiry</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHLMIQA</td>
<td>DSECT</td>
<td>LMIQ parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHLMIQM</td>
<td>Macro</td>
<td>LMIQ request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHLMIQT</td>
<td>CSECT (OCO)</td>
<td>LMIQ trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHLMLM</td>
<td>CSECT (OCO)</td>
<td>LM domain - services</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHLMLMA</td>
<td>DSECT</td>
<td>LMLM parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHLMLMI</td>
<td>CSECT</td>
<td></td>
<td>OS -</td>
</tr>
<tr>
<td>DFHLMLMM</td>
<td>Macro</td>
<td>LMLM request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHLMLMT</td>
<td>CSECT (OCO)</td>
<td>LMLM trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHLMTRI</td>
<td>CSECT (OCO)</td>
<td>Trace interpreter for LM domain</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHLNKS</td>
<td>Other</td>
<td>Cataloged procedure to link-edit CICS programs and application programs</td>
<td>18 -</td>
</tr>
<tr>
<td>DFHLOCK</td>
<td>Macro</td>
<td>KE domain - lock/unlock TCB entry</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHLONGN</td>
<td>Other</td>
<td>LD d11load long name conversion</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHLPUMD</td>
<td>Other</td>
<td>JCL to RECEIVE and APPLY the DFHGUUMOD</td>
<td>02 -</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SMP/E USERMOD</td>
<td></td>
</tr>
<tr>
<td>DFHLSCU</td>
<td>CSECT</td>
<td></td>
<td>- 03</td>
</tr>
<tr>
<td>DFHLSINT</td>
<td>CSECT</td>
<td></td>
<td>- 03</td>
</tr>
<tr>
<td>DFHLTRC</td>
<td>CSECT</td>
<td>Local terminal recovery module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHLUC</td>
<td>Macro</td>
<td>LU6.2 service request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHLUCM</td>
<td>Macro</td>
<td>LU6.2 migration request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHLUS</td>
<td>Macro</td>
<td>LU6.2 services manager driver macro</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHL2BA</td>
<td>CSECT (OCO)</td>
<td>Log Manager LGBA gate</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHL2BL1</td>
<td>CSECT (OCO)</td>
<td>Logger block initialize class procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHL2BL2</td>
<td>CSECT (OCO)</td>
<td>Logger block restore current position</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHL2BS1</td>
<td>CSECT (OCO)</td>
<td>Obtain and initialize BrowseableStream class data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHL2BS2</td>
<td>CSECT (OCO)</td>
<td>Construct a BrowseableStream object and return to caller</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHL2BS3</td>
<td>CSECT (OCO)</td>
<td>Destroy a BrowseableStream object</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHL2BS4</td>
<td>CSECT (OCO)</td>
<td>Terminate all browseable stream instances known to BrowseableStream class</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHL2CB</td>
<td>CSECT (OCO)</td>
<td>Log Manager LGCB gate</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHL2CC</td>
<td>CSECT (OCO)</td>
<td>Log Manager LGCC gate</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHL2CHA</td>
<td>CSECT (OCO)</td>
<td>Logger chain start browse all procedure</td>
<td>- 03</td>
</tr>
</tbody>
</table>
Table 123. CICS modules directory (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHL2CHE</td>
<td>CSECT (OCO)</td>
<td>Logger chain delete history procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHL2CHG</td>
<td>CSECT (OCO)</td>
<td>Logger chain get next chain procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHL2CHH</td>
<td>CSECT (OCO)</td>
<td>Logger chain start browse chains procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHL2CHI</td>
<td>CSECT (OCO)</td>
<td>Logger chain end browse chains procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHL2CHL</td>
<td>CSECT (OCO)</td>
<td>Logger chain end browse all procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHL2CHM</td>
<td>CSECT (OCO)</td>
<td>Logger chain move procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHL2CHN</td>
<td>CSECT (OCO)</td>
<td>Logger chain browse all get next procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHL2CHO</td>
<td>CSECT (OCO)</td>
<td></td>
<td>- 03</td>
</tr>
<tr>
<td>DFHL2CHP</td>
<td>CSECT (OCO)</td>
<td></td>
<td>- 03</td>
</tr>
<tr>
<td>DFHL2CHR</td>
<td>CSECT (OCO)</td>
<td>Logger chain restore procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHL2CHS</td>
<td>CSECT (OCO)</td>
<td>Logger chain set history procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHL2CH1</td>
<td>CSECT (OCO)</td>
<td>Logger chain initialize class procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHL2CH2</td>
<td>CSECT (OCO)</td>
<td>Logger chain create fresh procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHL2CH3</td>
<td>CSECT (OCO)</td>
<td>Logger chain start chain browse procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHL2CH4</td>
<td>CSECT (OCO)</td>
<td>Logger chain browse get next procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHL2CH5</td>
<td>CSECT (OCO)</td>
<td>Logger chain end chain browse procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHL2DOM</td>
<td>CSECT (OCO)</td>
<td>Log Manager L2 domain management</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHL2DOU</td>
<td>CSECT (OCO)</td>
<td>Log Manager L2_Dump_Formating_Module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHL2DB2</td>
<td>CSECT (OCO)</td>
<td></td>
<td>- 03</td>
</tr>
<tr>
<td>DFHL2H2S</td>
<td>CSECT (OCO)</td>
<td>Logger HardStream write MVS retry intro.</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHL2H5G</td>
<td>CSECT (OCO)</td>
<td>Logger HardStream read browse cursor</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHL2H5J</td>
<td>CSECT (OCO)</td>
<td>Logger HardStream end browse cursor</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHL2H5S</td>
<td>CSECT (OCO)</td>
<td>Logger HardStream connect procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHL2H53</td>
<td>CSECT (OCO)</td>
<td>Logger HardStream disconnect procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHL2H54</td>
<td>CSECT (OCO)</td>
<td>Logger HardStream delete all procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHL2HS5</td>
<td>CSECT (OCO)</td>
<td>Logger HardStream delete history procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHL2HS6</td>
<td>CSECT (OCO)</td>
<td>Logger HardStream start browse cursor</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHL2HS7</td>
<td>CSECT (OCO)</td>
<td>Logger HardStream start read procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHL2HS8</td>
<td>CSECT (OCO)</td>
<td>Logger HardStream read block procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHL2HS9</td>
<td>CSECT (OCO)</td>
<td>Logger HardStream end read procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHL2LB</td>
<td>CSECT (OCO)</td>
<td>Log Manager LGLB gate</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHL2MVC</td>
<td>CSECT (OCO)</td>
<td>Log Manager LGMV gate</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHL2T0F</td>
<td>CSECT (OCO)</td>
<td>Logger object factory initialize procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHL2SLE</td>
<td>CSECT (OCO)</td>
<td>Logger system log notify failure method</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHL2SLN</td>
<td>CSECT (OCO)</td>
<td>Logger system log open stream method</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHL2SL1</td>
<td>CSECT (OCO)</td>
<td>Logger system log initialize class procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHL2SR</td>
<td>CSECT (OCO)</td>
<td>Log Manager stream class</td>
<td>- 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>class declaration</td>
<td></td>
</tr>
<tr>
<td>DFHL2SR1</td>
<td>CSECT (OCO)</td>
<td>Logger stream class initialize class</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHL2SR2</td>
<td>CSECT (OCO)</td>
<td>Logger stream class construct procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHL2SR3</td>
<td>CSECT (OCO)</td>
<td>Logger stream class destruct procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHL2SR4</td>
<td>CSECT (OCO)</td>
<td>Logstream statistics module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHL2SR5</td>
<td>CSECT (OCO)</td>
<td>Logger stream class terminate all procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHL2T12</td>
<td>CSECT (OCO)</td>
<td></td>
<td>- 03</td>
</tr>
<tr>
<td>DFHL2TRA</td>
<td>CSECT (OCO)</td>
<td>Log Manager trace interpretation</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHL2VP1</td>
<td>CSECT (OCO)</td>
<td>Logger storage manager initialize class</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHL2WF</td>
<td>CSECT (OCO)</td>
<td>Log Manager LGWF gate</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHMAPDS</td>
<td>DSECT</td>
<td>BMS map description</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHMAPS</td>
<td>Other</td>
<td>Cataloged procedure to prepare physical and symbolic maps</td>
<td>18 -</td>
</tr>
<tr>
<td>DFHMAPT</td>
<td>Other</td>
<td></td>
<td>18 -</td>
</tr>
<tr>
<td>DFHMBCDS</td>
<td>DSECT</td>
<td>Transient data buffer control</td>
<td>0S -</td>
</tr>
<tr>
<td>DFHMBMWBI DSECT</td>
<td>File control DFHMBMWBI parameter list</td>
<td>0S -</td>
<td></td>
</tr>
<tr>
<td>DFHMBMWBI Macro</td>
<td>File control buffer management inline</td>
<td>0S -</td>
<td></td>
</tr>
<tr>
<td>DFHMCAD</td>
<td>Macro</td>
<td>Map control area</td>
<td>11 -</td>
</tr>
</tbody>
</table>
### CICS directory

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHMCBDS</td>
<td>DSECT</td>
<td>BMS message control block</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHMCBP</td>
<td>CSECT</td>
<td>BMS mapping control program</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHMCPAG</td>
<td>CSECT</td>
<td>BMS mapping control program (standard)</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHMCPG</td>
<td>CSECT</td>
<td>BMS minimum function mapping control</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHMCPGQ</td>
<td>CSECT</td>
<td>BMS mapping control program (minimum)</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHMCPIN</td>
<td>CSECT</td>
<td>BMS input mapping request handler</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHMCPPLK</td>
<td>Macro</td>
<td>Linkage to BMS modules</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHMCPPLG</td>
<td>CSECT</td>
<td>BMS mapping control program (full)</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHMCRDS</td>
<td>DSECT</td>
<td>BMS message control record</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHMCT</td>
<td>Macro</td>
<td>Monitoring control table</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHMCTAG</td>
<td>Sample</td>
<td>Monitoring control table for an AOR</td>
<td>19 -</td>
</tr>
<tr>
<td>DFHMCTDR</td>
<td>Macro</td>
<td>Monitoring dictionary definition</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHMCTDS</td>
<td>Macro</td>
<td>MCT root section definition</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHMCTDT</td>
<td>Macro</td>
<td>Transaction monitoring field and dictionary entry definition</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHMCTF</td>
<td>Sample</td>
<td>Monitoring control table for an FOR</td>
<td>19 -</td>
</tr>
<tr>
<td>DFHMCTMP</td>
<td>Macro</td>
<td>MCT class macro</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHMCTNM</td>
<td>Macro</td>
<td>Monitoring numeric string check</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHMCTSE</td>
<td>Macro</td>
<td>MCT option entry generator</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHMCTTG</td>
<td>Sample</td>
<td>Monitoring control table for a TOR</td>
<td>19 -</td>
</tr>
<tr>
<td>DFHMCT2G</td>
<td>Sample</td>
<td>Monitoring control table</td>
<td>19 03</td>
</tr>
<tr>
<td>DFHMCTX</td>
<td>CSECT</td>
<td>BMS fast path module</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHMCTY</td>
<td>CSECT</td>
<td>Process MAPPINGDEV Requests</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHMDC</td>
<td>Macro</td>
<td>Build C language symbolic description map</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHMDCM</td>
<td>Macro</td>
<td>Convert C field names to lowercase</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHMDF</td>
<td>Macro</td>
<td>Generate BMS field definition</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHMDI</td>
<td>Macro</td>
<td>Generate BMS map definition</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHMDX</td>
<td>Macro</td>
<td></td>
<td>11 -</td>
</tr>
<tr>
<td>DFHMEXC</td>
<td>CSECT</td>
<td>ME domain - DFHACxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>simplified Chinese version</td>
<td></td>
</tr>
<tr>
<td>DFHMEEACE</td>
<td>CSECT</td>
<td>ME domain - DFHACxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEEACK</td>
<td>CSECT</td>
<td>ME domain - DFHACxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEEADC</td>
<td>CSECT</td>
<td>ME domain - DFHACxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>simplified Chinese version</td>
<td></td>
</tr>
<tr>
<td>DFHMEEADE</td>
<td>CSECT</td>
<td>ME domain - DFHADxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEEADK</td>
<td>CSECT</td>
<td>ME domain - DFHADxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEEAI</td>
<td>CSECT</td>
<td>ME domain - DFHAIxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>simplified Chinese version</td>
<td></td>
</tr>
<tr>
<td>DFHMEEAIE</td>
<td>CSECT</td>
<td>ME domain - DFHAIxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEEAIK</td>
<td>CSECT</td>
<td>ME domain - DFHAIxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEEAMC</td>
<td>CSECT</td>
<td>ME domain - DFHACxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>simplified Chinese version</td>
<td></td>
</tr>
<tr>
<td>DFHMEEAME</td>
<td>CSECT</td>
<td>ME domain - DFHAMxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEEAMK</td>
<td>CSECT</td>
<td>ME domain - DFHAMxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEEAPC</td>
<td>CSECT</td>
<td>ME domain - DFHAxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>simplified Chinese version</td>
<td></td>
</tr>
<tr>
<td>DFHMEEAPE</td>
<td>CSECT</td>
<td>ME domain - DFHAPxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEEAPK</td>
<td>CSECT</td>
<td>ME domain - DFHAPxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEEAU</td>
<td>CSECT</td>
<td>ME domain - DFHAXxxxx message set</td>
<td>14 -</td>
</tr>
<tr>
<td>DFHMEBAC</td>
<td>CSECT</td>
<td>ME domain - DFHAXxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>simplified Chinese version</td>
<td></td>
</tr>
<tr>
<td>DFHMEBAE</td>
<td>CSECT</td>
<td>ME domain - DFHAXxxxx message set</td>
<td>14 03</td>
</tr>
</tbody>
</table>
### Table 123. CICS modules directory (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHMEBAK</td>
<td>CSECT (OCO)</td>
<td>ME domain - DFHBAnnn message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEBM</td>
<td>CSECT (OCO)</td>
<td>ME domain - batch message program</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHMEBMA</td>
<td>DSECT</td>
<td>MEBM parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHMEBMM</td>
<td>Macro</td>
<td>MEBM request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHMEBMT</td>
<td>CSECT (OCO)</td>
<td>MEBM trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHMEBRC</td>
<td>CSECT (OCO)</td>
<td>ME domain</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEBRE</td>
<td>CSECT (OCO)</td>
<td>ME domain</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEBRRK</td>
<td>CSECT (OCO)</td>
<td>ME domain</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEBU</td>
<td>CSECT (OCO)</td>
<td>ME domain - build message</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHMEBUA</td>
<td>DSECT</td>
<td>MEBU parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHMEBUM</td>
<td>Macro</td>
<td>MEBU request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHMEBUT</td>
<td>CSECT (OCO)</td>
<td>MEBU trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHMECAC</td>
<td>CSECT</td>
<td>ME domain - message set for GC/LC domains</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMECAE</td>
<td>CSECT</td>
<td>ME domain - DFHCAnnn message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMECAK</td>
<td>CSECT</td>
<td>ME domain - DFHCAnnn message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMECCC</td>
<td>CSECT</td>
<td>ME domain - DFHCCnnn message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMECCE</td>
<td>CSECT</td>
<td>ME domain - DFHCAnnn message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMECCEC</td>
<td>CSECT</td>
<td>ME domain - DFHCAnnn message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMECCEE</td>
<td>CSECT</td>
<td>ME domain - DFHCAnnn message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMECEK</td>
<td>CSECT (OCO)</td>
<td>ME domain - DFHCAnnn message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMECFE</td>
<td>CSECT</td>
<td>ME domain - DFHCAnnn message set</td>
<td>14 -</td>
</tr>
<tr>
<td>DFHMECPC</td>
<td>CSECT</td>
<td>ME domain - DFHCAnnn message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMECEPE</td>
<td>CSECT</td>
<td>ME domain - DFHCAnnn message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMECPK</td>
<td>CSECT (OCO)</td>
<td>ME domain - DFHCAnnn message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMECRC</td>
<td>CSECT</td>
<td>ME domain - DFHCAnnn message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMECREE</td>
<td>CSECT</td>
<td>ME domain - DFHCAnnn message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMECZC</td>
<td>CSECT</td>
<td>ME domain - DFHCAnnn message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMECZC</td>
<td>CSECT</td>
<td>ME domain - DFHCAnnn message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEDBC</td>
<td>CSECT</td>
<td>ME domain - DFHCBnnn message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEDBE</td>
<td>CSECT</td>
<td>ME domain - DFHCBnnn message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEDDK</td>
<td>CSECT (OCO)</td>
<td>ME domain - message set for DD domain</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEDHC</td>
<td>CSECT</td>
<td>ME domain - message set for DD domain</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEDHE</td>
<td>CSECT</td>
<td>ME domain - message set for DD domain</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEDDK</td>
<td>CSECT</td>
<td>ME domain - message set for DH domain</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEDMK</td>
<td>CSECT</td>
<td>ME domain - message set for DM domain</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEDME</td>
<td>CSECT</td>
<td>ME domain - message set for DM domain</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEDMK</td>
<td>CSECT</td>
<td>ME domain - message set for DM domain</td>
<td>14 03</td>
</tr>
</tbody>
</table>
### Table 123. CICS modules directory (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHMEDSC</td>
<td>CSECT</td>
<td>ME domain - message set for DS domain</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEDSE</td>
<td>CSECT</td>
<td>ME domain - message set for DS domain</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEDSK</td>
<td>CSECT</td>
<td>ME domain - message set for DS domain</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEDUC</td>
<td>CSECT</td>
<td>ME domain - message set for DU domain</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEDUE</td>
<td>CSECT (OCO)</td>
<td>ME domain - message set for DU domain</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEDUF</td>
<td>CSECT (OCO)</td>
<td>SDUMP formatter for ME domain</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHMEDUK</td>
<td>CSECT (OCO)</td>
<td>ME domain - message set for DU domain</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEDXC</td>
<td>CSECT</td>
<td>ME domain - DFHDXxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEDXE</td>
<td>CSECT</td>
<td>ME domain - DFHDXxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEDXK</td>
<td>CSECT (OCO)</td>
<td>ME domain - DFHDXxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEEC</td>
<td>CSECT</td>
<td>ME domain - DFHEJxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEJE</td>
<td>CSECT</td>
<td>ME domain - DFHEJxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEEMC</td>
<td>CSECT</td>
<td>ME domain - DFHEMxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEEME</td>
<td>CSECT</td>
<td>ME domain - DFHEMxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEEMK</td>
<td>CSECT (OCO)</td>
<td>ME domain - DFHEMxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEERC</td>
<td>CSECT</td>
<td>ME domain - DFHERxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEERE</td>
<td>CSECT</td>
<td>ME domain - DFHERxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEERK</td>
<td>CSECT</td>
<td>ME domain - DFHERxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEEXE</td>
<td>CSECT</td>
<td>ME domain - DFHEXxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEFAC</td>
<td>CSECT</td>
<td>ME domain - DFHFAxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEFAE</td>
<td>CSECT</td>
<td>ME domain - DFHFAxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEFAK</td>
<td>CSECT (OCO)</td>
<td>ME domain - DFHFAxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEFBC</td>
<td>CSECT</td>
<td>ME domain - DFHFBBxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEFBE</td>
<td>CSECT</td>
<td>ME domain - DFHFBBxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEFBB</td>
<td>CSECT (OCO)</td>
<td>ME domain - DFHFBBxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEFCC</td>
<td>CSECT</td>
<td>ME domain - DFHFCxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEFCE</td>
<td>CSECT</td>
<td>ME domain - DFHFCxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEFCK</td>
<td>CSECT (OCO)</td>
<td>ME domain - DFHFCxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEFDC</td>
<td>CSECT</td>
<td>ME domain - DFHFDCxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEFDE</td>
<td>CSECT</td>
<td>ME domain - DFHFDCxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEFDE</td>
<td>CSECT (OCO)</td>
<td>ME domain - DFHFDCxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEFEC</td>
<td>CSECT</td>
<td>ME domain - DFHFECxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEFCEE</td>
<td>CSECT</td>
<td>ME domain - DFHFECxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEFEE</td>
<td>CSECT (OCO)</td>
<td>ME domain - DFHFECxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEFED</td>
<td>CSECT</td>
<td>ME domain - format message subroutine</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHMEFOD</td>
<td>DSECT</td>
<td>MFO parameter list</td>
<td>05 -</td>
</tr>
<tr>
<td>DFHMEFOM</td>
<td>Macro</td>
<td>MFO request</td>
<td>05 -</td>
</tr>
<tr>
<td>DFHMEFOT</td>
<td>CSECT (OCO)</td>
<td>MFO trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHMEICC</td>
<td>CSECT</td>
<td>ME domain - DFHICxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEICE</td>
<td>CSECT</td>
<td>ME domain - DFHICxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEICK</td>
<td>CSECT (OCO)</td>
<td>ME domain - DFHICxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEIEC</td>
<td>CSECT</td>
<td>ME domain - DFHIExxxx message set</td>
<td>14 - simplified Chinese version</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
<td>Library</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
<td>--------------------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>DFHMEIEE</td>
<td>CSECT</td>
<td>ME domain - DFHIExxxx message set</td>
<td>14</td>
</tr>
<tr>
<td>DFHMEIEK</td>
<td>CSECT (OCO)</td>
<td>ME domain - DFHIExxxx message set</td>
<td>14</td>
</tr>
<tr>
<td>DFHMEIEIC</td>
<td>CSECT</td>
<td>ME domain - DFHIIxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>simplified Chinese version</td>
<td></td>
</tr>
<tr>
<td>DFHMEIEI</td>
<td>CSECT</td>
<td>ME domain - DFHIIxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEIEK</td>
<td>CSECT (OCO)</td>
<td>ME domain - DFHIIxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEIN</td>
<td>CSECT</td>
<td>ME domain - inquire message data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHMEINA</td>
<td>DSECT</td>
<td>MEIN parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHMEINC</td>
<td>DSECT</td>
<td>ME domain - DFHINxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>simplified Chinese version</td>
<td></td>
</tr>
<tr>
<td>DFHMEINE</td>
<td>DSECT</td>
<td>ME domain - DFHINxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEINK</td>
<td>DSECT</td>
<td>ME domain - DFHINxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Japanese (Kanji) version</td>
<td></td>
</tr>
<tr>
<td>DFHMEINM</td>
<td>Macro</td>
<td>MEIN request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHMEINT</td>
<td>CSECT (OCO)</td>
<td>MEIN trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHMEIRC</td>
<td>CSECT</td>
<td>ME domain - DFHIRxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>simplified Chinese version</td>
<td></td>
</tr>
<tr>
<td>DFHMEIRE</td>
<td>CSECT</td>
<td>ME domain - DFHIRxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEIRK</td>
<td>CSECT (OCO)</td>
<td>ME domain - DFHIRxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Japanese (Kanji) version</td>
<td></td>
</tr>
<tr>
<td>DFHMEJCC</td>
<td>CSECT</td>
<td>ME domain - DFHJCxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>simplified Chinese version</td>
<td></td>
</tr>
<tr>
<td>DFHMEJCE</td>
<td>CSECT</td>
<td>ME domain - DFHJCxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEJCK</td>
<td>CSECT (OCO)</td>
<td>ME domain - DFHJCxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEKCC</td>
<td>CSECT</td>
<td>ME domain - DFHKCxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>simplified Chinese version</td>
<td></td>
</tr>
<tr>
<td>DFHMEKCE</td>
<td>CSECT</td>
<td>ME domain - DFHKCxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Japanese (Kanji) version</td>
<td></td>
</tr>
<tr>
<td>DFHMEKEC</td>
<td>CSECT</td>
<td>ME domain - DFHKExxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>simplified Chinese version</td>
<td></td>
</tr>
<tr>
<td>DFHMEKEE</td>
<td>CSECT</td>
<td>ME domain - message set for KE domain</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEKEK</td>
<td>CSECT</td>
<td>ME domain - message set for KE domain</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMELEDC</td>
<td>CSECT</td>
<td>ME domain - DFHLDxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>simplified Chinese version</td>
<td></td>
</tr>
<tr>
<td>DFHMELEDE</td>
<td>CSECT</td>
<td>ME domain - message set for LD domain</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMELEDK</td>
<td>CSECT</td>
<td>ME domain - message set for LD domain</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMELGCC</td>
<td>CSECT</td>
<td>ME domain - DFHGLxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>simplified Chinese version</td>
<td></td>
</tr>
<tr>
<td>DFHMELEGE</td>
<td>CSECT</td>
<td>ME domain - DFHGLxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMELEGK</td>
<td>CSECT</td>
<td>ME domain - DFHGLxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Japanese (Kanji) version</td>
<td></td>
</tr>
<tr>
<td>DFHMELMC</td>
<td>CSECT</td>
<td>ME domain - DFHLMxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>simplified Chinese version</td>
<td></td>
</tr>
<tr>
<td>DFHMELMC</td>
<td>CSECT</td>
<td>ME domain - message set for LM domain</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>simplified Chinese version</td>
<td></td>
</tr>
<tr>
<td>DFHMEMECK</td>
<td>CSECT (OCO)</td>
<td>ME domain - main functions</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHMEMECC</td>
<td>CSECT</td>
<td>ME domain - DFHMExxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>simplified Chinese version</td>
<td></td>
</tr>
</tbody>
</table>
### Table 123. CICS modules directory (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHMEMEA</td>
<td>DSECT</td>
<td>MEME parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHMEMEE</td>
<td>CSECT</td>
<td>ME domain - DFHMExxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEMEK</td>
<td>CSECT (OCO)</td>
<td>ME domain - main functions</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEMEM</td>
<td>Macro</td>
<td>MEME request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHMEMET</td>
<td>CSECT (OCO)</td>
<td>MEME trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHMEMNC</td>
<td>CSECT</td>
<td>ME domain - DFHMNxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEMNE</td>
<td>CSECT</td>
<td>ME domain - message set for MN domain</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEMNK</td>
<td>CSECT</td>
<td>ME domain - message set for MN domain</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEMUC</td>
<td>CSECT</td>
<td>ME domain - DFHMUxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEMUE</td>
<td>CSECT</td>
<td>ME domain - DFHMUxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMENQK</td>
<td>CSECT</td>
<td>ME domain - DFHMQxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMENE</td>
<td>CSECT</td>
<td>ME domain - DFHMUxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMENQK</td>
<td>CSECT</td>
<td>ME domain - DFHMQxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEOQC</td>
<td>CSECT</td>
<td>ME domain - DFHMQxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEOQE</td>
<td>CSECT</td>
<td>ME domain - DFHOTxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEOQK</td>
<td>CSECT</td>
<td>ME domain - DFHOTxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEOQC</td>
<td>CSECT</td>
<td>ME domain - DFHOTxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMENPQC</td>
<td>CSECT</td>
<td>ME domain - DFHPCxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMENPGC</td>
<td>CSECT</td>
<td>ME domain - DFHPCxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMENPGK</td>
<td>CSECT (OCO)</td>
<td>ME domain - DFHPCxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMENPQK</td>
<td>CSECT (OCO)</td>
<td>ME domain - DFHPCxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMENPC</td>
<td>CSECT</td>
<td>ME domain - DFHRxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMENPE</td>
<td>CSECT</td>
<td>ME domain - DFHRxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMENPK</td>
<td>CSECT (OCO)</td>
<td>ME domain - DFHRxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMENPS</td>
<td>CSECT</td>
<td>ME domain - DFHRxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMENPM</td>
<td>CSECT</td>
<td>ME domain - DFHRxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMENPM</td>
<td>CSECT</td>
<td>ME domain - DFHRxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMENPM</td>
<td>CSECT</td>
<td>ME domain - DFHRxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMERDC</td>
<td>CSECT</td>
<td>ME domain - DFHROxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMERDC</td>
<td>CSECT (OCO)</td>
<td>ME domain - DFHROxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMERD</td>
<td>CSECT</td>
<td>ME domain - DFHROxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMERD</td>
<td>CSECT (OCO)</td>
<td>ME domain - DFHROxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMERD</td>
<td>CSECT (OCO)</td>
<td>ME domain - DFHROxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMERD</td>
<td>CSECT (OCO)</td>
<td>ME domain - DFHROxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMERD</td>
<td>CSECT (OCO)</td>
<td>ME domain - DFHROxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMERD</td>
<td>CSECT (OCO)</td>
<td>ME domain - DFHROxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMERD</td>
<td>CSECT (OCO)</td>
<td>ME domain - DFHROxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMERD</td>
<td>CSECT (OCO)</td>
<td>ME domain - DFHROxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMERD</td>
<td>CSECT (OCO)</td>
<td>ME domain - DFHROxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMERD</td>
<td>CSECT (OCO)</td>
<td>ME domain - DFHROxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMERD</td>
<td>CSECT (OCO)</td>
<td>ME domain - DFHROxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMERD</td>
<td>CSECT (OCO)</td>
<td>ME domain - DFHROxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMERD</td>
<td>CSECT (OCO)</td>
<td>ME domain - DFHROxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMERD</td>
<td>CSECT (OCO)</td>
<td>ME domain - DFHROxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMERD</td>
<td>CSECT (OCO)</td>
<td>ME domain - DFHROxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMERD</td>
<td>CSECT (OCO)</td>
<td>ME domain - DFHROxxxx message set</td>
<td>14 03</td>
</tr>
</tbody>
</table>
Table 123. CICS modules directory (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHMERO</td>
<td>CSECT</td>
<td>ME domain - DFHRPxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEROE</td>
<td>CSECT</td>
<td>ME domain - DFHRPxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEROK</td>
<td>CSECT (OCO)</td>
<td>ME domain - DFHRPxxxx message set</td>
<td>Japanese (Kanji) version</td>
</tr>
<tr>
<td>DFHMERPC</td>
<td>CSECT</td>
<td>ME domain - DFHRPxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMERP</td>
<td>CSECT (OCO)</td>
<td>ME domain - DFHRPxxxx message set</td>
<td>Japanese (Kanji) version</td>
</tr>
<tr>
<td>DFHMEROQC</td>
<td>CSECT</td>
<td>ME domain - DFHRPxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEROQE</td>
<td>CSECT</td>
<td>ME domain - DFHRPxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEROQK</td>
<td>CSECT (OCO)</td>
<td>ME domain - DFHRPxxxx message set</td>
<td>Japanese (Kanji) version</td>
</tr>
<tr>
<td>DFHMERQC</td>
<td>CSECT</td>
<td>ME domain - DFHRPxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMERC</td>
<td>CSECT (OCO)</td>
<td>ME domain - DFHRPxxxx message set</td>
<td>Japanese (Kanji) version</td>
</tr>
<tr>
<td>DFHMERTC</td>
<td>CSECT</td>
<td>ME domain - DFHRTxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMERTK</td>
<td>CSECT (OCO)</td>
<td>ME domain - DFHRTxxxx message set</td>
<td>Japanese (Kanji) version</td>
</tr>
<tr>
<td>DFHMERCUC</td>
<td>CSECT</td>
<td>ME domain - DFHRUxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMERCUE</td>
<td>CSECT</td>
<td>ME domain - DFHRUxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMERCUK</td>
<td>CSECT (OCO)</td>
<td>ME domain - DFHRUxxxx message set</td>
<td>simplified Chinese version</td>
</tr>
<tr>
<td>DFHMERCX</td>
<td>CSECT</td>
<td>ME domain - DFHRXxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMERCXE</td>
<td>CSECT</td>
<td>ME domain - DFHRXxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMERCXK</td>
<td>CSECT (OCO)</td>
<td>ME domain - DFHRXxxxx message set</td>
<td>simplified Chinese version</td>
</tr>
<tr>
<td>DFHMERCZE</td>
<td>CSECT</td>
<td>ME domain - DFHRZxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMERCZK</td>
<td>CSECT (OCO)</td>
<td>ME domain - DFHRZxxxx message set</td>
<td>simplified Chinese version</td>
</tr>
<tr>
<td>DFHMESH</td>
<td>CSECT</td>
<td>ME domain - DFSHxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMESHK</td>
<td>CSECT</td>
<td>ME domain - DFSHxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMESIC</td>
<td>CSECT (OCO)</td>
<td>ME domain - DFSHxxxx message set</td>
<td>simplified Chinese version</td>
</tr>
<tr>
<td>DFHMESIE</td>
<td>CSECT</td>
<td>ME domain - DFSIxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMESIK</td>
<td>CSECT (OCO)</td>
<td>ME domain - DFSIxxxx message set</td>
<td>Japanese (Kanji) version</td>
</tr>
<tr>
<td>DFHMESJC</td>
<td>CSECT</td>
<td>ME domain - DFSJxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMESJE</td>
<td>CSECT</td>
<td>ME domain - DFSJxxxx message set</td>
<td>14 03</td>
</tr>
</tbody>
</table>
## CICS directory

Table 123. CICS modules directory (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHMESJK</td>
<td>CSECT (OCO)</td>
<td>ME domain - DFHSJxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Japanese (Kanji) version</td>
<td></td>
</tr>
<tr>
<td>DFHMESKC</td>
<td>CSECT</td>
<td>ME domain - DFHSKxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>simplified Chinese version</td>
<td></td>
</tr>
<tr>
<td>DFHMESKE</td>
<td>CSECT</td>
<td>ME domain - DFHSKxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>simplified Chinese version</td>
<td></td>
</tr>
<tr>
<td>DFHMESKK</td>
<td>CSECT</td>
<td>ME domain - DFHSKxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>simplified Chinese version</td>
<td></td>
</tr>
<tr>
<td>DFHMESMC</td>
<td>CSECT</td>
<td>ME domain - DFHSMxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>simplified Chinese version</td>
<td></td>
</tr>
<tr>
<td>DFHMESME</td>
<td>CSECT</td>
<td>ME domain - message set for SM domain</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Japanese (Kanji) version</td>
<td></td>
</tr>
<tr>
<td>DFHMESMK</td>
<td>CSECT</td>
<td>ME domain - message set for SM domain</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Japanese (Kanji) version</td>
<td></td>
</tr>
<tr>
<td>DFHMESNC</td>
<td>CSECT</td>
<td>ME domain - DFHSNxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>simplified Chinese version</td>
<td></td>
</tr>
<tr>
<td>DFHMESNE</td>
<td>CSECT</td>
<td>ME domain - DFHSNxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>simplified Chinese version</td>
<td></td>
</tr>
<tr>
<td>DFHMESOK</td>
<td>CSECT</td>
<td>ME domain - DFHSOxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>simplified Chinese version</td>
<td></td>
</tr>
<tr>
<td>DFHMESOE</td>
<td>CSECT</td>
<td>ME domain - DFHSOxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>simplified Chinese version</td>
<td></td>
</tr>
<tr>
<td>DFHMESOC</td>
<td>CSECT</td>
<td>ME domain - DFHSOxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>simplified Chinese version</td>
<td></td>
</tr>
<tr>
<td>DFHMESRE</td>
<td>CSECT</td>
<td>ME domain - SIT overrides collection</td>
<td>03</td>
</tr>
<tr>
<td>DFHMESRA</td>
<td>DSECT</td>
<td>ME domain - DFHSRxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>simplified Chinese version</td>
<td></td>
</tr>
<tr>
<td>DFHMESRC</td>
<td>CSECT</td>
<td>ME domain - DFHSRxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>simplified Chinese version</td>
<td></td>
</tr>
<tr>
<td>DFHMESTC</td>
<td>CSECT</td>
<td>ME domain - message set for ST domain</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>simplified Chinese version</td>
<td></td>
</tr>
<tr>
<td>DFHMESTE</td>
<td>CSECT</td>
<td>ME domain - message set for ST domain</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>simplified Chinese version</td>
<td></td>
</tr>
<tr>
<td>DFHMESZC</td>
<td>CSECT (OCO)</td>
<td>ME domain - DFHSZxxxx message set (FEPI)</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>simplified Chinese version</td>
<td></td>
</tr>
<tr>
<td>DFHMESZE</td>
<td>CSECT (OCO)</td>
<td>ME domain - DFHSZxxxx message set (FEPI)</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>simplified Chinese version</td>
<td></td>
</tr>
<tr>
<td>DFHMESZK</td>
<td>CSECT (OCO)</td>
<td>ME domain - DFHSZxxxx message set (FEPI)</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>simplified Chinese version</td>
<td></td>
</tr>
<tr>
<td>DFHMETCC</td>
<td>CSECT</td>
<td>ME domain - DFHTCxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>simplified Chinese version</td>
<td></td>
</tr>
<tr>
<td>DFHMETCE</td>
<td>CSECT</td>
<td>ME domain - DFHTCxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>simplified Chinese version</td>
<td></td>
</tr>
<tr>
<td>DFHMETCK</td>
<td>CSECT (OCO)</td>
<td>ME domain - DFHTCxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>simplified Chinese version</td>
<td></td>
</tr>
<tr>
<td>DFHMETDC</td>
<td>CSECT</td>
<td>ME domain - DFHTDxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>simplified Chinese version</td>
<td></td>
</tr>
<tr>
<td>DFHMETDE</td>
<td>CSECT</td>
<td>ME domain - DFHTDxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>simplified Chinese version</td>
<td></td>
</tr>
<tr>
<td>DFHMETDK</td>
<td>CSECT</td>
<td>ME domain - DFHTDxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>simplified Chinese version</td>
<td></td>
</tr>
<tr>
<td>DFHMETF</td>
<td>CSECT</td>
<td>ME domain - DFHTFxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>simplified Chinese version</td>
<td></td>
</tr>
<tr>
<td>DFHMETFC</td>
<td>CSECT</td>
<td>ME domain - DFHTFxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>simplified Chinese version</td>
<td></td>
</tr>
<tr>
<td>DFHMETF</td>
<td>CSECT</td>
<td>ME domain - DFHTFxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>simplified Chinese version</td>
<td></td>
</tr>
<tr>
<td>DFHMETF</td>
<td>CSECT</td>
<td>ME domain - DFHTFxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>simplified Chinese version</td>
<td></td>
</tr>
<tr>
<td>DFHMETIC</td>
<td>CSECT</td>
<td>ME domain - DFHTIxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>simplified Chinese version</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
<td>Library</td>
</tr>
<tr>
<td>--------------</td>
<td>------</td>
<td>--------------------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>DFHMETIE</td>
<td>CSECT</td>
<td>ME domain - message set for TI domain</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMETIK</td>
<td>CSECT</td>
<td>ME domain - message set for TI domain</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMETMC</td>
<td>CSECT</td>
<td>ME domain - DFHTMxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>simplified Chinese version</td>
<td></td>
</tr>
<tr>
<td>DFHMETME</td>
<td>CSECT</td>
<td>ME domain - DFHTMxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Japanese (Kanji) version</td>
<td></td>
</tr>
<tr>
<td>DFHMETOK</td>
<td>CSECT</td>
<td>ME domain - DFHTOxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Japanese (Kanji) version</td>
<td></td>
</tr>
<tr>
<td>DFHMETPC</td>
<td>CSECT</td>
<td>ME domain - DFHTPxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>simplified Chinese version</td>
<td></td>
</tr>
<tr>
<td>DFHMETPE</td>
<td>CSECT</td>
<td>ME domain - DFHTPxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DFHMETRC</td>
<td>CSECT</td>
<td>ME domain - message set for TR domain</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>simplified Chinese version</td>
<td></td>
</tr>
<tr>
<td>DFHMETRE</td>
<td>CSECT</td>
<td>ME domain - message set for TR domain</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMETRI</td>
<td>CSECT</td>
<td>Trace interpreter for ME domain</td>
<td>- 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DFHMETSK</td>
<td>CSECT</td>
<td>ME domain - DFHTSxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>simplified Chinese version</td>
<td></td>
</tr>
<tr>
<td>DFHMETSE</td>
<td>CSECT</td>
<td>ME domain - DFHTSxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DFHMET1</td>
<td>CSECT</td>
<td>ME domain - DFHMET1x online message table</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Japanese (Kanji) version</td>
<td></td>
</tr>
<tr>
<td>DFHMET1E</td>
<td>CSECT</td>
<td>DFHMEU base messages link-edit module</td>
<td>14 -</td>
</tr>
<tr>
<td>DFHMET2</td>
<td>CSECT</td>
<td>ME domain - DFHMET2x offline translator</td>
<td>- 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>message table</td>
<td></td>
</tr>
<tr>
<td>DFHMET3</td>
<td>CSECT</td>
<td>ME domain - DFHMET3x offline message table</td>
<td>- 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>for DFHSTUP</td>
<td></td>
</tr>
<tr>
<td>DFHMET4</td>
<td>CSECT</td>
<td>Off-line message table for EXCI</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHMET5</td>
<td>CSECT</td>
<td>ME domain - DFHMET5x online message table</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHMET6</td>
<td>CSECT</td>
<td>ME domain - DFHMET6x online message table</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHMET5E</td>
<td>CSECT</td>
<td>DFHMEU ONC RPS messages link-edit module</td>
<td>14 -</td>
</tr>
<tr>
<td>DFHMET9</td>
<td>CSECT</td>
<td>ME domain - DFHMET9x online message table</td>
<td>0S 03</td>
</tr>
<tr>
<td>DFHMEU</td>
<td>CSECT</td>
<td>Message translation utility program</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHMEUA</td>
<td>DSECT</td>
<td>Message editing utility parameter list</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHMEUC</td>
<td>CSECT</td>
<td>Message editing utility copy message dataset</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHMEUCL</td>
<td>CSECT</td>
<td>Message editing utility copy message dataset06</td>
<td>-</td>
</tr>
<tr>
<td>DFHMEUD</td>
<td>CSECT</td>
<td>Message editing utility set/validate system</td>
<td>- 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>defaults</td>
<td></td>
</tr>
<tr>
<td>DFHMEUE</td>
<td>CSECT</td>
<td>Message editing utility edit message</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHMEUL</td>
<td>CSECT</td>
<td>Message editing utility compile, assemble</td>
<td>- 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>and link-edit message data sets</td>
<td></td>
</tr>
<tr>
<td>DFHMEULT</td>
<td>CSECT</td>
<td>Message editing utility CLIST to create language</td>
<td>06 -</td>
</tr>
<tr>
<td></td>
<td></td>
<td>codes table</td>
<td></td>
</tr>
<tr>
<td>DFHMEUM</td>
<td>Macro</td>
<td>Message editing utility ISPF editor profile</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHMEUP</td>
<td>CSECT</td>
<td>Message editing utility display PTF panel</td>
<td>- 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>and submit PTF job</td>
<td></td>
</tr>
<tr>
<td>DFHMEUPC</td>
<td>CSECT</td>
<td>ME domain - message set for UP domain</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>simplified Chinese version</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
<td>Library</td>
</tr>
<tr>
<td>---------</td>
<td>-------</td>
<td>------------------------------------------------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>DFHMEUPE</td>
<td>CSECT</td>
<td>ME domain - DFHUPxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEUPK</td>
<td>CSECT</td>
<td>ME domain - DFHUPxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEUSC</td>
<td>CSECT</td>
<td>Message editing utility check state of message data set simplified Chinese version</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEUSE</td>
<td>CSECT</td>
<td>Message editing utility check state of message data set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEUSK</td>
<td>CSECT</td>
<td>Message editing utility check state of message data set Japanese (Kanji) version</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEUU</td>
<td>CSECT</td>
<td>Message editing utility compare PTF and English message data sets</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHMEU00</td>
<td>CSECT</td>
<td>Message editing utility help index panel</td>
<td>16 -</td>
</tr>
<tr>
<td>DFHMEU01</td>
<td>CSECT</td>
<td>Message editing utility main help panel 1</td>
<td>16 -</td>
</tr>
<tr>
<td>DFHMEU10</td>
<td>CSECT</td>
<td>Message editing utility main panel</td>
<td>16 -</td>
</tr>
<tr>
<td>DFHMEU11</td>
<td>CSECT</td>
<td>Message editing utility main help panel 2</td>
<td>16 -</td>
</tr>
<tr>
<td>DFHMEU12</td>
<td>CSECT</td>
<td>Message editing utility main help panel 3</td>
<td>16 -</td>
</tr>
<tr>
<td>DFHMEU20</td>
<td>CSECT</td>
<td>Message editing utility set defaults panel (part 1 of 2)</td>
<td>16 -</td>
</tr>
<tr>
<td>DFHMEU21</td>
<td>CSECT</td>
<td>Message editing utility set defaults (part 1) help panel 1</td>
<td>16 -</td>
</tr>
<tr>
<td>DFHMEU22</td>
<td>CSECT</td>
<td>Message editing utility set defaults (part 1) help panel 2</td>
<td>16 -</td>
</tr>
<tr>
<td>DFHMEU30</td>
<td>CSECT</td>
<td>Message editing utility set defaults panel (part 2 of 2)</td>
<td>16 -</td>
</tr>
<tr>
<td>DFHMEU31</td>
<td>CSECT</td>
<td>Message editing utility set defaults (part 2) help panel</td>
<td>16 -</td>
</tr>
<tr>
<td>DFHMEU40</td>
<td>CSECT</td>
<td>Message editing utility language selection help panel</td>
<td>16 -</td>
</tr>
<tr>
<td>DFHMEU41</td>
<td>CSECT</td>
<td>Message editing utility language selection help panel</td>
<td>16 -</td>
</tr>
<tr>
<td>DFHMEU50</td>
<td>CSECT</td>
<td>Message editing utility message selection help panel</td>
<td>16 -</td>
</tr>
<tr>
<td>DFHMEU51</td>
<td>CSECT</td>
<td>Message editing utility message selection help panel</td>
<td>16 -</td>
</tr>
<tr>
<td>DFHMEU60</td>
<td>CSECT</td>
<td>Message editing utility message edit help panel</td>
<td>16 -</td>
</tr>
<tr>
<td>DFHMEU61</td>
<td>CSECT</td>
<td>Message editing utility message edit help panel</td>
<td>16 -</td>
</tr>
<tr>
<td>DFHMEU70</td>
<td>CSECT</td>
<td>Message editing utility apply PTF updates help panel</td>
<td>16 -</td>
</tr>
<tr>
<td>DFHMEU71</td>
<td>CSECT</td>
<td>Message editing utility apply PTF updates help panel</td>
<td>16 -</td>
</tr>
<tr>
<td>DFHMEWBC</td>
<td>CSECT</td>
<td>ME domain</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEWEB</td>
<td>CSECT</td>
<td>ME domain</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEWKB</td>
<td>CSECT</td>
<td>ME domain</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEXS</td>
<td>CSECT</td>
<td>ME domain - write symptom string to SYS1.LOGREC</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHMEXS</td>
<td>DSECT</td>
<td>MEWS parameter list</td>
<td>0S -</td>
</tr>
<tr>
<td>DFHMEXSM</td>
<td>Macro</td>
<td>MEWS request</td>
<td>0S -</td>
</tr>
<tr>
<td>DFHMEXST</td>
<td>CSECT</td>
<td>MEWS trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHMEXT</td>
<td>CSECT</td>
<td>ME domain - WTOR service routine</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHMEXTA</td>
<td>DSECT</td>
<td>MEWT parameter list</td>
<td>0S -</td>
</tr>
<tr>
<td>DFHMEXTM</td>
<td>Macro</td>
<td>MEWT request</td>
<td>0S -</td>
</tr>
<tr>
<td>DFHMEXTT</td>
<td>CSECT</td>
<td>MEWT trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHMEXAC</td>
<td>CSECT</td>
<td>ME domain - message set for XA domain simplified Chinese version</td>
<td>14 03</td>
</tr>
</tbody>
</table>
Table 123. CICS modules directory (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHMEXAE</td>
<td>CSECT</td>
<td>ME domain - DFHXAxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEXAX</td>
<td>CSECT</td>
<td>ME domain - DFHXAxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEXCC</td>
<td>CSECT</td>
<td>ME domain - message set for XC domain</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>simplified Chinese version</td>
<td></td>
</tr>
<tr>
<td>DFHMEXCE</td>
<td>CSECT</td>
<td>ME domain - DFHXCxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEXCK</td>
<td>CSECT</td>
<td>ME domain - DFHXCxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEXGC</td>
<td>CSECT</td>
<td>ME domain - DFHGXxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>simplified Chinese version</td>
<td></td>
</tr>
<tr>
<td>DFHMEXGE</td>
<td>CSECT</td>
<td>ME domain - DFHGXxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEXGK</td>
<td>CSECT</td>
<td>(OCO) ME domain - DFHGXxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Japanese (Kanji) version</td>
<td></td>
</tr>
<tr>
<td>DFHMEXMC</td>
<td>CSECT</td>
<td>ME domain - DFHXMxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>simplified Chinese version</td>
<td></td>
</tr>
<tr>
<td>DFHMEXME</td>
<td>CSECT</td>
<td>ME domain - DFHXMxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEXMK</td>
<td>CSECT</td>
<td>(OCO) ME domain - DFHXMxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Japanese (Kanji) version</td>
<td></td>
</tr>
<tr>
<td>DFHMEXOC</td>
<td>CSECT</td>
<td>ME domain - DFHXXxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>simplified Chinese version</td>
<td></td>
</tr>
<tr>
<td>DFHMEXOE</td>
<td>CSECT</td>
<td>ME domain - DFHXXxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEXOK</td>
<td>CSECT</td>
<td>ME domain - DFHXXxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEXQE</td>
<td>CSECT</td>
<td>ME domain - DFHXXxxxx message set</td>
<td>14 -</td>
</tr>
<tr>
<td>DFHMEXSC</td>
<td>CSECT</td>
<td>ME domain - DFHXXxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>simplified Chinese version</td>
<td></td>
</tr>
<tr>
<td>DFHMEXSE</td>
<td>CSECT</td>
<td>ME domain - DFHXXxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEXSK</td>
<td>CSECT</td>
<td>(OCO) ME domain - DFHXXxxxx message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Japanese (Kanji) version</td>
<td></td>
</tr>
<tr>
<td>DFHMEXAC</td>
<td>CSECT</td>
<td>ME domain - DFHZXXXX message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>simplified Chinese version</td>
<td></td>
</tr>
<tr>
<td>DFHMEXAE</td>
<td>CSECT</td>
<td>ME domain - DFHZXXXX message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEXAK</td>
<td>CSECT</td>
<td>(OCO) ME domain - DFHZXXXX message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Japanese (Kanji) version</td>
<td></td>
</tr>
<tr>
<td>DFHMEXBC</td>
<td>CSECT</td>
<td>ME domain - DFHZXXXX message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>simplified Chinese version</td>
<td></td>
</tr>
<tr>
<td>DFHMEXBE</td>
<td>CSECT</td>
<td>ME domain - DFHZXXXX message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEXBK</td>
<td>CSECT</td>
<td>(OCO) ME domain - DFHZXXXX message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Japanese (Kanji) version</td>
<td></td>
</tr>
<tr>
<td>DFHMEXCC</td>
<td>CSECT</td>
<td>ME domain - DFHZXXXX message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>simplified Chinese version</td>
<td></td>
</tr>
<tr>
<td>DFHMEXCE</td>
<td>CSECT</td>
<td>ME domain - DFHZXXXX message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEXCK</td>
<td>CSECT</td>
<td>(OCO) ME domain - DFHZXXXX message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Japanese (Kanji) version</td>
<td></td>
</tr>
<tr>
<td>DFHMEXDC</td>
<td>CSECT</td>
<td>ME domain - DFHZXXXX message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>simplified Chinese version</td>
<td></td>
</tr>
<tr>
<td>DFHMEXDE</td>
<td>CSECT</td>
<td>ME domain - DFHZXXXX message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEXDK</td>
<td>CSECT</td>
<td>(OCO) ME domain - DFHZXXXX message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Japanese (Kanji) version</td>
<td></td>
</tr>
<tr>
<td>DFHMEXEC</td>
<td>CSECT</td>
<td>ME domain - DFHZXXXX message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>simplified Chinese version</td>
<td></td>
</tr>
<tr>
<td>DFHMEXEE</td>
<td>CSECT</td>
<td>ME domain - DFHZXXXX message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEXEK</td>
<td>CSECT</td>
<td>(OCO) ME domain - DFHZXXXX message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Japanese (Kanji) version</td>
<td></td>
</tr>
<tr>
<td>DFHMEXNC</td>
<td>CSECT</td>
<td>ME domain - DFHZXXXX message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>simplified Chinese version</td>
<td></td>
</tr>
<tr>
<td>DFHMEXNE</td>
<td>CSECT</td>
<td>ME domain - DFHZXXXX message set</td>
<td>14 03</td>
</tr>
<tr>
<td>DFHMEXNK</td>
<td>CSECT</td>
<td>(OCO) ME domain - DFHZXXXX message set</td>
<td>14 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Japanese (Kanji) version</td>
<td></td>
</tr>
</tbody>
</table>
## CICS directory

Table 123. CICS modules directory (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DFHME00C</strong> CSECT</td>
<td>ME domain - NLS message language globals simplified Chinese version</td>
<td>14 03</td>
<td></td>
</tr>
<tr>
<td><strong>DFHME00E</strong> CSECT</td>
<td>ME domain - NLS message language globals simplified Chinese version</td>
<td>14 03</td>
<td></td>
</tr>
<tr>
<td><strong>DFHME00K</strong> CSECT (OCO)</td>
<td>ME domain - NLS message language globals Japanese (Kanji) version</td>
<td>14 03</td>
<td></td>
</tr>
<tr>
<td><strong>DFHME01E</strong> CSECT</td>
<td>ME domain - DFH1Uxx message set</td>
<td>14 03</td>
<td></td>
</tr>
<tr>
<td><strong>DFHME1UC</strong> CSECT</td>
<td>ME domain</td>
<td>14 03</td>
<td></td>
</tr>
<tr>
<td><strong>DFHME1UE</strong> CSECT</td>
<td>ME domain - DFH1Uxx message set</td>
<td>14 03</td>
<td></td>
</tr>
<tr>
<td><strong>DFHME1UK</strong> CSECT (OCO)</td>
<td>ME domain - DFH1Uxx message set</td>
<td>14 03</td>
<td></td>
</tr>
<tr>
<td><strong>DFHME42E</strong> CSECT</td>
<td>ME domain - DFH42xx message set</td>
<td>14 03</td>
<td></td>
</tr>
<tr>
<td><strong>DFHME70C</strong> CSECT</td>
<td>ME domain - DFH70xx message set simplified Chinese version</td>
<td>14 03</td>
<td></td>
</tr>
<tr>
<td><strong>DFHME70E</strong> CSECT</td>
<td>ME domain - DFH70xx message set simplified Chinese version</td>
<td>14 03</td>
<td></td>
</tr>
<tr>
<td><strong>DFHME70K</strong> CSECT (OCO)</td>
<td>ME domain - DFH70xx message set</td>
<td>14 03</td>
<td></td>
</tr>
<tr>
<td><strong>DFHME71C</strong> CSECT</td>
<td>ME domain - DFH71xx message set simplified Chinese version</td>
<td>14 03</td>
<td></td>
</tr>
<tr>
<td><strong>DFHME71E</strong> CSECT</td>
<td>ME domain - DFH71xx message set simplified Chinese version</td>
<td>14 03</td>
<td></td>
</tr>
<tr>
<td><strong>DFHME71K</strong> CSECT (OCO)</td>
<td>ME domain - DFH71xx message set</td>
<td>14 03</td>
<td></td>
</tr>
<tr>
<td><strong>DFHME42E</strong> CSECT</td>
<td>ME domain - DFH42xx message set</td>
<td>14 03</td>
<td></td>
</tr>
<tr>
<td><strong>DFHME72C</strong> CSECT</td>
<td>ME domain - DFH72xx message set simplified Chinese version</td>
<td>14 03</td>
<td></td>
</tr>
<tr>
<td><strong>DFHME72E</strong> CSECT</td>
<td>ME domain - DFH72xx message set simplified Chinese version</td>
<td>14 03</td>
<td></td>
</tr>
<tr>
<td><strong>DFHME72K</strong> CSECT (OCO)</td>
<td>ME domain - DFH72xx message set</td>
<td>14 03</td>
<td></td>
</tr>
<tr>
<td><strong>DFHME73E</strong> CSECT</td>
<td>ME domain - DFH73xx message set</td>
<td>14 03</td>
<td></td>
</tr>
<tr>
<td><strong>DFHME74E</strong> CSECT</td>
<td>ME domain - DFH74xx message set</td>
<td>14 03</td>
<td></td>
</tr>
<tr>
<td><strong>DFHME75E</strong> CSECT</td>
<td>ME domain - DFH75xx message set</td>
<td>14 03</td>
<td></td>
</tr>
<tr>
<td><strong>DFHME76E</strong> CSECT</td>
<td>ME domain - DFH76xx message set</td>
<td>14 03</td>
<td></td>
</tr>
<tr>
<td><strong>DFHME77E</strong> CSECT</td>
<td>ME domain - DFH77xx message set</td>
<td>14 03</td>
<td></td>
</tr>
<tr>
<td><strong>DFHME78E</strong> CSECT</td>
<td>ME domain - DFH78xx message set</td>
<td>14 03</td>
<td></td>
</tr>
<tr>
<td><strong>DFHME79E</strong> CSECT</td>
<td>ME domain - DFH79xx message set</td>
<td>14 03</td>
<td></td>
</tr>
<tr>
<td><strong>DFHME80E</strong> CSECT</td>
<td>ME domain - DFH80xx message set</td>
<td>14 03</td>
<td></td>
</tr>
<tr>
<td><strong>DFHME81E</strong> CSECT</td>
<td>ME domain - DFH81xx message set</td>
<td>14 03</td>
<td></td>
</tr>
<tr>
<td><strong>DFHME82E</strong> CSECT</td>
<td>ME domain - DFH82xx message set</td>
<td>14 03</td>
<td></td>
</tr>
<tr>
<td><strong>DFHME83E</strong> CSECT</td>
<td>ME domain - DFH83xx message set</td>
<td>14 03</td>
<td></td>
</tr>
<tr>
<td><strong>DFHME84E</strong> CSECT</td>
<td>ME domain - DFH84xx message set</td>
<td>14 03</td>
<td></td>
</tr>
<tr>
<td><strong>DFHME85E</strong> CSECT</td>
<td>ME domain - DFH85xx message set</td>
<td>14 03</td>
<td></td>
</tr>
<tr>
<td><strong>DFHME86E</strong> CSECT</td>
<td>ME domain - DFH86xx message set</td>
<td>14 03</td>
<td></td>
</tr>
<tr>
<td><strong>DFHME87E</strong> CSECT</td>
<td>ME domain - DFH87xx message set</td>
<td>14 03</td>
<td></td>
</tr>
<tr>
<td><strong>DFHME88E</strong> CSECT</td>
<td>ME domain - DFH88xx message set</td>
<td>14 03</td>
<td></td>
</tr>
<tr>
<td><strong>DFHME89E</strong> CSECT</td>
<td>ME domain - DFH89xx message set</td>
<td>14 03</td>
<td></td>
</tr>
<tr>
<td><strong>DFHME90E</strong> CSECT</td>
<td>ME domain - DFH90xx message set</td>
<td>14 03</td>
<td></td>
</tr>
</tbody>
</table>

**DFHMGM** CSECT DFHMGP NLS message support

**DFHMGP00** CSECT DFHMGP error message find

**DFHMGT** Message generation table

**DFHMGT01** CSECT Subsystem interface message table segment

**DFHMGT20** CSECT Message generation table segment

**DFHMGT21** CSECT Message generation table segment

**DFHMGT22** CSECT Message generation table segment

**DFHMGT24** CSECT Message generation table segment

**DFHMGT26** CSECT Message generation table segment

**DFHMGT33** CSECT Message generation table segment

**DFHMGT34** CSECT Message generation table segment

**DFHMGT35** CSECT Message generation table segment

**DFHMGT37** CSECT Message generation table segment

**DFHMGT44** CSECT Message generation table segment

**DFHMGT49** CSECT Message generation table segment

**DFHMGT50** CSECT Message generation table segment

**DFHMGT51** CSECT Message generation table segment

**DFHMGT90** CSECT Message generation table segment

**DFHMNW** Source BMS 3270 input mapping

**DFHMNRS** CSECT ISC request shipping - mirror program

**DFHMNIR** Other

**DFHMNKS** CSECT Alias for MEUKEYS

**DFHMNL1** CSECT BMS LU1 printer mapping program

**DFHMN** Macro MN domain - inline request

**DFHMNDEF** Macro MN domain - some control blocks

**DFHMNMD** CSECT (OCO) MN domain - initialization/termination

**DFHMNDF** CSECT (OCO) MN domain - SDUMP formatter for MN domain

**DFHMNDUP** CSECT (OCO) Monitoring dictionary utility

**DFHMNEXC** Macro MN domain - monitoring exception record

**DFHMNGDS** DSECT MN domain - global statistics
### Table 123. CICS modules directory (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHMNGDS</td>
<td>DSECT</td>
<td>MN domain - global statistics</td>
<td>C2 07</td>
</tr>
<tr>
<td>DFHMNNN</td>
<td>CSECT (OCO)</td>
<td>MN domain - functions</td>
<td>- 05</td>
</tr>
<tr>
<td>DFHMNNNA</td>
<td>DSECT</td>
<td>MNNN parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHMNNMM</td>
<td>Macro</td>
<td>MNNN request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHMNNMT</td>
<td>CSECT</td>
<td>MNNN trace interpretation data</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHMNNMX</td>
<td>Macro</td>
<td>MNNN request (XPI)</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHMNNNY</td>
<td>DSECT</td>
<td>MNNN parameter list (XPI)</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHMNNNT</td>
<td>CSECT (OCO)</td>
<td>MN domain - XM notify gate</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHMNPBI</td>
<td>Macro</td>
<td>MN domain - access to MVS WLM performance block token</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHMNPD A</td>
<td>CSECT</td>
<td>Monitoring facility performance class record</td>
<td>19 -</td>
</tr>
<tr>
<td>DFHMNSMF</td>
<td>Macro</td>
<td>MN domain - monitoring SMF header and SMF product section</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHMNSR</td>
<td>CSECT (OCO)</td>
<td>MN domain - services</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHMNSRA</td>
<td>DSECT</td>
<td>MNSR parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHMNSRM</td>
<td>Macro</td>
<td>MNSR request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHMNSRT</td>
<td>CSECT</td>
<td>MNSR trace interpretation data</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHMNST</td>
<td>CSECT (OCO)</td>
<td>MN domain - statistics services</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHMNSU</td>
<td>CSECT (OCO)</td>
<td>MN domain - subroutines</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHMNSUA</td>
<td>DSECT</td>
<td>MNSU parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHMNSUM</td>
<td>Macro</td>
<td>MNSU request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHMNSUT</td>
<td>CSECT</td>
<td>MNSU trace interpretation data</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHMNSVC</td>
<td>CSECT (OCO)</td>
<td>MN domain - authorized service routine</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHMNTDS</td>
<td>DSECT</td>
<td>MN domain - transaction monitoring data</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHMNTDS</td>
<td>DSECT</td>
<td>MN domain - transaction monitoring data</td>
<td>C2 07</td>
</tr>
<tr>
<td>DFHMNTI</td>
<td>CSECT (OCO)</td>
<td>MN domain - timer gate</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHMNTRI</td>
<td>CSECT (OCO)</td>
<td>Trace interpreter for MN domain</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHMNUE</td>
<td>CSECT (OCO)</td>
<td>MN domain - user exit service</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHMNXM</td>
<td>CSECT (OCO)</td>
<td>MN domain functional gate</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHMNXMT</td>
<td>DSECT</td>
<td>MNXM translate tables</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHMMOVE</td>
<td>Macro</td>
<td>Domain call argument MOVE macro</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHMMPARS</td>
<td>CSECT</td>
<td>Parameter syntax checking</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHMRCDS</td>
<td>DSECT</td>
<td>Transient data VSAM control</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHMROUF</td>
<td>CSECT (OCO)</td>
<td>MRO SDUMP formatter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHMROQM</td>
<td>Macro</td>
<td>MRO work queue manager interface</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHMROQP</td>
<td>CSECT</td>
<td>MRO work queue manager - enable/disable</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHMROSM</td>
<td>Macro</td>
<td>MRO work queue manager quickcell interface</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHMROQDS</td>
<td>DSECT</td>
<td>MRO work queue manager control blocks</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHMRXM</td>
<td>CSECT</td>
<td>TF XM transaction attach</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHMSCAN</td>
<td>CSECT</td>
<td>Macro scan utility</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHMSE D</td>
<td>Macro</td>
<td>Generate BMS map set definition</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHMSET</td>
<td>CSECT</td>
<td>Parameter syntax checking record</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHMSG</td>
<td>Macro</td>
<td>Generate a message</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHMSGIF</td>
<td>CSECT</td>
<td>CZ Direct_to_CICS</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHMSG00</td>
<td>CSECT</td>
<td>MEU MEU00x message set (alias MEU00)</td>
<td>12 -</td>
</tr>
<tr>
<td>DFHMSG01</td>
<td>CSECT</td>
<td>MEU MEU01x message set (alias MEU01)</td>
<td>12 -</td>
</tr>
<tr>
<td>DFHMSG02</td>
<td>CSECT</td>
<td>MEU MEU02x message set (alias MEU02)</td>
<td>12 -</td>
</tr>
<tr>
<td>DFHMSG03</td>
<td>CSECT</td>
<td>MEU MEU03x message set (alias MEU03)</td>
<td>12 -</td>
</tr>
<tr>
<td>DFHMSG04</td>
<td>CSECT</td>
<td>MEU MEU04x message set (alias MEU04)</td>
<td>12 -</td>
</tr>
<tr>
<td>DFHMSG05</td>
<td>CSECT</td>
<td>MEU MEU05x message set (alias MEU05)</td>
<td>12 -</td>
</tr>
<tr>
<td>DFHMSGEN</td>
<td>Macro</td>
<td>Generate messages in BMS modules</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHMSP</td>
<td>CSECT</td>
<td>Message switching program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHMSPUT</td>
<td>Macro</td>
<td>Put messages to terminals in BMS</td>
<td>OS -</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
<td>Library</td>
</tr>
<tr>
<td>------------</td>
<td>----------</td>
<td>--------------------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>DFHMSRCA</td>
<td>Symbolic</td>
<td>Magnetic slot reader control values</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHMSRCA</td>
<td>Symbolic</td>
<td>Magnetic slot reader control values</td>
<td>C2 07</td>
</tr>
<tr>
<td>DFHMSRCA</td>
<td>Symbolic</td>
<td>Magnetic slot reader control values</td>
<td>D3 08</td>
</tr>
<tr>
<td>DFHMSX</td>
<td>Symbolic</td>
<td></td>
<td>11 -</td>
</tr>
<tr>
<td>DFHMVRMS</td>
<td>CSECT (OCO)</td>
<td>MVS recovery/termination manager RESMGR exit stub</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHMWCDS</td>
<td>DSECT</td>
<td>Transient data wait control</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHMXP</td>
<td>CSECT</td>
<td>Local queuing shipper</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHM32</td>
<td>CSECT</td>
<td>BMS 3270 mapping</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHM32AG</td>
<td>CSECT</td>
<td>BMS 3270 mapping (standard)</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHM32IG</td>
<td>CSECT</td>
<td>BMS 3270 mapping (full)</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHNCASM</td>
<td>Macro</td>
<td>Named counter service interface</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHNC</td>
<td>DSECT</td>
<td>Named counter service interface</td>
<td>- 08</td>
</tr>
<tr>
<td>DFHNCNF</td>
<td>DSECT</td>
<td>Named counter service interface</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHNCNB</td>
<td>DSECT</td>
<td>Named counter service interface</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHNCDF</td>
<td>DSECT</td>
<td>Named counter server AXM definitions</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHNCN</td>
<td>DSECT</td>
<td>NC ENF event interface</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHNCSEU</td>
<td>Macro</td>
<td>Named counter server interface</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHNCIF</td>
<td>CSECT</td>
<td>Named counter server interface</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHNCMN</td>
<td>CSECT</td>
<td>Named counter server main program</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHNCMS</td>
<td>CSECT</td>
<td>Named counter server messages</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHCN</td>
<td>Macro</td>
<td>Named counter option table definition</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHCNOP</td>
<td>CSECT</td>
<td>Named counter server operator commands</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHCNCOPT</td>
<td>CSECT</td>
<td>Named counter server sample option table</td>
<td>19 03</td>
</tr>
<tr>
<td>DFHCNCLI</td>
<td>CSECT</td>
<td>Named counter service interface</td>
<td>17 -</td>
</tr>
<tr>
<td>DFHCNCR</td>
<td>CSECT</td>
<td>Named counter server parameter routine</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHCNPS</td>
<td>CSECT</td>
<td>Named counter server pool selection</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHCNCL</td>
<td>CSECT</td>
<td>Named counter server pool reload</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHCNCQS</td>
<td>CSECT</td>
<td>Named counter server request routine</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHCNCRS</td>
<td>CSECT</td>
<td>NC ARM Restart Support</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHCNST</td>
<td>CSECT</td>
<td>Named counter server statistics support</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHNCUSD</td>
<td>Macro</td>
<td>Named counter server list str stats</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHNCXSD</td>
<td>Macro</td>
<td>Named counter server storage statistics</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHNCXTR</td>
<td>CSECT</td>
<td>Named counter server interface stub</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHCNUL</td>
<td>CSECT</td>
<td>Named counter server pool unload</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHNPCEA</td>
<td>DSECT</td>
<td>NEP communication area</td>
<td>D2 -</td>
</tr>
<tr>
<td>DFHNPCEA</td>
<td>Macro</td>
<td>NEP communication area</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHNQDM</td>
<td>CSECT</td>
<td>NQ domain management</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHNDQUDF</td>
<td>CSECT</td>
<td>NQ offline dump formatting</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHNDQED</td>
<td>CSECT</td>
<td>NQED format enqueue/dequeue</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHNDQEDA</td>
<td>CSECT</td>
<td>NQED parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHNDQEDM</td>
<td>Macro</td>
<td>NQED request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHNDQEDT</td>
<td>DSECT</td>
<td>NQED translate tables</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHNDQEDX</td>
<td>Macro</td>
<td></td>
<td>11 -</td>
</tr>
<tr>
<td>DFHNDQEDY</td>
<td>Macro</td>
<td></td>
<td>11 -</td>
</tr>
<tr>
<td>DFHNDQEDS</td>
<td>CSECT</td>
<td>NQ enqueue manager statistics</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHNDQEDS</td>
<td>CSECT</td>
<td>NQ enqueue manager statistics</td>
<td>C2 07</td>
</tr>
<tr>
<td>DFHNDQIB</td>
<td>CSECT</td>
<td>NQ inquire/browse module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHNDQIBA</td>
<td>CSECT</td>
<td>NQIB parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHNDQIBM</td>
<td>Macro</td>
<td>NQIB request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHNDQIBT</td>
<td>DSECT</td>
<td>NQIB translate tables</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHNDQIE</td>
<td>CSECT</td>
<td>NQ default enqueue interpreter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHNDQING</td>
<td>CSECT</td>
<td>NQ main functions</td>
<td>- 03</td>
</tr>
</tbody>
</table>
### Table 123. CICS modules directory (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHNQNQA</td>
<td>CSECT</td>
<td>NQNQ parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHNQNQM</td>
<td>Macro</td>
<td>NQNQ request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHNQNQT</td>
<td>DSECT</td>
<td>NQNQ translate tables</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHNQRN</td>
<td>CSECT</td>
<td>Sysplex resource names services</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHNQRNA</td>
<td>Other</td>
<td>NQRN interface parameter area</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHNQRNM</td>
<td>Macro</td>
<td>DFHNQRN interface macro</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHNQRNT</td>
<td>CSECT</td>
<td></td>
<td>- 03</td>
</tr>
<tr>
<td>DFHNQST</td>
<td>CSECT</td>
<td>(OCO) NQ statistics</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHNQTRI</td>
<td>CSECT</td>
<td>(OCO) NQ offline trace interpretation</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHNQURED</td>
<td>Macro</td>
<td>EXEC argument list for ENQ/DEQ user exits</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHNXDF</td>
<td>CSECT</td>
<td>(OCO) SDUMP control block index processor</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHOPSRC</td>
<td>Other</td>
<td>JCL to install optional source tapes</td>
<td>02 -</td>
</tr>
<tr>
<td>DFHOTCO</td>
<td>CSECT</td>
<td>OTCO CDURUN and Gate Module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHOTCOT</td>
<td>CSECT</td>
<td></td>
<td>- 03</td>
</tr>
<tr>
<td>DFHOTCPT</td>
<td>CSECT</td>
<td></td>
<td>- 03</td>
</tr>
<tr>
<td>DFHOTDM</td>
<td>CSECT</td>
<td>OT Domain Management</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHOTDUF</td>
<td>CSECT</td>
<td>OT Domain Dump Formatting</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHOTIS1</td>
<td>CSECT</td>
<td></td>
<td>- 03</td>
</tr>
<tr>
<td>DFHOTIS2</td>
<td>CSECT</td>
<td></td>
<td>- 03</td>
</tr>
<tr>
<td>DFHOTN</td>
<td>CSECT</td>
<td>OTS Resync Transaction</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHOTRM</td>
<td>CSECT</td>
<td>Run Transaction Syncpoint Processor</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHOTRPI</td>
<td>CSECT</td>
<td></td>
<td>- 03</td>
</tr>
<tr>
<td>DFHOTRS</td>
<td>CSECT</td>
<td>OTRS CDURUN and Gate Module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHOTRST</td>
<td>CSECT</td>
<td></td>
<td>- 03</td>
</tr>
<tr>
<td>DFHOTSU</td>
<td>CSECT</td>
<td>OTSU CDURUN and Gate Module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHOTSUT</td>
<td>CSECT</td>
<td></td>
<td>- 03</td>
</tr>
<tr>
<td>DFHOTTR</td>
<td>CSECT</td>
<td>OTTR CDURUN and Gate Module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHOTTTR</td>
<td>CSECT</td>
<td>OT Domain Trace Interpretation</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHOTLTV1</td>
<td>CSECT</td>
<td></td>
<td>- 03</td>
</tr>
<tr>
<td>DFHPADM</td>
<td>CSECT</td>
<td>(OCO) PA domain - initialization/termination</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHPADUF</td>
<td>CSECT</td>
<td>(OCO) SDUMP formatter for PA domain</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHPAGP</td>
<td>CSECT</td>
<td>(OCO) PA domain - get parameters service</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHPAGPA</td>
<td>DSECT</td>
<td>PAGP parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHPAGPM</td>
<td>Macro</td>
<td>PAGP request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHPAGPT</td>
<td>CSECT</td>
<td>(OCO) PAGP trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHPAOI</td>
<td>CSECT</td>
<td>(OCO) PA domain - communication with SYSIN</td>
<td>- 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>data set and operator console</td>
<td></td>
</tr>
<tr>
<td>DFHPAOIA</td>
<td>DSECT</td>
<td>PAIO parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHPAIO</td>
<td>Macro</td>
<td>PAIO request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHPAIO</td>
<td>DSECT</td>
<td>PAIO trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHPAPL</td>
<td>Macro</td>
<td>DBCTL architected parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHPASY</td>
<td>CSECT</td>
<td>(OCO) PA domain - system initialization</td>
<td>- 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>parameter checker and syntax analyzer</td>
<td></td>
</tr>
<tr>
<td>DFHPASYA</td>
<td>DSECT</td>
<td>PASY parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHPASY</td>
<td>Macro</td>
<td>PASY request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHPASYT</td>
<td>CSECT</td>
<td>(OCO) PASY trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHPATCH</td>
<td>Macro</td>
<td>Generate patch area</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHPATRI</td>
<td>CSECT</td>
<td>(OCO) Trace interpreter for PA domain</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHPBP</td>
<td>CSECT</td>
<td>BMS page and text build</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHPBPAG</td>
<td>CSECT</td>
<td>BMS page and text build (standard)</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHPBPAG1G</td>
<td>CSECT</td>
<td>BMS page and text build (full)</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHPC</td>
<td>Macro</td>
<td>Program service request</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHPCEDS</td>
<td>DSECT</td>
<td>EXEC argument list for Program Control</td>
<td>11 -</td>
</tr>
</tbody>
</table>
### CICS directory

**Table 123. CICS modules directory (continued)**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHPCEXT</td>
<td>CSECT</td>
<td>AP recovery point when called from kernel</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHPCOM</td>
<td>Macro</td>
<td>PEP communication area</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHPCOMD</td>
<td>DSECT</td>
<td>PEP communication area</td>
<td>- 08</td>
</tr>
<tr>
<td>DFHPPC2</td>
<td>CSECT</td>
<td>PCP interface to COBOL stub for OS/VS</td>
<td>08 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COBOL V1 R2.3 or R2.4 application programs</td>
<td></td>
</tr>
<tr>
<td>DFHPCPG</td>
<td>CSECT</td>
<td>PM domain - interface program</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHPCCTD</td>
<td>DSECT</td>
<td>Program control table</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHPCFTF</td>
<td>Macro</td>
<td>Generate a profile entry</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHPCUE</td>
<td>DSECT</td>
<td>Program control data block for user exits</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHPCXDF</td>
<td>CSECT</td>
<td>DU domain - transaction dump formatter for program related areas</td>
<td>08 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DFHPEID</td>
<td>Macro</td>
<td>Generate BMS partition definition</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHPDKW</td>
<td>CSECT (OCO)</td>
<td>SDUMP formatting - CICSDATA operand</td>
<td>- 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DFHPEPF</td>
<td>CSECT</td>
<td>User-replaceable program error program</td>
<td>19 03</td>
</tr>
<tr>
<td>DFHPEP</td>
<td>Macro</td>
<td>Program error program - C/370</td>
<td>- 19</td>
</tr>
<tr>
<td>DFHPEP</td>
<td>Sample</td>
<td>Program environment save area (PESA)</td>
<td>08 -</td>
</tr>
<tr>
<td>DFHPGACD</td>
<td>Macro</td>
<td>PG domain - autoinstall exit program</td>
<td>11 -</td>
</tr>
<tr>
<td></td>
<td></td>
<td>parameter list - Assembler</td>
<td></td>
</tr>
<tr>
<td>DFHPGACH</td>
<td>CSECT</td>
<td>PG domain - autoinstall exit program</td>
<td>- 08</td>
</tr>
<tr>
<td></td>
<td></td>
<td>parameter list - C/370</td>
<td></td>
</tr>
<tr>
<td>DFHPGACL</td>
<td>CSECT</td>
<td>PG domain - autoinstall exit program</td>
<td>P2 -</td>
</tr>
<tr>
<td></td>
<td></td>
<td>parameter list - PL/I</td>
<td></td>
</tr>
<tr>
<td>DFHPGACO</td>
<td>CSECT</td>
<td>PG domain - autoinstall exit program</td>
<td>C2 -</td>
</tr>
<tr>
<td></td>
<td></td>
<td>parameter list - COBOL</td>
<td></td>
</tr>
<tr>
<td>DFHPGADS</td>
<td>DSECT</td>
<td>BMS page control area</td>
<td>08 -</td>
</tr>
<tr>
<td>DFHPGADX</td>
<td>CSECT</td>
<td>Program autoinstall exit - Assembler</td>
<td>19 03</td>
</tr>
<tr>
<td>DFHPGAHX</td>
<td>Sample</td>
<td>Program autoinstall exit - C/370</td>
<td>- 19</td>
</tr>
<tr>
<td>DFHPGAI</td>
<td>CSECT</td>
<td>Program autoinstall function</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHPGAIL</td>
<td>CSECT</td>
<td>PGAI trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHPGALX</td>
<td>Sample</td>
<td>Program autoinstall exit - PL/I</td>
<td>- 19</td>
</tr>
<tr>
<td>DFHPGAX</td>
<td>Sample</td>
<td>Program autoinstall exit - COBOL</td>
<td>- 19</td>
</tr>
<tr>
<td>DFHPGAQ</td>
<td>CSECT</td>
<td>PG domain - inquire/set autoinstall</td>
<td>- 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DFHPGAQA</td>
<td>DSECT</td>
<td>PGAQ parameter list</td>
<td>08 -</td>
</tr>
<tr>
<td>DFHPGAQM</td>
<td>Macro</td>
<td>PGAQ request</td>
<td>08 -</td>
</tr>
<tr>
<td>DFHPGAQT</td>
<td>CSECT</td>
<td>PGAQ trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHPGAQY</td>
<td>Macro</td>
<td>PGAQ request</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHPGAX</td>
<td>DSECT</td>
<td>PGAQ parameter list</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHPGCD</td>
<td>Source</td>
<td>PG domain anchor block</td>
<td>08 -</td>
</tr>
<tr>
<td>DFHPGDD</td>
<td>CSECT (OCO)</td>
<td>PG domain - define/delete program</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHPGDDA</td>
<td>DSECT</td>
<td>PGDD parameter list</td>
<td>08 -</td>
</tr>
<tr>
<td>DFHPGDDM</td>
<td>Macro</td>
<td>PGDD request</td>
<td>08 -</td>
</tr>
<tr>
<td>DFHPGDDT</td>
<td>CSECT (OCO)</td>
<td>PGDD trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHPGDM</td>
<td>CSECT</td>
<td>PG domain - initialize, quiesce, and terminate</td>
<td>- 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>terminate domain functions</td>
<td></td>
</tr>
<tr>
<td>DFHPGDUF</td>
<td>CSECT (OCO)</td>
<td>PG domain - SDUMP formatter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHPGEX</td>
<td>CSECT (OCO)</td>
<td>PG domain - initialize and terminate</td>
<td>- 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>exit functions</td>
<td></td>
</tr>
<tr>
<td>DFHPGEAX</td>
<td>DSECT</td>
<td>PGEX parameter list</td>
<td>08 -</td>
</tr>
<tr>
<td>DFHPGEAX</td>
<td>Macro</td>
<td>PGEX inline version of DFHPGEX</td>
<td>08 -</td>
</tr>
<tr>
<td>DFHPGEAX</td>
<td>Macro</td>
<td>PGEX request</td>
<td>08 -</td>
</tr>
<tr>
<td>DFHPGEAX</td>
<td>Macro (OCO)</td>
<td>PGEX trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHPGDS</td>
<td>Macro</td>
<td>PG domain - statistics</td>
<td>11 -</td>
</tr>
</tbody>
</table>
### Table 123. CICS modules directory (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHPGGDS</td>
<td>Macro</td>
<td>PG domain - statistics</td>
<td>C2 07</td>
</tr>
<tr>
<td>DFHPGHM</td>
<td>CSECT (OCO)</td>
<td>PG domain - handle manager services</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHPGHMA</td>
<td>DSECT</td>
<td>PGHM parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHPGHMM</td>
<td>Macro</td>
<td>PGHM request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHPGHMT</td>
<td>CSECT (OCO)</td>
<td>PGHM trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHPGIS</td>
<td>CSECT (OCO)</td>
<td>PG domain - PGIS functions</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHPGISA</td>
<td>DSECT</td>
<td>PGIS parameter list</td>
<td>- 11</td>
</tr>
<tr>
<td>DFHPGISI</td>
<td>Macro</td>
<td>PGIS inline version of DFHPGHMM</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHPGISM</td>
<td>Macro</td>
<td>PGIS request</td>
<td>- 11</td>
</tr>
<tr>
<td>DFHPGISST</td>
<td>CSECT (OCO)</td>
<td>PGIS trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHPGISX</td>
<td>Macro</td>
<td>PGIS request</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHPGGLD</td>
<td>CSECT (OCO)</td>
<td>PG domain - load and release functions</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHPGLDA</td>
<td>DSECT</td>
<td>PGLD parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHPGLDM</td>
<td>Macro</td>
<td>PGLD request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHPGLDT</td>
<td>CSECT (OCO)</td>
<td>PGLD trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHPGLE</td>
<td>CSECT (OCO)</td>
<td>PG domain - link exec function</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHPGLEA</td>
<td>DSECT</td>
<td>PGLE parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHPGLEM</td>
<td>Macro</td>
<td>PGLE request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHPGLET</td>
<td>CSECT (OCO)</td>
<td>PGLE trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHPGLK</td>
<td>CSECT (OCO)</td>
<td>PG domain - link and link PLT functions</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHPGLKA</td>
<td>DSECT</td>
<td>PGLK parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHPGLKMK</td>
<td>Macro</td>
<td>PGLK request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHPGLKXT</td>
<td>CSECT (OCO)</td>
<td>PGLK trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHPGLU</td>
<td>CSECT (OCO)</td>
<td>PG domain - link URM function</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHPGLUA</td>
<td>DSECT</td>
<td>PGLU parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHPGLUM</td>
<td>Macro</td>
<td>PGLU request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHPGLUT</td>
<td>CSECT (OCO)</td>
<td>PGLU trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHPGP</td>
<td>Macro</td>
<td>Validate group name for PCT/PPT migrate</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHPGPB</td>
<td>CSECT (OCO)</td>
<td>PG domain - initial link function</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHPGPG</td>
<td>DSECT</td>
<td>PGPG parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHPGPGM</td>
<td>Macro</td>
<td>PGPG request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHPGPXT</td>
<td>CSECT (OCO)</td>
<td>PGPG trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHPGREG</td>
<td>CSECT (OCO)</td>
<td>PGRE trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHPGREGA</td>
<td>DSECT</td>
<td>PGRE parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHPGREGM</td>
<td>Macro</td>
<td>PGRE request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHPGRET</td>
<td>CSECT (OCO)</td>
<td>PGRE trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHPGPGB</td>
<td>CSECT (OCO)</td>
<td>PG domain - recovery program</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHPGPBPT</td>
<td>CSECT (OCO)</td>
<td>PGPB trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHPGST</td>
<td>CSECT (OCO)</td>
<td>PG domain - statistics</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHPGTRI</td>
<td>CSECT (OCO)</td>
<td>PG domain - trace interpreter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHPGUE</td>
<td>CSECT (OCO)</td>
<td>PG domain - service requests user exit</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHPGXE</td>
<td>CSECT (OCO)</td>
<td>PG domain - prepare XCTL function</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHPGXEIA</td>
<td>DSECT</td>
<td>PGXE parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHPGXEIA</td>
<td>Macro</td>
<td>PGXE request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHPGXET</td>
<td>CSECT (OCO)</td>
<td>PGXE trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHPGXEM</td>
<td>CSECT (OCO)</td>
<td>PG domain - initialize and terminate transactions</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHPGXEM</td>
<td>DSECT</td>
<td>PGXEM parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHPGXEM</td>
<td>Macro</td>
<td>PGXEM request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHPGXMT</td>
<td>CSECT (OCO)</td>
<td>PGXM trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHPH</td>
<td>Macro</td>
<td>Partition handling macro</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHPHBN</td>
<td>CSECT</td>
<td>Phonetic code conversion</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHPHP</td>
<td>CSECT</td>
<td>Partition handling program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHPLARG</td>
<td>DSECT</td>
<td>Generalized domain call parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(header, standard fields, responses)</td>
<td></td>
</tr>
<tr>
<td>DFHPPLT</td>
<td>Macro</td>
<td>Program list table</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHPPLTD</td>
<td>DSECT</td>
<td>Program list table definition</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHPPFS</td>
<td>DSECT</td>
<td>KC domain - profile data</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHPFRM</td>
<td>CSECT (OCO)</td>
<td>Partner resource manager command interface</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHPFRMMA</td>
<td>DSECT</td>
<td>PRCM parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHPFRMNN</td>
<td>Macro</td>
<td>PRCM request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHPFRMT</td>
<td>CSECT (OCO)</td>
<td>PRCM trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHPRDFU</td>
<td>CSECT (OCO)</td>
<td>Partner resource manager SDUMP formatter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHPRFS</td>
<td>CSECT (OCO)</td>
<td>Partner resource manager interface to SAA</td>
<td>- 03</td>
</tr>
</tbody>
</table>
### CICS directory

Table 123. CICS modules directory (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHPRFSA</td>
<td>DSECT</td>
<td>PRFS parameter list</td>
<td>OS</td>
</tr>
<tr>
<td>DFHPRFST</td>
<td>CSECT (OCO)</td>
<td>PRFS trace interpretation data</td>
<td>-</td>
</tr>
<tr>
<td>DFHPRINA</td>
<td>DSECT</td>
<td>PRIN parameter list</td>
<td>OS</td>
</tr>
<tr>
<td>DFHPRINM</td>
<td>Macro</td>
<td>PRIN request</td>
<td>OS</td>
</tr>
<tr>
<td>DFHPRINT</td>
<td>Macro</td>
<td>DSECT print control</td>
<td>11</td>
</tr>
<tr>
<td>DFHPRINU</td>
<td>CSECT (OCO)</td>
<td>PRIN trace interpretation data</td>
<td>-</td>
</tr>
<tr>
<td>DFHPRIN1</td>
<td>CSECT (OCO)</td>
<td>Partner resource manager initialization management program</td>
<td>-</td>
</tr>
<tr>
<td>DFHPRIN2</td>
<td>CSECT (OCO)</td>
<td>Partner resource manager initialization subtask program</td>
<td>-</td>
</tr>
<tr>
<td>DFHPRK</td>
<td>CSECT</td>
<td>3270 print key program</td>
<td>OS, 03</td>
</tr>
<tr>
<td>DFHPRMCK</td>
<td>Macro</td>
<td>Parameter checking macro</td>
<td>11</td>
</tr>
<tr>
<td>DFHPROLG</td>
<td>Source</td>
<td>Prologue to DFHENTER</td>
<td>OS</td>
</tr>
<tr>
<td>DFHPROLM</td>
<td>Source</td>
<td>Acquire LIFO storage application prolog</td>
<td>OS</td>
</tr>
<tr>
<td>DFHPRPT</td>
<td>CSECT (OCO)</td>
<td>Partner resource table (PRT) manager</td>
<td>-</td>
</tr>
<tr>
<td>DFHPRPTA</td>
<td>DSECT</td>
<td>PRPT parameter list</td>
<td>OS</td>
</tr>
<tr>
<td>DFHPRPTM</td>
<td>Macro</td>
<td>PRPT request</td>
<td>OS</td>
</tr>
<tr>
<td>DFHPRPPTT</td>
<td>CSECT (OCO)</td>
<td>PRPT trace interpretation data</td>
<td>-</td>
</tr>
<tr>
<td>DFHPRPP</td>
<td>CSECT (OCO)</td>
<td>Partner resource manager recovery program</td>
<td>-</td>
</tr>
<tr>
<td>DFHPRRPA</td>
<td>DSECT</td>
<td>PRRP parameter list</td>
<td>OS</td>
</tr>
<tr>
<td>DFHPRRPM</td>
<td>Macro</td>
<td>PRRP request</td>
<td>OS</td>
</tr>
<tr>
<td>DFHPRRPTT</td>
<td>CSECT (OCO)</td>
<td>PRRP trace interpretation data</td>
<td>-</td>
</tr>
<tr>
<td>DFHPRSDS</td>
<td>DSECT</td>
<td>Partner static storage area</td>
<td>OS</td>
</tr>
<tr>
<td>DFHPS</td>
<td>Macro</td>
<td>System spooling interface</td>
<td>OS</td>
</tr>
<tr>
<td>DFHPSD</td>
<td>Macro</td>
<td>Generate BMS partition set definition</td>
<td>11</td>
</tr>
<tr>
<td>DFHPSDDS</td>
<td>DSECT</td>
<td>Partition set control block</td>
<td>OS</td>
</tr>
<tr>
<td>DFHPSGDS</td>
<td>DSECT</td>
<td>Spooler global control block</td>
<td>11</td>
</tr>
<tr>
<td>DFHPSIP</td>
<td>CSECT</td>
<td>Spooler initialization program</td>
<td>OS, 03</td>
</tr>
<tr>
<td>DFHPSP</td>
<td>CSECT</td>
<td>System spooling interface</td>
<td>OS, 03</td>
</tr>
<tr>
<td>DFHPSCK</td>
<td>CSECT</td>
<td>System spooling subsystem activator</td>
<td>OS, 03</td>
</tr>
<tr>
<td>DFHPSPDW</td>
<td>CSECT</td>
<td>System spooling interface, DWE processor</td>
<td>OS, 03</td>
</tr>
<tr>
<td>DFHPSPS</td>
<td>CSECT</td>
<td>System spooling JES interface subtask</td>
<td>OS, 03</td>
</tr>
<tr>
<td>DFHPSPST</td>
<td>CSECT</td>
<td>System spooling JES interface control</td>
<td>OS, 03</td>
</tr>
<tr>
<td>DFHPSVC</td>
<td>CSECT</td>
<td>System spooling interface, retrieve a data set name</td>
<td>OS, 03</td>
</tr>
<tr>
<td>DFHPTDF</td>
<td>CSECT (OCO)</td>
<td>Program control table SDUMP formatter</td>
<td>-</td>
</tr>
<tr>
<td>DFHPUKAB</td>
<td>CSECT</td>
<td>CSUDP - initialize RDO parameter fields and address list (DFHUPA)</td>
<td>OS, 03</td>
</tr>
<tr>
<td>DFHPUKAC</td>
<td>CSECT</td>
<td>CSUDP - initialize RDO parameter fields and address list (DFHUPA)</td>
<td>OS, 03</td>
</tr>
<tr>
<td>DFHPUKB</td>
<td>CSECT</td>
<td>CSUDP - RDO parameter utility program, batch environment (DFHUPU batch)</td>
<td>OS, 03</td>
</tr>
<tr>
<td>DFHPUPC</td>
<td>CSECT</td>
<td>RDO parameter utility program, CICS environment (DFHPUPE CICS)</td>
<td>OS, 03</td>
</tr>
<tr>
<td>DFHPUPOB</td>
<td>CSECT</td>
<td>CSUDP - default parameter values lookup (DFHUPPDP batch)</td>
<td>OS, 03</td>
</tr>
<tr>
<td>DFHPUPODC</td>
<td>CSECT</td>
<td>RDO parameter utility - default parameter values lookup (DFHUPDPC CICS)</td>
<td>OS, 03</td>
</tr>
<tr>
<td>DFHPUPOXB</td>
<td>CSECT</td>
<td>CSUDP - language table referencing functions (DFHUPDX batch)</td>
<td>OS, 03</td>
</tr>
<tr>
<td>DFHUPUXC</td>
<td>CSECT</td>
<td>RDO parameter utility - language table referencing functions (DFHUPUX CICS)</td>
<td>OS, 03</td>
</tr>
<tr>
<td>DFHPS3270</td>
<td>CSECT</td>
<td>3270 print function support</td>
<td>OS, 03</td>
</tr>
<tr>
<td>DFHMDY</td>
<td>CSECT</td>
<td>Query transaction</td>
<td>OS, 03</td>
</tr>
<tr>
<td>DFHSSS</td>
<td>CSECT (OCO)</td>
<td>Qualified subsystem services</td>
<td>-</td>
</tr>
<tr>
<td>DFHRCX</td>
<td>CSECT</td>
<td>Recovery control enable exit</td>
<td>OS, 03</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
<td>Library</td>
</tr>
<tr>
<td>------------</td>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>DFHRCNO</td>
<td>Other</td>
<td>Used by DFHSTART cataloged procedure</td>
<td>19 -</td>
</tr>
<tr>
<td>DFHRCSDS</td>
<td>DSECT</td>
<td>Recovery control static storage</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHRCYES</td>
<td>Other</td>
<td>Used by DFHSTART cataloged procedure</td>
<td>19 -</td>
</tr>
<tr>
<td>DFHRDDUF</td>
<td>CSECT</td>
<td>Resource definition recovery offline dump exit</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRDJPN</td>
<td>CSECT (OCO)</td>
<td>CSO utilities - RDL for Japanese language feature upgrade</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHREGS</td>
<td>Macro</td>
<td>Standard register name definition</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHREQ</td>
<td>Macro</td>
<td>Attention ID coding macro</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHREST</td>
<td>CSECT</td>
<td>User-replaceable restart program</td>
<td>19 03</td>
</tr>
<tr>
<td>DFHRITRI</td>
<td>CSECT</td>
<td>RMI trace interpretation routine</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRKB</td>
<td>CSECT</td>
<td>3270 release keyboard program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHRLR</td>
<td>CSECT</td>
<td>BMS route list resolution</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHRLRAG</td>
<td>CSECT</td>
<td>BMS route list resolution (standard)</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHRLRIG</td>
<td>CSECT</td>
<td>BMS route list resolution (full)</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHRMCAL</td>
<td>Macro</td>
<td>Resource manager call</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHRMCD</td>
<td>CSECT</td>
<td>Recovery manager client directory</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMCD0</td>
<td>CSECT</td>
<td>RMCD parameter list</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMCD1</td>
<td>Macro</td>
<td>RMCD request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHRMCDT</td>
<td>DSECT</td>
<td>RMCD translate tables</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMCD1</td>
<td>CSECT</td>
<td>RM client directory class initialization</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMCD2</td>
<td>CSECT</td>
<td>RM client directory class quiesce proc</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMCD12</td>
<td>CSECT</td>
<td>RM client directory set gate procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMCD3</td>
<td>CSECT</td>
<td>RM client directory wait for client proc</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMCD4</td>
<td>CSECT</td>
<td>RM client directory send procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMDEA</td>
<td>CSECT</td>
<td>RMDE parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHRMDEM</td>
<td>Macro</td>
<td>RMDE request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHRMDET</td>
<td>DSECT</td>
<td>RMDE translate tables</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMDDM</td>
<td>CSECT</td>
<td>Recovery manager domain management</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMDMA</td>
<td>CSECT</td>
<td>RMDD parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHRMDDM</td>
<td>Macro</td>
<td>RMDD request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHRMDT</td>
<td>DSECT</td>
<td>RMDD translate tables</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMDU0</td>
<td>CSECT</td>
<td>RMCI dump formatting</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMDU1</td>
<td>CSECT</td>
<td>RMDU start work token browse procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMDU3</td>
<td>CSECT</td>
<td>RMDU get next work token procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMDU4</td>
<td>CSECT</td>
<td>RMDU end work token browse procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMDU5</td>
<td>CSECT</td>
<td>- 03</td>
<td></td>
</tr>
<tr>
<td>DFHRMGDS</td>
<td>CSECT</td>
<td>Recovery manager global statistics</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHRMGDS</td>
<td>CSECT</td>
<td>Recovery manager global statistics</td>
<td>C2 07</td>
</tr>
<tr>
<td>DFHRMKA0</td>
<td>CSECT</td>
<td>RMKD parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHRMKA1</td>
<td>Macro</td>
<td>RMKD request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHRMKT</td>
<td>DSECT</td>
<td>RMKD translate tables</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMKPA</td>
<td>CSECT</td>
<td>RM KP parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHRMKPM</td>
<td>Macro</td>
<td>RM KP request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHRMKPT</td>
<td>DSECT</td>
<td>RM KP translate tables</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMLKQ</td>
<td>CSECT</td>
<td>RMLK quiesce procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMLKT</td>
<td>DSECT</td>
<td>RMLK translate tables</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMLK1</td>
<td>CSECT</td>
<td>RMLK initialize class procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMLK2</td>
<td>CSECT</td>
<td>RMLK initiate recovery2 procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMLK3</td>
<td>CSECT</td>
<td>RMLK inquire logname procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMLK4</td>
<td>CSECT</td>
<td>RMLK clear pending2 procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMLK5</td>
<td>CSECT</td>
<td>RMLK collect statistics procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMLN</td>
<td>CSECT</td>
<td>RMLN gate handler module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMLNA</td>
<td>CSECT</td>
<td>RMLN parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHRMLNM</td>
<td>Macro</td>
<td>RMLN request</td>
<td>OS -</td>
</tr>
</tbody>
</table>
### CICS directory

**Table 123. CICS modules directory (continued)**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHMLNT</td>
<td>DSECT</td>
<td>RMLN translate table</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMSD</td>
<td>CSECT</td>
<td>Recovery Manager LinkSet class declaration</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMLSF</td>
<td>CSECT</td>
<td>RMLS inquire awaiting forget procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMLSO</td>
<td>CSECT</td>
<td>RMLS commit procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMLSP</td>
<td>CSECT</td>
<td>RMLS prepare procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMLSS</td>
<td>CSECT</td>
<td>RMLS shunt procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMLSU</td>
<td>CSECT</td>
<td>RMLS unshunt procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMLSD</td>
<td>CSECT</td>
<td>Recovery Manager LinkSet class declaration</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMLSF</td>
<td>CSECT</td>
<td>RMLS inquire awaiting forget procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMLSO</td>
<td>CSECT</td>
<td>RMLS commit procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMLSP</td>
<td>CSECT</td>
<td>RMLS prepare procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMLSS</td>
<td>CSECT</td>
<td>RMLS shunt procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMLSU</td>
<td>CSECT</td>
<td>RMLS unshunt procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMLSD</td>
<td>CSECT</td>
<td>Recovery Manager Lognames class</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMNMA</td>
<td>CSECT</td>
<td>RMMN parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHRMNMM</td>
<td>Macro</td>
<td>RMMN translate tables</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHRMNMT</td>
<td>DSECT</td>
<td>RMMN translate tables</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMNMI</td>
<td>CSECT</td>
<td>RMMN initialize class procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMNS1</td>
<td>CSECT</td>
<td>RMNS initialize class procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMNS2</td>
<td>CSECT</td>
<td>RMNS quiesce procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMDF1</td>
<td>CSECT</td>
<td>RMDF initialize procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMOT</td>
<td>CSECT</td>
<td>RMOT CDURUN and Gate Module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMOTT</td>
<td>CSECT</td>
<td>RMOT CDURUN and Gate Module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMREA</td>
<td>CSECT</td>
<td>RMRE parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHRMEM</td>
<td>Macro</td>
<td>RMRE request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHRMRT</td>
<td>DSECT</td>
<td>RMRE translate tables</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMR0</td>
<td>CSECT</td>
<td>RM resource owner class</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMROA</td>
<td>CSECT</td>
<td>RMRO parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHRMROM</td>
<td>Macro</td>
<td>RMRO request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHRMROO</td>
<td>CSECT</td>
<td>RMRO forgotten procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMROS</td>
<td>CSECT</td>
<td>RMRO shunt procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMROT</td>
<td>CSECT</td>
<td>RMRO translate tables</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMROU</td>
<td>CSECT</td>
<td>RMRO unshunt procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMROV</td>
<td>CSECT</td>
<td>RMRO avail procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMR01</td>
<td>CSECT</td>
<td>RMRO initialize class procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMR02</td>
<td>CSECT</td>
<td>RMRO start back out procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMR03</td>
<td>CSECT</td>
<td>RMRO deliver back out data procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMR04</td>
<td>CSECT</td>
<td>RMRO end back out procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMR05</td>
<td>CSECT</td>
<td>RM RMCDURUN and Gate Module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMR1D</td>
<td>CSECT</td>
<td>RMRO deliver data procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMR1E</td>
<td>CSECT</td>
<td>RMRO end delivery procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMR1K</td>
<td>CSECT</td>
<td>RMRO take keypoint procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMR1S</td>
<td>CSECT</td>
<td>RMRO start delivery procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMRSL</td>
<td>CSECT</td>
<td>RM system log class</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTRSLA</td>
<td>CSECT</td>
<td>RMSL parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHTRSLF</td>
<td>CSECT</td>
<td>RMSL force procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTRSLJ</td>
<td>CSECT</td>
<td>RMSL notify disjoint chains procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTRSLL</td>
<td>CSECT</td>
<td>RMSL close chain procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTRSLM</td>
<td>Macro</td>
<td>RMSL request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHTRSLO</td>
<td>CSECT</td>
<td>RMSL open chain procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTRSLT</td>
<td>CSECT</td>
<td>RMSL translate tables</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTRSLV</td>
<td>CSECT</td>
<td>RMSL move chain procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTRSLW</td>
<td>CSECT</td>
<td>RMSL write procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTRSLI</td>
<td>CSECT</td>
<td>RMSL initialize class procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTRSL2</td>
<td>CSECT</td>
<td>RMSL start chain browse procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTRSL3</td>
<td>CSECT</td>
<td>RMSL chain browse read procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTRSL4</td>
<td>CSECT</td>
<td>RMSL end chain browse procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTRSL5</td>
<td>CSECT</td>
<td>RMSL restart procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTRSL6</td>
<td>CSECT</td>
<td>RMSL schedule keypoint procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTRSL7</td>
<td>CSECT</td>
<td>RMSL take keypoint procedure</td>
<td>- 03</td>
</tr>
</tbody>
</table>
### Table 123. CICS modules directory (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHRMST</td>
<td>CSECT</td>
<td>RM statistics class</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMST1</td>
<td>CSECT</td>
<td>RMST initialize class procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMSTY</td>
<td>CSECT</td>
<td>Resource Manager resynchronization program</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMSTRI</td>
<td>CSECT</td>
<td>Offline trace formatting interpretation routine</td>
<td>- 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>parameter list</td>
<td></td>
</tr>
<tr>
<td>DFHRMUC</td>
<td>CSECT</td>
<td>Resource Manager create UOW</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMUD0</td>
<td>CSECT</td>
<td>Resource Manager commit UOW</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMUD1</td>
<td>CSECT</td>
<td>Resource Manager unit of work class</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMUDTL</td>
<td>CSECT</td>
<td>Resource Manager batch utility program</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMUWA</td>
<td>CSECT</td>
<td>RMUW parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHRMUWB</td>
<td>CSECT</td>
<td>RMUW deliver backout procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMUWE</td>
<td>CSECT</td>
<td>RMUW unshunt reply procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMUWF</td>
<td>CSECT</td>
<td>RMUW force procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMUMH</td>
<td>CSECT</td>
<td>RMUW hold procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMUW1</td>
<td>Macro</td>
<td>RMUW inquire UOQ ID</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHRMUW3</td>
<td>CSECT</td>
<td>RMUW force heurism procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMUL</td>
<td>CSECT</td>
<td>RMUW forget links procedure</td>
<td>- 05</td>
</tr>
<tr>
<td>DFHRMUNW</td>
<td>Macro</td>
<td>RMUW request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHRMUNW</td>
<td>CSECT</td>
<td>RMUW unshunt procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMUWP</td>
<td>CSECT</td>
<td>RMUW process avail procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMUWQ</td>
<td>CSECT</td>
<td>RMUW process indoubt resolution procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMUWS</td>
<td>CSECT</td>
<td>RMUW record decision procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMUWT</td>
<td>DSECT</td>
<td>RM unit of work class (timeout)</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMUWU</td>
<td>CSECT</td>
<td>RMUW set local lu name procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMUWV</td>
<td>CSECT</td>
<td>RMUW avail procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMUW</td>
<td>CSECT</td>
<td>RMUW write procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMUWO</td>
<td>CSECT</td>
<td>RMUW release procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMWW</td>
<td>CSECT</td>
<td>RMUW initialize class procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMWW2</td>
<td>CSECT</td>
<td>RMUW collect statistics procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMWW3</td>
<td>CSECT</td>
<td>RMUW inquire work token procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMWWW</td>
<td>DSECT</td>
<td>Define parts of UOW objects accessible by OS</td>
<td>- 05</td>
</tr>
<tr>
<td></td>
<td></td>
<td>inline macros</td>
<td></td>
</tr>
<tr>
<td>DFHRMUWIC</td>
<td>CSECT</td>
<td>RMUW set chain token procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMUWJD</td>
<td>CSECT</td>
<td>RMUW deliver data procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMUWJE</td>
<td>CSECT</td>
<td>RMUW end delivery procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMUWIF</td>
<td>CSECT</td>
<td>RMUW wait timeout notify procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMUWJG</td>
<td>CSECT</td>
<td>RMUW</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMUWJH</td>
<td>CSECT</td>
<td>RMUW inquire disjoint chains procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMUWJK</td>
<td>CSECT</td>
<td>RMUW take keypoint procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMUL</td>
<td>CSECT</td>
<td>XMPP force purge inhibit query gate</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMUWL</td>
<td>CSECT</td>
<td>RMU1 force purge query procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMUQ</td>
<td>CSECT</td>
<td>TSR notify gate</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMUQ</td>
<td>CSECT</td>
<td>RMUW restart procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMUQL</td>
<td>CSECT</td>
<td>RMUW start delivery procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMUQUL</td>
<td>CSECT</td>
<td>RMUW process restart procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMUQIV</td>
<td>CSECT</td>
<td>RMUW request wait timeout procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMUQIW</td>
<td>CSECT</td>
<td>RMUW cancel wait timeout procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMVPI</td>
<td>CSECT</td>
<td>RMVP initialize class procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMTA</td>
<td>CSECT</td>
<td>RMNT parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHRMWTA</td>
<td>Macro</td>
<td>Supports the Inquire_work_token and Set_work_token of RMNT CDURUN interface</td>
<td>OS -</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RMNT request</td>
<td></td>
</tr>
<tr>
<td>DFHRMWT</td>
<td>DSECT</td>
<td>RMNT translate tables</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMXNE</td>
<td>CSECT</td>
<td>RMXN reattach procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMXN2</td>
<td>CSECT</td>
<td>RMXN schedule keypoint procedure</td>
<td>- 03</td>
</tr>
</tbody>
</table>
### CICS directory

**Table 123. CICS modules directory (continued)**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHRMNX3</td>
<td>CSECT</td>
<td>RMNX keypoint transaction</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMNX4</td>
<td>CSECT</td>
<td>RMNX restart procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRMNX5</td>
<td>CSECT</td>
<td>RMNX inc trandef statistic procedure</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHROIINA</td>
<td>CSECT</td>
<td>ROIN parameter list</td>
<td>0S -</td>
</tr>
<tr>
<td>DFHROIINM</td>
<td>Macro</td>
<td>ROIN request</td>
<td>0S -</td>
</tr>
<tr>
<td>DFHROIINT</td>
<td>DSECT</td>
<td>ROIN translate tables</td>
<td>0S 03</td>
</tr>
<tr>
<td>DFHROPAL</td>
<td>CSECT</td>
<td>ONC RPC Feature alias list</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHROPALT</td>
<td>DSECT</td>
<td>RPAL translate tables</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHROAS</td>
<td>CSECT</td>
<td>(OCO) ONC RPC alias main program</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHROINT</td>
<td>DSECT</td>
<td>(OCO) RPCC parameter list</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRPCC</td>
<td>DSECT</td>
<td>(OCO) RPCC parameter list</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRPCB</td>
<td>Macro</td>
<td>Extension to DL/I PCB control block</td>
<td>0S -</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- contains ISC information about PCB</td>
<td></td>
</tr>
<tr>
<td>DFHRPCDH</td>
<td>CSECT</td>
<td>RPPC caller DFHRPC parameter list</td>
<td>- 08</td>
</tr>
<tr>
<td>DFHRPCDO</td>
<td>CSECT</td>
<td>RPPC caller DFHRPC parameter list</td>
<td>- 07</td>
</tr>
<tr>
<td>DFHRPCOA</td>
<td>CSECT</td>
<td>CRPC dataset list processing</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRPCOB</td>
<td>CSECT</td>
<td>CRPC common subroutines</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRPCOD</td>
<td>CSECT</td>
<td>CRPC register remote procedures</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRPCOE</td>
<td>CSECT</td>
<td>CRPC register remote procedures</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRPCO1</td>
<td>CSECT</td>
<td>CRPC initial processing</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRPCO1</td>
<td>CSECT</td>
<td>CRPC manage feature dataset</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRPCO2</td>
<td>CSECT</td>
<td>CRPC disable processing</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRPCO5</td>
<td>CSECT</td>
<td>CRPC manage feature dataset</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRPCO6</td>
<td>CSECT</td>
<td>CRPC update feature</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRPCO8</td>
<td>CSECT</td>
<td>CRPC ONC RPC feature</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRPC09</td>
<td>CSECT</td>
<td>CRPC registration table management</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRPC10</td>
<td>CSECT</td>
<td>CRPC alias list processing</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRPC14</td>
<td>CSECT</td>
<td>CRPC alias list processing</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRPC14</td>
<td>CSECT</td>
<td>CRPC RPC server controller</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHPRPC42</td>
<td>CSECT</td>
<td>CRPC enable request processing</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHPRDUF</td>
<td>CSECT</td>
<td>System dump formatting routine for ONC/RPC</td>
<td>0S 03</td>
</tr>
<tr>
<td>DFHPRMS</td>
<td>CSECT</td>
<td>ONC RPC feature server controller</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHPRPAR</td>
<td>CSECT</td>
<td>RPRSC parameter list</td>
<td>- 08</td>
</tr>
<tr>
<td>DFHPRPAR</td>
<td>CSECT</td>
<td>RPRSC parameter list</td>
<td>- 07</td>
</tr>
<tr>
<td>DFHPRPRA</td>
<td>CSECT</td>
<td>(OCO) ONC RPC feature server controller</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHPRPRA</td>
<td>CSECT</td>
<td>(OCO) ONC RPC feature RPC caller</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHPRPRA</td>
<td>CSECT</td>
<td>(OCO) RPPP call structured parameter list</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRPRT1</td>
<td>CSECT</td>
<td>ONC RPC feature trace interpretation</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRPRT1</td>
<td>CSECT</td>
<td>ONC RPC task-related user exit</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRPCO10</td>
<td>CSECT</td>
<td>Constants used by user replaceable programs</td>
<td>- 08</td>
</tr>
<tr>
<td>DFHRPCO10</td>
<td>CSECT</td>
<td>Constants used by user replaceable programs</td>
<td>- 07</td>
</tr>
<tr>
<td>DFHRP0</td>
<td>CSECT</td>
<td>BMS mapset for CRPC main panels</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRP0H</td>
<td>CSECT</td>
<td>CRPC DFHRP0 help panels</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRST</td>
<td>Macro</td>
<td>DBCTL XRF recoverable service table</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHRTC</td>
<td>CSECT</td>
<td>CRTE cancel command processor</td>
<td>0S 03</td>
</tr>
<tr>
<td>DFHRTTE</td>
<td>CSECT</td>
<td>Transaction routing program</td>
<td>0S 03</td>
</tr>
<tr>
<td>DFHRTSU</td>
<td>CSECT</td>
<td>Surrogate terminal interface program</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRTSUA</td>
<td>CSECT</td>
<td>RTSU parameter list</td>
<td>0S -</td>
</tr>
<tr>
<td>DFHRTSUI</td>
<td>CSECT</td>
<td>Provide Assign/Relay relay link functions</td>
<td>0S -</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- contains ISC information about PCB</td>
<td></td>
</tr>
<tr>
<td>DFHRTSUI</td>
<td>CSECT</td>
<td>RTSU request</td>
<td>0S -</td>
</tr>
<tr>
<td>DFHRTSUT</td>
<td>DSECT</td>
<td>RTSU translate tables</td>
<td>0S 03</td>
</tr>
<tr>
<td>DFHRTTRE</td>
<td>CSECT</td>
<td>ISC transaction routing (APRT) trace interpreter</td>
<td>0S 03</td>
</tr>
<tr>
<td>DFHRTTRE</td>
<td>CSECT</td>
<td>RTSU trace interpretation</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHRXAST</td>
<td>CSECT</td>
<td>RX Domain Management</td>
<td>03 -</td>
</tr>
<tr>
<td>DFHRXDMA</td>
<td>CSECT</td>
<td>RXDM interface parameter area</td>
<td>0S -</td>
</tr>
</tbody>
</table>
### Table 123. CICS modules directory (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHRXDM Macro</td>
<td>DFHRXDM interface macro</td>
<td>OS -</td>
<td></td>
</tr>
<tr>
<td>DFHRXDMT CSECT</td>
<td>-</td>
<td>RFHRXDM interface macro</td>
<td>03</td>
</tr>
<tr>
<td>DFHRXDUF CSECT</td>
<td>CSECT</td>
<td>RX Domain Dump Formatting</td>
<td>03</td>
</tr>
<tr>
<td>DFHRXSVC CSECT</td>
<td>CSECT</td>
<td>RX Domain Management</td>
<td>03</td>
</tr>
<tr>
<td>DFHRXTRI CSECT</td>
<td>DFHRXTRI Design</td>
<td>-</td>
<td>03</td>
</tr>
<tr>
<td>DFHRXUXW CSECT</td>
<td>CSECT</td>
<td>RX Domain UOW Manager</td>
<td>03</td>
</tr>
<tr>
<td>DFHRXUXWA Other</td>
<td>Other</td>
<td>RXUW interface parameter area</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHRXUXWM Macro</td>
<td>DFHRXUXWM interface macro</td>
<td>OS -</td>
<td></td>
</tr>
<tr>
<td>DFHRXUXWT CSECT</td>
<td>-</td>
<td>-</td>
<td>03</td>
</tr>
<tr>
<td>DFHRXUXMA Other</td>
<td>Other</td>
<td>RXXM interface parameter area</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHRXUXMM Macro</td>
<td>DFHRXUXMM interface macro</td>
<td>OS -</td>
<td></td>
</tr>
<tr>
<td>DFHRXUXMT CSECT</td>
<td>-</td>
<td>-</td>
<td>03</td>
</tr>
<tr>
<td>DFHRXUXRG CSECT</td>
<td>-</td>
<td>-</td>
<td>03</td>
</tr>
<tr>
<td>DFHRXUXRM CSECT</td>
<td>-</td>
<td>-</td>
<td>03</td>
</tr>
<tr>
<td>DFHRZDM CSECT</td>
<td>-</td>
<td>-</td>
<td>03</td>
</tr>
<tr>
<td>DFHRZDUF CSECT</td>
<td>RequestStreams remote join interface</td>
<td>-</td>
<td>03</td>
</tr>
<tr>
<td>DFHRZIX CSECT</td>
<td>-</td>
<td>-</td>
<td>03</td>
</tr>
<tr>
<td>DFHRZJN CSECT</td>
<td>-</td>
<td>-</td>
<td>03</td>
</tr>
<tr>
<td>DFHRZLN CSECT</td>
<td>-</td>
<td>-</td>
<td>03</td>
</tr>
<tr>
<td>DFHRZJR2 CSECT</td>
<td>-</td>
<td>-</td>
<td>03</td>
</tr>
<tr>
<td>DFHRZFOF CSECT</td>
<td>-</td>
<td>-</td>
<td>03</td>
</tr>
<tr>
<td>DFHRZRG2 CSECT</td>
<td>-</td>
<td>-</td>
<td>03</td>
</tr>
<tr>
<td>DFHRZKJ CSECT</td>
<td>RequestStreams remote join interface</td>
<td>-</td>
<td>03</td>
</tr>
<tr>
<td>DFHRZKJT CSECT</td>
<td>-</td>
<td>-</td>
<td>03</td>
</tr>
<tr>
<td>DFHRZRM CSECT</td>
<td>RZRM Gate Module for RM RO callback</td>
<td>- 03</td>
<td></td>
</tr>
<tr>
<td>DFHRZRS1 CSECT</td>
<td>-</td>
<td>-</td>
<td>03</td>
</tr>
<tr>
<td>DFHRZRT CSECT</td>
<td>RZRT CDURUN and Gate Module</td>
<td>- 03</td>
<td></td>
</tr>
<tr>
<td>DFHRZRT1 CSECT</td>
<td>-</td>
<td>-</td>
<td>03</td>
</tr>
<tr>
<td>DFHRZRT2 CSECT</td>
<td>-</td>
<td>-</td>
<td>03</td>
</tr>
<tr>
<td>DFHRZSO CSECT</td>
<td>-</td>
<td>-</td>
<td>03</td>
</tr>
<tr>
<td>DFHRZSOT CSECT</td>
<td>-</td>
<td>-</td>
<td>03</td>
</tr>
<tr>
<td>DFHRZSOI CSECT</td>
<td>-</td>
<td>-</td>
<td>03</td>
</tr>
<tr>
<td>DFHRZTA CSECT</td>
<td>-</td>
<td>-</td>
<td>03</td>
</tr>
<tr>
<td>DFHRZTAT CSECT</td>
<td>-</td>
<td>-</td>
<td>03</td>
</tr>
<tr>
<td>DFHRZTXC CSECT</td>
<td>-</td>
<td>-</td>
<td>03</td>
</tr>
<tr>
<td>DFHRZTRI CSECT</td>
<td>RequestStreams Trace interpretation</td>
<td>- 03</td>
<td></td>
</tr>
<tr>
<td>DFHRZTR1 CSECT</td>
<td>-</td>
<td>-</td>
<td>03</td>
</tr>
<tr>
<td>DFHRZVP1 CSECT</td>
<td>-</td>
<td>-</td>
<td>03</td>
</tr>
<tr>
<td>DFHRZXM CSECT</td>
<td>RequestStreams XM Attach Client</td>
<td>- 03</td>
<td></td>
</tr>
<tr>
<td>DFHRZTR2 CSECT</td>
<td>-</td>
<td>-</td>
<td>03</td>
</tr>
<tr>
<td>DFHSAADS DSECT</td>
<td>Storage accounting area</td>
<td>11 -</td>
<td></td>
</tr>
<tr>
<td>DFHSAOBS DSECT</td>
<td>Subsystem anchor block</td>
<td>OS -</td>
<td></td>
</tr>
<tr>
<td>DFHSAIQ CSECT (OCO)</td>
<td>AP domain - system data inquire and set</td>
<td>- 03</td>
<td></td>
</tr>
<tr>
<td>DFHSAIQT CSECT (OCO)</td>
<td>SAIQ trace interpretation data</td>
<td>- 03</td>
<td></td>
</tr>
<tr>
<td>DFHSAIQX Macro</td>
<td>SAIQ request</td>
<td>11 -</td>
<td></td>
</tr>
<tr>
<td>DFHSAIQY DSECT</td>
<td>SAIQ parameter list</td>
<td>11 -</td>
<td></td>
</tr>
<tr>
<td>DFHSAXDF CSECT</td>
<td>DU domain - transaction dump formatter for system areas (CSA, TCA, and so on)</td>
<td>03 03</td>
<td></td>
</tr>
<tr>
<td>DFHSC Macro</td>
<td>Storage service request</td>
<td>11 -</td>
<td></td>
</tr>
<tr>
<td>DFHSCA DSECT</td>
<td>Language Environment/370 - set common anchor area</td>
<td>OS 03</td>
<td></td>
</tr>
<tr>
<td>DFHSCALL Macro</td>
<td>EXEC interface call macro for CICSPlex SM commands in assembler-language pgms</td>
<td>11 -</td>
<td></td>
</tr>
<tr>
<td>DFHSCCSS Macro</td>
<td>Storage control class of storage</td>
<td>OS -</td>
<td></td>
</tr>
<tr>
<td>DFHSDGDS DSECT</td>
<td>System dump global statistics</td>
<td>11 -</td>
<td></td>
</tr>
</tbody>
</table>
## CICS directory

### Table 123. CICS modules directory (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHSDGDS</td>
<td>DSECT</td>
<td>System dump global statistics</td>
<td>C2 07</td>
</tr>
<tr>
<td>DFHSDMP</td>
<td>Macro</td>
<td>System dump parameter area and MD=L expansion</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHSDRDS</td>
<td>DSECT</td>
<td>System dump statistics by dump code</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHSDRDS</td>
<td>DSECT</td>
<td>System dump statistics by dump code</td>
<td>C2 07</td>
</tr>
<tr>
<td>DFHSFP</td>
<td>CSECT</td>
<td>Sign-off program</td>
<td>0S 03</td>
</tr>
<tr>
<td>DFHSGTC</td>
<td>CSECT</td>
<td></td>
<td>OS -</td>
</tr>
<tr>
<td>DFHSITM</td>
<td>CSECT</td>
<td></td>
<td>OS -</td>
</tr>
<tr>
<td>DFHSHDM</td>
<td>CSECT</td>
<td>SH Domain Management</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSHDUF</td>
<td>CSECT</td>
<td>SH Domain Dump Formatting</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSHOFI</td>
<td>CSECT</td>
<td></td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSHPR</td>
<td>CSECT</td>
<td>SHPR CDURUN and Gate Module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSHPRT</td>
<td>CSECT</td>
<td></td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSHREI</td>
<td>CSECT</td>
<td></td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSHRM</td>
<td>CSECT</td>
<td>SHRM CDURUN and Gate Module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSHRQ</td>
<td>CSECT</td>
<td>Scheduler Services - Request Queue</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSHRQA</td>
<td>Other</td>
<td>SHRQ interface parameter area</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHSHRMQ</td>
<td>Macro</td>
<td>DFHSHRQ interface macro</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHSHRQT</td>
<td>CSECT</td>
<td></td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSHRQ1</td>
<td>CSECT</td>
<td></td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSHRR</td>
<td>CSECT</td>
<td>SHRR CDURUN and Gate Module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSHRRP</td>
<td>CSECT</td>
<td></td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSHRT</td>
<td>CSECT</td>
<td></td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSHRSP</td>
<td>CSECT</td>
<td></td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSHRT</td>
<td>CSECT</td>
<td>SHRT CDURUN and Gate Module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSHRTT</td>
<td>CSECT</td>
<td></td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSHRT1</td>
<td>CSECT</td>
<td></td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSHRT2</td>
<td>CSECT</td>
<td></td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSHSGS</td>
<td>CSECT</td>
<td>Component modules</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSHTC</td>
<td>CSECT</td>
<td></td>
<td>OS -</td>
</tr>
<tr>
<td>DFHSHTI</td>
<td>CSECT</td>
<td></td>
<td>OS -</td>
</tr>
<tr>
<td>DFHSHTRI</td>
<td>CSECT</td>
<td>SH Domain Trace Interpretation</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSHVPI</td>
<td>CSECT</td>
<td></td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSHWPL</td>
<td>DSECT</td>
<td>File control SHOWCAT parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHSHXM</td>
<td>CSECT</td>
<td>Scheduler Services XM Attach Client</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSA1</td>
<td>CSECT</td>
<td>System initialization - module A1</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSAI</td>
<td>CSECT</td>
<td>System initialization - module B1</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSAIB</td>
<td>Source</td>
<td>DFHSIB1 pre-nucleus load routines</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHSAIB1</td>
<td>Source</td>
<td>DFHSIB1 nucleus load routine</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHSAIBC</td>
<td>Source</td>
<td>DFHSIB1 post-nucleus load routine</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHSAIBD</td>
<td>Source</td>
<td>DFHSIB1 subroutines</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHSICOM</td>
<td>Macro</td>
<td>System initialization definitions</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHSCI</td>
<td>CSECT</td>
<td>System initialization - module C1</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSID</td>
<td>CSECT</td>
<td>System initialization - module D1</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSID1</td>
<td>CSECT</td>
<td>System initialization - module F1</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSIDG</td>
<td>CSECT</td>
<td>System initialization - module G1</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSHI1</td>
<td>CSECT</td>
<td>System initialization - module H1</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSHII1</td>
<td>CSECT</td>
<td>System initialization - module I1</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSHII1</td>
<td>CSECT</td>
<td>System initialization - module J1</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSHIPD</td>
<td>Macro</td>
<td>Generate system initialization</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHSHIRP</td>
<td>DSECT</td>
<td>SIP communication area</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHSHIPT</td>
<td>CSECT</td>
<td>System initialization - PLT processor</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSHIT</td>
<td>Macro</td>
<td>System initialization table</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHSHIT1</td>
<td>Sample</td>
<td>Default system initialization table</td>
<td>19 03</td>
</tr>
<tr>
<td>DFHSHIT6</td>
<td>Sample</td>
<td>System initialization table</td>
<td>19 03</td>
</tr>
</tbody>
</table>
### Table 123. CICS modules directory (continued)

#### Name Type Description Library

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHSJAS</td>
<td>CSECT</td>
<td>SJ Assembler routines for DFHSJCS</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSJCSO</td>
<td>CSECT</td>
<td>Autocall SCEEOBJ</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSJDM</td>
<td>CSECT</td>
<td>SJ SJVM Domain</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSJDF</td>
<td>CSECT</td>
<td>SJ SJVM Domain</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSJGDS</td>
<td>DSECT</td>
<td>Jvmpool Global Statistics</td>
<td>11 07</td>
</tr>
<tr>
<td>DFHSJIN</td>
<td>CSECT</td>
<td>SJ SVM Domain</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSJINT</td>
<td>CSECT</td>
<td>SJ JVM Domain</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSJIS</td>
<td>CSECT</td>
<td>SJ JVM Domain</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSJIST</td>
<td>CSECT</td>
<td>SJ JVM Domain</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSJJBH</td>
<td>CSECT</td>
<td>SJ JVM Domain</td>
<td>- 08</td>
</tr>
<tr>
<td>DFHSJJBO</td>
<td>CSECT</td>
<td>SJ JVM Domain</td>
<td>0S 03</td>
</tr>
<tr>
<td>DFHSJJST</td>
<td>CSECT</td>
<td>(SOCKETS) Statistics functions</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSJJTRI</td>
<td>CSECT</td>
<td>SJ SJVM Domain</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSK</td>
<td>Macro</td>
<td>Subtasking interface</td>
<td>0S -</td>
</tr>
<tr>
<td>DFHSKC</td>
<td>CSECT</td>
<td>Subtask control program</td>
<td>0S 03</td>
</tr>
<tr>
<td>DFHSKE</td>
<td>CSECT</td>
<td>Subtask execution program</td>
<td>0S 03</td>
</tr>
<tr>
<td>DFHSKM</td>
<td>CSECT</td>
<td>Subtask manager</td>
<td>0S 03</td>
</tr>
<tr>
<td>DFHSKR</td>
<td>Macro</td>
<td>Generate SKR table entries in SIT</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHSKTSK</td>
<td>CSECT</td>
<td>General purpose subtask entry point</td>
<td>0S 03</td>
</tr>
<tr>
<td>DFHSLDC</td>
<td>DSECT</td>
<td>System logical device code table</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHSMAD</td>
<td>CSECT</td>
<td>SM domain - add/delete subpool</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSMADTA</td>
<td>DSECT</td>
<td>SMAD parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHSMADM</td>
<td>Macro</td>
<td>SMAD request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHSMADTG</td>
<td>CSECT</td>
<td>SMAD trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSMCAF</td>
<td>DSECT</td>
<td>SMAF parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHSMFA</td>
<td>CSECT</td>
<td>SMAF trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSMMS</td>
<td>CSECT</td>
<td>SM domain - handle functions</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSMSC</td>
<td>CSECT</td>
<td>SM domain - storage checking/recovery</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSMCC</td>
<td>DSECT</td>
<td>SMCC parameter list</td>
<td>0S -</td>
</tr>
<tr>
<td>DFHSMCK</td>
<td>Macro</td>
<td>SMCK request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHSMCKTA</td>
<td>CSECT</td>
<td>SMCK trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSMDDS</td>
<td>DSECT</td>
<td>SM domain - storage statistics for domain subpools</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHSMDDS</td>
<td>DSECT</td>
<td>SM domain - storage statistics for domain subpools</td>
<td>C2 07</td>
</tr>
<tr>
<td>DFHSMDDM</td>
<td>CSECT</td>
<td>SM domain - initialization/termination</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSMDFU</td>
<td>DSECT</td>
<td>SDUMP formatter for SM domain</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSMDFS</td>
<td>DSECT</td>
<td>SMF header and product section (JC/MN/ST)</td>
<td>11 07</td>
</tr>
<tr>
<td>DFHSMGF</td>
<td>CSECT</td>
<td>SM domain - getmain/freemain</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSMGFA</td>
<td>DSECT</td>
<td>SMGF parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHSMGF1</td>
<td>Macro</td>
<td>SM domain - inline getmain/freemain</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHSMGFM</td>
<td>Macro</td>
<td>SMGF request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHSMGFT</td>
<td>CSECT</td>
<td>SMGF trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSSMCA</td>
<td>DSECT</td>
<td>SMMC parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHSSMCCI</td>
<td>CSECT</td>
<td>SM domain - macro-compatibility initialize</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSSMCM</td>
<td>Macro</td>
<td>SMMC request</td>
<td>0S -</td>
</tr>
<tr>
<td>DFHSSMCT</td>
<td>CSECT</td>
<td>SMMC trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSSMCCX</td>
<td>Macro</td>
<td>SMMC request (XPI)</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHSSMCCY</td>
<td>DSECT</td>
<td>SMMC parameter list (XPI)</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHSSMCC2</td>
<td>CSECT</td>
<td>SM domain - macro-compatibility system freemain functions</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSSMCF</td>
<td>CSECT</td>
<td>SM domain - macro-compatibility freemain interface</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSSMCF</td>
<td>CSECT</td>
<td>SM domain - macro-compatibility getmain interface</td>
<td>- 03</td>
</tr>
</tbody>
</table>
### Table 123. CICS modules directory (continued)

**Name**  | **Type** | **Description** | **Library**
---|---|---|---
DFHSMNTA | DSECT | SMNT parameter list | OS -
DFHSMNTM | Macro | SMNT request | OS -
DFHSMNTT | CSECT (OCO) | SMNT trace interpretation data | - 03
DFHSMPE | Other | Cataloged procedure to execute SMP/E | 02 -
DFHSMPP | CSECT (OCO) | SM domain - pagepool manager functions 1 | - 03
DFHSMPTC | CSECT (OCO) | SMPP trace interpretation data | - 03
DFHSMPPQ | CSECT (OCO) | SM domain - pagepool manager functions 2 | - 03
DFHSMPTQ | CSECT (OCO) | SMPQ trace interpretation data | - 03
DFHSMPT | Macro | SMP/E control card generator | 11 -
DFHSMSCP | CSECT (OCO) | Storage control program | - 03
DFHSMSDS | DSECT | SM domain - storage statistics for DSAs | 11 -
DFHSMSDS | DSECT | SM domain - storage statistics for DSAs | C2 07
DFHSMSQ | CSECT (OCO) | SM domain - suspend queue manager | - 03
DFHSMSQT | CSECT (OCO) | SMSQ trace interpretation data | - 03
DFHSMSR | CSECT (OCO) | SM domain - services | - 03
DFHSMSRA | DSECT | SMRR parameter list | OS -
DFHSMRRI | CSECT | SM domain - in-line INQUIRE_ACCESS | OS -
DFHSMRRM | Macro | SMSR request | OS -
DFHSMRRT | CSECT (OCO) | SMSR trace interpretation data | - 03
DFHSMSRX | Macro (OCO) | SMSR request (XPI) | 11 -
DFHSMSRY | DSECT (OCO) | SMSR parameter list | 11 -
DFHSMST | CSECT (OCO) | SM domain - statistics collection | - 03
DFHSMUX | CSECT (OCO) | Subspace manager | - 03
DFHSMUT | CSECT (OCO) | Subspace manager trace interpretation data | - 03
DFHSMSCV | CSECT (OCO) | SM domain - authorized service routine | - 03
DFHSMSY | CSECT (OCO) | SM domain - system task | - 03
DFHSMTAB | CSECT | CICSplex SM commands language table | - 03
DFHSMQDS | CSECT | SM domain - storage statistics for task | 11 -
DFHSMQDS | DSECT | SM domain - storage statistics for task | C2 -
DFHSMQDS | DSECT | SM domain - storage statistics for task | P2 -
DFHSMQDS | DSECT | SM domain - storage statistics for task | subpools
DFHSMQDS | DSECT | SM domain - storage statistics for task | subpools
DFHSMTI | CSECT (OCO) | Trace interpreter for SM domain | - 03
DFHSMTUL | CSECT | SM Catalog Update Program | 03 03
DFHSXMLF | CSECT (OCO) | Transaction dump - task subpools | - 03
DFHSNAS | CSECT | create signon/sign-off ATI sessions | - 03
DFHSNEP | Macro | Node error program generator | 11 -
DFHSNEPH | Macro | NEP inner macro | 11 -
DFHSNET | Macro | Node error table generator | 11 -
DFHSNEX | Macro | Signon extension block generator | 11 -
DFHSNEXD | DSECT | Signon extension to TCTTE | 05 -
DFHSNGND | DSECT | CEGN parameter list | 05 -
DFHSNGSD | DSECT | GNTRAN parameter list | 11 -
DFHSNGSH | DSECT | GNTRAN parameter list (C/370) | - 08
DFHSNGSL | DSECT | GNTRAN parameter list (PL/I) | - 17
DFHSNGSO | DSECT | GNTRAN parameter list (COBOL) | C2 -
DFHSNLE | CSECT | Signon large screens map set | 05 03
DFHSNLK | CSECT (OCO) | Signon large screens map set | - 03
DFHSNMEG | CSECT | Signon table migration utility | 05 03
DFHSNNFY | CSECT | RACF CICS segment notify exit | 05 03
DFHSPN | CSECT | Signon program | 05 03
DFHSNP10 | CSECT | CICS segment (RACF) TIMEOUT keyword | - 03
DFHSNP11 | CSECT | Signon program | print exit routine
DFHSNPU | CSECT | Preset userid signon/sign-off | - 03
### Table 123. CICS modules directory (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHSNNSC</td>
<td>CSECT</td>
<td>Timeout transaction (CESC) scheduler</td>
<td>03</td>
</tr>
<tr>
<td>DFHSNSSCA</td>
<td>CSECT</td>
<td>SNSC parameter list</td>
<td>OS</td>
</tr>
<tr>
<td>DFHSNSCM</td>
<td>Macro</td>
<td>SNSC requests</td>
<td>OS</td>
</tr>
<tr>
<td>DFHSNSE</td>
<td>CSECT</td>
<td>Signon small screens map set</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSNSG</td>
<td>CSECT</td>
<td>Surrogate terminal signon/off</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSNSGI</td>
<td>Macro</td>
<td>Surrogate terminals sign-on and signoff requests</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHSNSK</td>
<td>CSECT</td>
<td>Signon small screens map set</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSNSTA</td>
<td>DSECT</td>
<td>ISC/IRC attach-time statistics area</td>
<td>OS</td>
</tr>
<tr>
<td>DFHSNSU</td>
<td>CSECT</td>
<td>Session userid signon/sign-off</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSNTRI</td>
<td>CSECT</td>
<td>SN trace interpreter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSNTU</td>
<td>CSECT</td>
<td>Terminal userid signon/sign-off</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSNUS</td>
<td>CSECT</td>
<td>US domain - local and remote signon</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSNUSA</td>
<td>DSECT</td>
<td>SNUS parameter list</td>
<td>OS</td>
</tr>
<tr>
<td>DFHSNUSM</td>
<td>Macro</td>
<td>SNUS macro</td>
<td>OS</td>
</tr>
<tr>
<td>DFHSNUST</td>
<td>CSECT</td>
<td>SNUS trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSNVCL</td>
<td>CSECT</td>
<td>RACF CICS segment OPCLASS validation exit</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSNVIO</td>
<td>CSECT</td>
<td>RACF CICS segment OPIDENT validation exit</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSNVPR</td>
<td>CSECT</td>
<td>RACF CICS segment OPPRTY validation exit</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSNVTO</td>
<td>CSECT</td>
<td>RACF CICS segment TIMEOUT validation exit</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSNXR</td>
<td>CSECT</td>
<td>XRF reflecting signon state</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSNXRA</td>
<td>DSECT</td>
<td>SNXR parameter list</td>
<td>OS</td>
</tr>
<tr>
<td>DFHSNXRM</td>
<td>Macro</td>
<td>SNXR requests</td>
<td>OS</td>
</tr>
<tr>
<td>DFHSNXRT</td>
<td>CSECT</td>
<td>SNXR trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSOAD</td>
<td>CSECT</td>
<td>SO Domain - SOAD gate functions</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSOADT</td>
<td>CSECT</td>
<td>-</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSOCTB</td>
<td>CSECT</td>
<td>-</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSOCK</td>
<td>CSECT</td>
<td>Sockets send/receive/close</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSOCKT</td>
<td>CSECT</td>
<td>-</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSODM</td>
<td>CSECT</td>
<td>Sockets Domain Initialization</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSODUF</td>
<td>CSECT</td>
<td>Sockets Domain Dump Formatting</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSODS</td>
<td>DSECT</td>
<td>Sockets Global Statistics</td>
<td>11 07</td>
</tr>
<tr>
<td>DFHSOGH@</td>
<td>DSECT</td>
<td>-</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSOIS</td>
<td>CSECT</td>
<td>Sockets Domain Inquire/Set</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSOIST</td>
<td>CSECT</td>
<td>-</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSOISL</td>
<td>CSECT</td>
<td>Sockets Domain Listener Task</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSOISTS</td>
<td>CSECT</td>
<td>Sockets Listener</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSOISTE</td>
<td>CSECT</td>
<td>-</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSOLX</td>
<td>CSECT</td>
<td>Sockets Domain Asynchronous exit routine</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSOPI</td>
<td>CSECT</td>
<td>SO Domain CEEPIPI service routines</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSORD</td>
<td>CSECT</td>
<td>SO Domain Sockets Register/Deregister</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSORDS</td>
<td>CSECT</td>
<td>SO Domain TCP/IP Service Statistics</td>
<td>11 07</td>
</tr>
<tr>
<td>DFHSORDT</td>
<td>CSECT</td>
<td>-</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSORT</td>
<td>Macro</td>
<td>Auxiliary sort</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHSOSE</td>
<td>CSECT</td>
<td>Sockets Domain Secure Sockets Layer</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSOSET</td>
<td>CSECT</td>
<td>-</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSOSKO</td>
<td>CSECT</td>
<td>-</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSOST</td>
<td>CSECT</td>
<td>Sockets Statistics Functions</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSOS00</td>
<td>CSECT</td>
<td>-</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSOS01</td>
<td>CSECT</td>
<td>-</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSOS02</td>
<td>CSECT</td>
<td>-</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSOS03</td>
<td>CSECT</td>
<td>-</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSOS04</td>
<td>CSECT</td>
<td>-</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSOS05</td>
<td>CSECT</td>
<td>-</td>
<td>- 03</td>
</tr>
</tbody>
</table>
Table 123. CICS modules directory (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHSOS06</td>
<td>CSECT</td>
<td>- 03</td>
<td></td>
</tr>
<tr>
<td>DFHSOS07</td>
<td>CSECT</td>
<td>- 03</td>
<td></td>
</tr>
<tr>
<td>DFHSOS08</td>
<td>CSECT</td>
<td>- 03</td>
<td></td>
</tr>
<tr>
<td>DFHSOS09</td>
<td>CSECT</td>
<td>- 03</td>
<td></td>
</tr>
<tr>
<td>DFHSOS10</td>
<td>CSECT</td>
<td>- 03</td>
<td></td>
</tr>
<tr>
<td>DFHSOS11</td>
<td>CSECT</td>
<td>- 03</td>
<td></td>
</tr>
<tr>
<td>DFHSOS12</td>
<td>CSECT</td>
<td>- 03</td>
<td></td>
</tr>
<tr>
<td>DFHSOS13</td>
<td>CSECT</td>
<td>- 03</td>
<td></td>
</tr>
<tr>
<td>DFHSOS14</td>
<td>CSECT</td>
<td>- 03</td>
<td></td>
</tr>
<tr>
<td>DFHSOS15</td>
<td>CSECT</td>
<td>- 03</td>
<td></td>
</tr>
<tr>
<td>DFHSOS16</td>
<td>CSECT</td>
<td>- 03</td>
<td></td>
</tr>
<tr>
<td>DFHSOS17</td>
<td>CSECT</td>
<td>- 03</td>
<td></td>
</tr>
<tr>
<td>DFHSOS18</td>
<td>CSECT</td>
<td>- 03</td>
<td></td>
</tr>
<tr>
<td>DFHSOS19</td>
<td>CSECT</td>
<td>- 03</td>
<td></td>
</tr>
<tr>
<td>DFHSOS20</td>
<td>CSECT</td>
<td>- 03</td>
<td></td>
</tr>
<tr>
<td>DFHSOS21</td>
<td>CSECT</td>
<td>- 03</td>
<td></td>
</tr>
<tr>
<td>DFHSOS22</td>
<td>CSECT</td>
<td>- 03</td>
<td></td>
</tr>
<tr>
<td>DFHSOS23</td>
<td>CSECT</td>
<td>- 03</td>
<td></td>
</tr>
<tr>
<td>DFHSOTB</td>
<td>CSECT</td>
<td>SO Domain SOTB Gate Functions</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSOTBT</td>
<td>CSECT</td>
<td></td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSOTI</td>
<td>CSECT</td>
<td>Sockets Timer</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSOTRI</td>
<td>CSECT</td>
<td>Sockets Domain Trace Interpretation</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSOUE</td>
<td>CSECT</td>
<td>Sockets Domain User Exit Services</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSOXMR</td>
<td>CSECT</td>
<td>Sockets Attach Client</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSP</td>
<td>Macro</td>
<td>Syncpoint service request</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHSPBAB</td>
<td>CSECT</td>
<td>- 03</td>
<td></td>
</tr>
<tr>
<td>DFHSPBAC</td>
<td>CSECT</td>
<td>- 03</td>
<td></td>
</tr>
<tr>
<td>DFHSPBCE</td>
<td>CSECT</td>
<td>- 03</td>
<td></td>
</tr>
<tr>
<td>DFHSOBB</td>
<td>CSECT</td>
<td>OS 03</td>
<td></td>
</tr>
<tr>
<td>DFHSOBB</td>
<td>CSECT</td>
<td>OS 03</td>
<td></td>
</tr>
<tr>
<td>DFHSOBB</td>
<td>CSECT</td>
<td>OS 03</td>
<td></td>
</tr>
<tr>
<td>DFHSPDCB</td>
<td>CSECT</td>
<td>OS 03</td>
<td></td>
</tr>
<tr>
<td>DFHSPDCB</td>
<td>CSECT</td>
<td>OS 03</td>
<td></td>
</tr>
<tr>
<td>DFHSPDCB</td>
<td>CSECT</td>
<td>OS 03</td>
<td></td>
</tr>
<tr>
<td>DFHSPDCE</td>
<td>CSECT</td>
<td>- 03</td>
<td></td>
</tr>
<tr>
<td>DFHSPDHE</td>
<td>CSECT</td>
<td>- 03</td>
<td></td>
</tr>
<tr>
<td>DFHSPJE</td>
<td>CSECT</td>
<td>- 03</td>
<td></td>
</tr>
<tr>
<td>DFHSPJE</td>
<td>CSECT</td>
<td>- 03</td>
<td></td>
</tr>
<tr>
<td>DFHSPJE</td>
<td>CSECT</td>
<td>- 03</td>
<td></td>
</tr>
<tr>
<td>DFHSPFIB</td>
<td>CSECT</td>
<td>CSDUP - cross-keyword validation for files</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSPFIC</td>
<td>CSECT</td>
<td>RDO - cross-keyword validation for files</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSPFIE</td>
<td>CSECT</td>
<td>RDO file definition validation</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSPKCB</td>
<td>CSECT</td>
<td>CSDUP - cross-keyword validation for transactions and profiles</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSPKCC</td>
<td>CSECT</td>
<td>RDO - cross-keyword validation for transactions and profiles</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSPKCE</td>
<td>CSECT</td>
<td>RDO txn control definition validation</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSPLMC</td>
<td>CSECT</td>
<td>RDO JournalModel definition validation</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSPLMC</td>
<td>CSECT</td>
<td>RDO JournalModel definition validation</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSPLMC</td>
<td>CSECT</td>
<td>RDO JournalModel definition validation</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSPLMC</td>
<td>CSECT</td>
<td>RDO JournalModel definition validation</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSPMLB</td>
<td>CSECT</td>
<td>CSDUP - cross-keyword validation for LSR pools</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSPMLB</td>
<td>CSECT</td>
<td>RDO - cross-keyword validation for LSR pools</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSPLNB</td>
<td>CSECT</td>
<td>RDO - Lsrpool definition validation</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSPNQ7</td>
<td>CSECT</td>
<td>OS 03</td>
<td></td>
</tr>
<tr>
<td>DFHSPNQ7</td>
<td>CSECT</td>
<td>OS 03</td>
<td></td>
</tr>
<tr>
<td>DFHSPNQ7</td>
<td>CSECT</td>
<td>OS 03</td>
<td></td>
</tr>
</tbody>
</table>
### Table 123. CICS modules directory (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHSPOPB</td>
<td>CSECT</td>
<td>-03</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSPOPC</td>
<td>CSECT</td>
<td>-03</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSPOPE</td>
<td>CSECT</td>
<td>-03</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSPPCB</td>
<td>CSECT</td>
<td>CSDUP - cross-keyword validation for programs, map sets, and partition sets</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSPPCC</td>
<td>CSECT</td>
<td>RDO - cross-keyword validation for programs, map sets, and partition sets</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSPPCE</td>
<td>CSECT</td>
<td>RDO - program definition validation</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSPPNB</td>
<td>CSECT</td>
<td>CSDUP - cross-keyword validation for partners</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSPPNC</td>
<td>CSECT</td>
<td>RDO - cross-keyword validation for partners</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSPPNE</td>
<td>CSECT</td>
<td>RDO - partner definition validation</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSPPOB</td>
<td>CSECT</td>
<td>-03</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSPPOC</td>
<td>CSECT</td>
<td>-03</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSPPOE</td>
<td>CSECT</td>
<td>-03</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSPPNB</td>
<td>CSECT</td>
<td>CSDUP - cross-keyword validation for OS 03 partners</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSPPNC</td>
<td>CSECT</td>
<td>RDO - cross-keyword validation for OS 03 partners</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSPPNE</td>
<td>CSECT</td>
<td>RDO - partner definition validation</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSPPOB</td>
<td>CSECT</td>
<td>-03</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSPPOC</td>
<td>CSECT</td>
<td>-03</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSPPOE</td>
<td>CSECT</td>
<td>-03</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSPPCB</td>
<td>CSECT</td>
<td>CSDUP - cross-keyword validation for terminals</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSPPCC</td>
<td>CSECT</td>
<td>RDO - cross-keyword validation for terminals</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSPPCE</td>
<td>CSECT</td>
<td>RDO - terminal definition validation</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSPPDB</td>
<td>CSECT</td>
<td>RDO - TDQueue definition validation</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSPPDC</td>
<td>CSECT</td>
<td>RDO - TDQueue definition validation</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSPPDE</td>
<td>CSECT</td>
<td>RDO - TDQueue definition validation</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSPPTB</td>
<td>CSECT</td>
<td>CSDUP - cross-keyword validation for sessions</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSPPTC</td>
<td>CSECT</td>
<td>RDO - cross-keyword validation for sessions</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSPPTD</td>
<td>CSECT</td>
<td>RDO - sessions definition validation</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSPPTE</td>
<td>CSECT</td>
<td>RDO - sessions definition validation</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSPPTF</td>
<td>CSECT</td>
<td>RDO - sessions definition validation</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSPPTG</td>
<td>CSECT</td>
<td>RDO - sessions definition validation</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSPPTH</td>
<td>CSECT</td>
<td>RDO - cross-keyword validation for connections</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSPPTI</td>
<td>CSECT</td>
<td>SPI trace interpreter</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSPPTJ</td>
<td>CSECT</td>
<td>RDO - cross-keyword validation for connections</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSPPTK</td>
<td>CSECT</td>
<td>RDO - cross-keyword validation for transactions</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSPPTL</td>
<td>CSECT</td>
<td>RDO - cross-keyword validation for transactions</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSPPTN</td>
<td>CSECT</td>
<td>RDO - cross-keyword validation for typeterms</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSPPTO</td>
<td>CSECT</td>
<td>RDO - cross-keyword validation for typeterms</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSPPTP</td>
<td>CSECT</td>
<td>RDO - Typeterms definition validation</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSPPTQ</td>
<td>CSECT</td>
<td>Syncpoint program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSPPTR</td>
<td>CSECT</td>
<td>Sample TCAM MCP for TCAM 10</td>
<td>19 -</td>
</tr>
<tr>
<td>DFHSPPTS</td>
<td>CSECT</td>
<td>Sample TCAM MCP for TCAM 10</td>
<td>19 -</td>
</tr>
<tr>
<td>DFHSPPTU</td>
<td>CSECT</td>
<td>CSDUP - cross-keyword validation for transactions</td>
<td>03 -</td>
</tr>
<tr>
<td>DFHSPPTV</td>
<td>CSECT</td>
<td>RDO - cross-keyword validation for transactions</td>
<td>03 -</td>
</tr>
<tr>
<td>DFHSPPTW</td>
<td>CSECT</td>
<td>RDO - TranClass definition validation</td>
<td>03 -</td>
</tr>
<tr>
<td>DFHSPPTX</td>
<td>CSECT</td>
<td>SRB interface control area</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHSPPTY</td>
<td>CSECT</td>
<td>Alias for SRRHASM</td>
<td>11 -</td>
</tr>
</tbody>
</table>
### Table 123. CICS modules directory (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHSROB</td>
<td>CSECT</td>
<td>Alias for SRR Marseille</td>
<td>C2</td>
</tr>
<tr>
<td>DFHSRED</td>
<td>CSECT</td>
<td>System recovery error data for XSRAB exit</td>
<td>11</td>
</tr>
<tr>
<td>DFHSLII</td>
<td>CSECT</td>
<td>SRP LIFO storage subroutine</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSRLIA</td>
<td>DSECT</td>
<td>SRL parameter list</td>
<td>OS</td>
</tr>
<tr>
<td>DFHSLIM</td>
<td>Macro</td>
<td>SRL request</td>
<td>OS</td>
</tr>
<tr>
<td>DFHSLIT</td>
<td>CSECT</td>
<td>SRL interpretation data</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSLIPI</td>
<td>CSECT</td>
<td>System recovery program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSRPII</td>
<td>CSECT</td>
<td>Alias for SRR Marseille</td>
<td>P2</td>
</tr>
<tr>
<td>DFHSRSR</td>
<td>Source</td>
<td>SRR parameter list</td>
<td>OS</td>
</tr>
<tr>
<td>DFHSRSRM</td>
<td>Source</td>
<td>SRR request</td>
<td>OS</td>
</tr>
<tr>
<td>DFHST</td>
<td>Macro</td>
<td>System recovery table</td>
<td>11</td>
</tr>
<tr>
<td>DFHSTDS</td>
<td>DSECT</td>
<td>System recovery table</td>
<td>OS</td>
</tr>
<tr>
<td>DFHSRT1</td>
<td>Sample</td>
<td>System recovery table</td>
<td>19 03</td>
</tr>
<tr>
<td>DFHSTRXS</td>
<td>DSECT</td>
<td>SRB and extensions in SQA</td>
<td>OS</td>
</tr>
<tr>
<td>DFHSRV1</td>
<td>CSECT</td>
<td>System recovery program</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSSAD</td>
<td>Macro</td>
<td>Static storage area address list</td>
<td>11</td>
</tr>
<tr>
<td>DFHSSUF</td>
<td>CSECT (OCO)</td>
<td>SDUMP formatter for static storage areas</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSSHEN</td>
<td>CSECT</td>
<td>Subsystem interface EOT and EDM routine</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSSGC</td>
<td>CSECT</td>
<td>Subsystem interface generic connect</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSSIN</td>
<td>CSECT</td>
<td>CICS subsystem initialization</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSSMGMP</td>
<td>CSECT</td>
<td>Subsystem interface message program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSSMTG</td>
<td>CSECT</td>
<td>Subsystem interface message table</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSSREQ</td>
<td>Macro</td>
<td>Subsystem interface (SSI) request</td>
<td>OS</td>
</tr>
<tr>
<td>DFHSSWT</td>
<td>CSECT</td>
<td>Subsystem interface WTO router</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSSWF</td>
<td>CSECT</td>
<td>SSI MODIFY command password suppression</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSSWTO</td>
<td>CSECT</td>
<td>SSI CICS console message reformatting</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSTAB</td>
<td>Macro</td>
<td>Table scan macro</td>
<td>11</td>
</tr>
<tr>
<td>DFHSTACK</td>
<td>Macro</td>
<td>Save/restore registers on subroutine calls</td>
<td>OS</td>
</tr>
<tr>
<td>DFHSTAB</td>
<td>Other</td>
<td>CICS startup cataloged procedure</td>
<td>02</td>
</tr>
<tr>
<td>DFHSTDBX</td>
<td>CSECT (OCO)</td>
<td>STUP - DBCTL statistics summary formatter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTDX</td>
<td>CSECT (OCO)</td>
<td>STUP - DCE services domain extended formatter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTDG</td>
<td>CSECT (OCO)</td>
<td>ST domain - initialization/termination</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTDS</td>
<td>CSECT (OCO)</td>
<td>STUP - DS domain stats summary formatter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTDF</td>
<td>CSECT (OCO)</td>
<td>STUP - DUMP ST format for ST domain</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTDX</td>
<td>CSECT (OCO)</td>
<td>STUP - DU domain stats summary formatter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTI1</td>
<td>Macro</td>
<td>Standard names of domains, gates, formats</td>
<td>11</td>
</tr>
<tr>
<td>DFHSTI1X</td>
<td>CSECT</td>
<td>Stats.Util.EJ Domain Extended formatting</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTE15</td>
<td>CSECT (OCO)</td>
<td>STUP - DFSORT interface to E15 user exit</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTE35</td>
<td>CSECT (OCO)</td>
<td>STUP - DFSORT interface to E35 user exit</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTFC</td>
<td>CSECT</td>
<td>AP domain - file control statistics</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTGDS</td>
<td>CSECT</td>
<td>ST domain - global statistics</td>
<td>11</td>
</tr>
<tr>
<td>DFHSTGDS</td>
<td>CSECT</td>
<td>ST domain - global statistics</td>
<td>C2 07</td>
</tr>
<tr>
<td>DFHSTIDS</td>
<td>CSECT</td>
<td>Statistics common record header and record identifiers</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHSTID15</td>
<td>CSECT</td>
<td>Stats.Util.II Domain Extended formatting</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTPDF</td>
<td>CSECT (OCO)</td>
<td>STUP - DFSORT E15 user exit input routine</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTDE15</td>
<td>CSECT (OCO)</td>
<td>STUP - LD domain stats summary formatter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTDFX</td>
<td>CSECT (OCO)</td>
<td>Logger Domain statistics extended</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTK</td>
<td>CSECT</td>
<td>AP domain - ISC/IRC statistics</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTLS</td>
<td>CSECT</td>
<td>AP domain - LSR pool statistics</td>
<td>- 03</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
<td>Library</td>
</tr>
<tr>
<td>--------------</td>
<td>----------</td>
<td>--------------------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>DFHSTMNX</td>
<td>CSECT (OCO)</td>
<td>STUP - MN domain stats summary formatter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTNQX</td>
<td>CSECT (OCO)</td>
<td>Enqueue Manager domain statistics</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTOT</td>
<td>CSECT (OCO)</td>
<td>STUP - DFSORT E35 user exit output routine</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTP</td>
<td>CSECT</td>
<td>System termination program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSTPGX</td>
<td>CSECT</td>
<td>STUP - PG domain autoinstall statistics</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTRDA</td>
<td>DSECT</td>
<td>STRD parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHSTRDM</td>
<td>Macro</td>
<td>STRD request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHSTRMX</td>
<td>CSECT (OCO)</td>
<td>Recovery Manager domain statistics</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTJX</td>
<td>CSECT</td>
<td>Stats.Util.JVM Domain Extended formatting</td>
<td>11 - product section</td>
</tr>
<tr>
<td>DFHSTSMF</td>
<td>Macro</td>
<td>ST domain - statistics SMF header and SMF</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHSTSMX</td>
<td>CSECT (OCO)</td>
<td>STUP - SM domain stats summary formatter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTTX</td>
<td>CSECT (OCO)</td>
<td>STUP - ST domain stats summary formatter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTSZ</td>
<td>CSECT</td>
<td>AP domain - FEPI statistics</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTTI</td>
<td>CSECT</td>
<td>AP domain - transient data statistics</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTTM</td>
<td>CSECT</td>
<td>AP domain - table manager statistics</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTTTQX</td>
<td>CSECT</td>
<td>STUP - TDQueue id extended formatting</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTTR</td>
<td>CSECT</td>
<td>AP domain - terminal statistics</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTTRI</td>
<td>CSECT (OCO)</td>
<td>Trace interpreter for ST domain</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTTX</td>
<td>CSECT (OCO)</td>
<td>Shared TS statistics</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTDB</td>
<td>CSECT (OCO)</td>
<td>STUP - DBCTL statistics formatter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTUDC</td>
<td>CSECT (OCO)</td>
<td>STUP - DE domain statistics formatter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTUDS</td>
<td>CSECT (OCO)</td>
<td>STUP - DS domain statistics formatter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTUDU</td>
<td>CSECT (OCO)</td>
<td>STUP - DU domain statistics formatter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTUDQ</td>
<td>CSECT (OCO)</td>
<td>STUP - DU domain statistics formatter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTUE</td>
<td>CSECT (OCO)</td>
<td>ST domain - user exit service</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTUEJ</td>
<td>CSECT</td>
<td>STUP - EJ Domain formatting routine</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTUII</td>
<td>CSECT</td>
<td>STUP - II Domain formatting routine</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTULD</td>
<td>CSECT (OCO)</td>
<td>STUP - LD domain statistics formatter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTULG</td>
<td>CSECT (OCO)</td>
<td>STUP - Logger domain formatting routine</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTUMN</td>
<td>CSECT (OCO)</td>
<td>STUP - MN domain statistics formatter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTUNQ</td>
<td>CSECT (OCO)</td>
<td>STUP - Enqueue manager domain statistics</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTUPG</td>
<td>CSECT (OCO)</td>
<td>STUP - PG domain autoinstall statistics</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTUP1</td>
<td>CSECT (OCO)</td>
<td>STUP - preinitialize</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTURM</td>
<td>CSECT (OCO)</td>
<td>STUP - Recovery manager domain statistics</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTURS</td>
<td>CSECT (OCO)</td>
<td>STUP - US domain statistics formatter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTURX</td>
<td>CSECT (OCO)</td>
<td>STUP - US domain statistics summary</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTUSJ</td>
<td>CSECT</td>
<td>STUP - Scaleable JVM Domain formatting</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTUSM</td>
<td>CSECT (OCO)</td>
<td>STUP - SM domain statistics formatter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTUSO</td>
<td>CSECT</td>
<td>STUP - Sockets Domain formatting route</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTUST</td>
<td>CSECT (OCO)</td>
<td>STUP - ST domain statistics formatter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTUTQ</td>
<td>CSECT (OCO)</td>
<td>STUP - Transient data statistics</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTUTS</td>
<td>CSECT</td>
<td>Shared TS statistics</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTUXC</td>
<td>CSECT (OCO)</td>
<td>STUP - Transaction manager domain statistics</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTUXM</td>
<td>CSECT (OCO)</td>
<td>STUP - XM domain statistics formatter</td>
<td>- 03</td>
</tr>
</tbody>
</table>
### CICS directory

**Table 123. CICS modules directory (continued)**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHSTU03</td>
<td>CSECT (OCO)</td>
<td>STUP - VTAM statistics formatter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTU04</td>
<td>CSECT (OCO)</td>
<td>STUP - autoinstall terminals statistics formatter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTU06</td>
<td>CSECT (OCO)</td>
<td>STUP - terminal statistics formatter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTU08</td>
<td>CSECT (OCO)</td>
<td>STUP - LSRPOOL resource statistics formatter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTU09</td>
<td>CSECT (OCO)</td>
<td>STUP - LSRPOOL file statistics formatter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTU14</td>
<td>CSECT (OCO)</td>
<td>STUP - ISC/IRC statistics formatter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTU16</td>
<td>CSECT (OCO)</td>
<td>STUP - table manager statistics formatter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTU17</td>
<td>CSECT (OCO)</td>
<td>STUP - file control statistics formatter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTU21</td>
<td>CSECT (OCO)</td>
<td>STUP - ISC/IRC attach-time statistics formatter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTU22</td>
<td>CSECT (OCO)</td>
<td>STUP - FEPI statistics formatter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTWR</td>
<td>CSECT (OCO)</td>
<td>STUP - write interface</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTWRM</td>
<td>DSECT</td>
<td>STWR parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHSTXLE</td>
<td>CSECT</td>
<td>Transaction manager domain extended formatting routine for TranClass Stats</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTXMLE</td>
<td>CSECT</td>
<td>Off-line Statistics Utility Program</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTXM</td>
<td>CSECT</td>
<td>STUP - XM statistics extended formatter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTQ3X</td>
<td>CSECT (OCO)</td>
<td>STUP - VTAM statistics summary formatter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTQ4X</td>
<td>CSECT (OCO)</td>
<td>STUP - autoinstall terminals statistics summary formatter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTQ6X</td>
<td>CSECT (OCO)</td>
<td>STUP - terminal stats summary formatter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTQ8X</td>
<td>CSECT (OCO)</td>
<td>STUP - LSRPOOL resource statistics summary formatter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSTQ9X</td>
<td>CSECT (OCO)</td>
<td>STUP - LSRPOOL file statistics summary formatter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHST14X</td>
<td>CSECT (OCO)</td>
<td>STUP - ISC/IRC stats summary formatter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHST16X</td>
<td>CSECT (OCO)</td>
<td>STUP - table manager statistics summary formatter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHST17X</td>
<td>CSECT (OCO)</td>
<td>STUP - file control statistics summary formatter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHST21X</td>
<td>CSECT (OCO)</td>
<td>STUP - ISC/IRC attach-time statistics summary formatter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHST22X</td>
<td>CSECT (OCO)</td>
<td>STUP - FEPI statistics summary formatter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSUDUF</td>
<td>CSECT (OCO)</td>
<td>SDUMP formatter for DU domain summary</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSEUX</td>
<td>CSECT</td>
<td>User exit handler subroutine</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSEUXA</td>
<td>DSECT</td>
<td>SUEX parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHSEUXM</td>
<td>Macro</td>
<td>SUEX request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHSEUXT</td>
<td>CSECT</td>
<td>SUEX trace interpretation data</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSUME</td>
<td>CSECT (OCO)</td>
<td>ME domain - produce and issue messages subroutine (used by ME and LM domains)</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSUMEA</td>
<td>DSECT</td>
<td>SUME parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHSUMEM</td>
<td>Macro</td>
<td>SUME request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHSUMET</td>
<td>CSECT</td>
<td>SUME trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSUNP</td>
<td>Other</td>
<td></td>
<td>OS -</td>
</tr>
<tr>
<td>DFHSUX</td>
<td>CSECT</td>
<td>XRF signon</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSUSXA</td>
<td>DSECT</td>
<td>SUSX parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHSUSXM</td>
<td>Macro</td>
<td>SUSX request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHSUSXT</td>
<td>DSECT</td>
<td>SUSX translate tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSUTRI</td>
<td>CSECT</td>
<td>WTO/WTOR subroutine trace interpreter</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSUWT</td>
<td>CSECT</td>
<td>WTO/WTOR interface subroutine</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSUWTA</td>
<td>DSECT</td>
<td>SUWT parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
<td>Library</td>
</tr>
<tr>
<td>----------</td>
<td>------------</td>
<td>--------------------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>DFHSUWTM</td>
<td>Macro</td>
<td>SUWT request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHSUWTT</td>
<td>CSECT</td>
<td>SUWT trace interpretation data</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSUZX</td>
<td>CSECT</td>
<td>ZC trace controller</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSUZXA</td>
<td>DSECT</td>
<td>SUZX parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHSUZXM</td>
<td>Macro</td>
<td>SUZX request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHSUZXT</td>
<td>CSECT</td>
<td>SUZX trace interpretation data</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHSVCKH</td>
<td>Macro</td>
<td>Switch execution key routine</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHSYS</td>
<td>Macro</td>
<td>System definition macro</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHSZAPA</td>
<td>DSECT</td>
<td>FEPI programming copybook - assembler</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHSZAPC</td>
<td>DSECT</td>
<td>FEPI programming copybook - C/370</td>
<td>- 08</td>
</tr>
<tr>
<td>DFHSZAPD</td>
<td>DSECT</td>
<td>FEPI programming copybook - COBOL</td>
<td>C2 -</td>
</tr>
<tr>
<td>DFHSZAPF</td>
<td>DSECT</td>
<td>FEPI programming copybook - PL/I</td>
<td>P2 17</td>
</tr>
<tr>
<td>DFHSZATC</td>
<td>CSECT (OCO)</td>
<td>FEPI adaptor command tables</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSZATR</td>
<td>CSECT (OCO)</td>
<td>FEPI adaptor program</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSZBCL</td>
<td>CSECT (OCO)</td>
<td>FEPI cleanup API requests at error routine</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSZBCS</td>
<td>CSECT (OCO)</td>
<td>FEPI RM collect statistics</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSZBFT</td>
<td>CSECT (OCO)</td>
<td>FEPI FREE transaction requests scheduler</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSZBLO</td>
<td>CSECT (OCO)</td>
<td>FEPI lost session reporter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSZBRS</td>
<td>CSECT (OCO)</td>
<td>FEPI RM collect resource ID statistics</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSZBSI</td>
<td>CSECT (OCO)</td>
<td>FEPI signon exit scheduler</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSZBST</td>
<td>CSECT (OCO)</td>
<td>FEPI STSN transaction scheduler</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSZBUN</td>
<td>CSECT (OCO)</td>
<td>FEPI unsolicited data transaction scheduler</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSZBUS</td>
<td>CSECT (OCO)</td>
<td>FEPI RM unsolicited statistics recording</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSZDUF</td>
<td>CSECT (OCO)</td>
<td>FEPI dump formatting routine</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSZFRD</td>
<td>CSECT (OCO)</td>
<td>FEPI formatted 3270 RECEIVE support</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSZFSO</td>
<td>CSECT (OCO)</td>
<td>FEPI formatted 3270 SEND support</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSZIDX</td>
<td>CSECT (OCO)</td>
<td>FEPI SLU P queue install/discard exit</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSZPCP</td>
<td>CSECT (OCO)</td>
<td>FEPI SLU P flow controller</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSZPDX</td>
<td>CSECT (OCO)</td>
<td>FEPI SLU P drain completion exit</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSZPID</td>
<td>CSECT (OCO)</td>
<td>FEPI SLU P send data processor</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSZPIX</td>
<td>CSECT (OCO)</td>
<td>FEPI SLU P send completion exit</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSZPOA</td>
<td>CSECT (OCO)</td>
<td>FEPI SLU P send response processor</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSZPOD</td>
<td>CSECT (OCO)</td>
<td>FEPI SLU P receive data processor</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSZPOR</td>
<td>CSECT (OCO)</td>
<td>FEPI SLU P response processor</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSZPOX</td>
<td>CSECT (OCO)</td>
<td>FEPI SLU P receive specific response exit</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSZPOY</td>
<td>CSECT (OCO)</td>
<td>FEPI SLU P receive specific response processor</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSZPSA</td>
<td>CSECT (OCO)</td>
<td>FEPI SLU P REQSESS (request session) issuer</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSZPSQ</td>
<td>CSECT (OCO)</td>
<td>FEPI SLU P REQSESS exit</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSZPSB</td>
<td>CSECT (OCO)</td>
<td>FEPI SLU P bind processor</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSZPSR</td>
<td>CSECT (OCO)</td>
<td>FEPI SLU P session controller</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSZPSD</td>
<td>CSECT (OCO)</td>
<td>FEPI SLU P SDT processor</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSZPSH</td>
<td>CSECT (OCO)</td>
<td>FEPI SLU P SHUTC processor</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSZPSQ</td>
<td>CSECT (OCO)</td>
<td>FEPI SLU P quiesce complete (QC) processor</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSZPSR</td>
<td>CSECT (OCO)</td>
<td>FEPI RESESSR processor CSECT</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSZPSA</td>
<td>CSECT (OCO)</td>
<td>FEPI SLU P STSN processor</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSZPSX</td>
<td>CSECT (OCO)</td>
<td>FEPI SLU P OPNSEC completion exit</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSZPTE</td>
<td>CSECT (OCO)</td>
<td>FEPI SLU P TERMSESS processor</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSZRCA</td>
<td>CSECT (OCO)</td>
<td>FEPI node control processor</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSZRCT</td>
<td>CSECT (OCO)</td>
<td>FEPI issue processor</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSZRDC</td>
<td>CSECT (OCO)</td>
<td>FEPI delete connection processor</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSZRSG</td>
<td>CSECT (OCO)</td>
<td>FEPI discard node processor</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSZRDN</td>
<td>CSECT (OCO)</td>
<td>FEPI delete node processor</td>
<td>- 03</td>
</tr>
</tbody>
</table>
### CICS modules directory (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHSZRDPC</td>
<td>CSECT</td>
<td>FEPI dispatcher</td>
<td>03</td>
</tr>
<tr>
<td>DFHSZRDSD</td>
<td>CSECT</td>
<td>FEPI discard property set processor</td>
<td>03</td>
</tr>
<tr>
<td>DFHSZRDTE</td>
<td>CSECT</td>
<td>FEPI discard target processor</td>
<td>03</td>
</tr>
<tr>
<td>DFHSZRDREF</td>
<td>CSECT</td>
<td>FEPI request processor</td>
<td>03</td>
</tr>
<tr>
<td>DFHSZRDICF</td>
<td>CSECT</td>
<td>FEPI FREE completion processor</td>
<td>03</td>
</tr>
<tr>
<td>DFHSZRDICG</td>
<td>CSECT</td>
<td>FEPI Dispatcher work queue processor</td>
<td>03</td>
</tr>
<tr>
<td>DFHSZRDIAH</td>
<td>CSECT</td>
<td>FEPI allocate processor</td>
<td>03</td>
</tr>
<tr>
<td>DFHSZRDIFI</td>
<td>CSECT</td>
<td>FEPI install free processor</td>
<td>03</td>
</tr>
<tr>
<td>DFHSZRIDII</td>
<td>CSECT</td>
<td>FEPI install processor</td>
<td>03</td>
</tr>
<tr>
<td>DFHSZRIDII</td>
<td>CSECT</td>
<td>FEPI install node processor</td>
<td>03</td>
</tr>
<tr>
<td>DFHSZRIDIO</td>
<td>CSECT</td>
<td>FEPI ACB open processor</td>
<td>03</td>
</tr>
<tr>
<td>DFHSZRIDIP</td>
<td>CSECT</td>
<td>FEPI install pool processor</td>
<td>03</td>
</tr>
<tr>
<td>DFHSZRIQ</td>
<td>CSECT</td>
<td>FEPI inquire processor</td>
<td>03</td>
</tr>
<tr>
<td>DFHSZRVIS</td>
<td>CSECT</td>
<td>FEPI install processor</td>
<td>03</td>
</tr>
<tr>
<td>DFHSZRIAT</td>
<td>CSECT</td>
<td>FEPI install target processor</td>
<td>03</td>
</tr>
<tr>
<td>DFHSZRIWIN</td>
<td>CSECT</td>
<td>FEPI SET processor</td>
<td>03</td>
</tr>
<tr>
<td>DFHSZRN1</td>
<td>CSECT</td>
<td>FEPI NODE processor</td>
<td>03</td>
</tr>
<tr>
<td>DFHSZRN2</td>
<td>CSECT</td>
<td>FEPI NOOP processor</td>
<td>03</td>
</tr>
<tr>
<td>DFHSZRN3</td>
<td>CSECT</td>
<td>FEPI timer services</td>
<td>03</td>
</tr>
<tr>
<td>DFHSZRN4</td>
<td>CSECT</td>
<td>FEPI request preparation</td>
<td>03</td>
</tr>
<tr>
<td>DFHSZRN5</td>
<td>CSECT</td>
<td>FEPI queue for REQSESS processing</td>
<td>03</td>
</tr>
<tr>
<td>DFHSZRN6</td>
<td>CSECT</td>
<td>FEPI request queue processor</td>
<td>03</td>
</tr>
<tr>
<td>DFHSZRN7</td>
<td>CSECT</td>
<td>FEPI request release processor</td>
<td>03</td>
</tr>
<tr>
<td>DFHSZRN8</td>
<td>CSECT</td>
<td>FEPI connection processor</td>
<td>03</td>
</tr>
<tr>
<td>DFHSZDNEM</td>
<td>CSECT</td>
<td>FEPI recovery services</td>
<td>03</td>
</tr>
<tr>
<td>DFHSZDNEX</td>
<td>CSECT</td>
<td>FEPI EXTRACT processor</td>
<td>03</td>
</tr>
<tr>
<td>DFHSZRN22</td>
<td>CSECT</td>
<td>FEPI TERMINATE processor</td>
<td>03</td>
</tr>
<tr>
<td>DFHSZRN30</td>
<td>CSECT</td>
<td>FEPI storage control block</td>
<td>11</td>
</tr>
<tr>
<td>DFHSZRNIP</td>
<td>CSECT</td>
<td>FEPI initialization processor</td>
<td>03</td>
</tr>
<tr>
<td>DFHSZRN2F</td>
<td>CSECT</td>
<td>FEPI copy NIB mask to real NIB</td>
<td>03</td>
</tr>
<tr>
<td>DFHSZRNQ</td>
<td>CSECT</td>
<td>FEPI get queue element FIFO</td>
<td>03</td>
</tr>
<tr>
<td>DFHSZRQG</td>
<td>CSECT</td>
<td>FEPI REQSESS dispatcher</td>
<td>03</td>
</tr>
<tr>
<td>DFHSZRTA</td>
<td>CSECT</td>
<td>FEPI VTAM receive_any processor</td>
<td>03</td>
</tr>
<tr>
<td>DFHSZRTI</td>
<td>CSECT</td>
<td>FEPI VTAM receive_any issuer</td>
<td>03</td>
</tr>
<tr>
<td>DFHSZRTJ</td>
<td>CSECT</td>
<td>FEPI delayed bind processor</td>
<td>03</td>
</tr>
<tr>
<td>DFHSZRTK</td>
<td>CSECT</td>
<td>FEPI SETLOGON request issuer</td>
<td>03</td>
</tr>
<tr>
<td>DFHSZRTL</td>
<td>CSECT</td>
<td>FEPI VTAM feedback interpreter</td>
<td>03</td>
</tr>
<tr>
<td>DFHSZRTM</td>
<td>CSECT</td>
<td>FEPI VTAM feedback interpreter</td>
<td>03</td>
</tr>
<tr>
<td>DFHSZRTN</td>
<td>CSECT</td>
<td>FEPI VTAM feedback interpreter</td>
<td>03</td>
</tr>
<tr>
<td>DFHSZRTO</td>
<td>CSECT</td>
<td>FEPI RPL exit after SETLOGON</td>
<td>03</td>
</tr>
<tr>
<td>DFHSZRTP</td>
<td>CSECT</td>
<td>FEPI VTAM DFASY exit</td>
<td>03</td>
</tr>
<tr>
<td>DFHSZRTQ</td>
<td>CSECT</td>
<td>FEPI RPL exit to free request block</td>
<td>03</td>
</tr>
<tr>
<td>DFHSZRTL0</td>
<td>CSECT</td>
<td>FEPI VTAM logon exit</td>
<td>03</td>
</tr>
<tr>
<td>DFHSZT10</td>
<td>CSECT</td>
<td>FEPI VTAM LOSTERM (lost terminal) exit</td>
<td>03</td>
</tr>
<tr>
<td>DFHSZT11</td>
<td>CSECT</td>
<td>FEPI VTAM NSEXIT (network services) exit</td>
<td>03</td>
</tr>
<tr>
<td>DFHSZT12</td>
<td>CSECT</td>
<td>FEPI STIMER IRB exit routine</td>
<td>03</td>
</tr>
<tr>
<td>DFHSZT13</td>
<td>CSECT</td>
<td>FEPI VTAM RECEIVE ANY exit</td>
<td>03</td>
</tr>
<tr>
<td>DFHSZT14</td>
<td>CSECT</td>
<td>FEPI VTAM SCIP (session control) exit</td>
<td>03</td>
</tr>
<tr>
<td>DFHSZT15</td>
<td>CSECT</td>
<td>FEPI VTAM TPEND exit</td>
<td>03</td>
</tr>
<tr>
<td>DFHSZT16</td>
<td>CSECT</td>
<td>FEPI RPL exit following logon reject</td>
<td>03</td>
</tr>
</tbody>
</table>
### Table 123. CICS modules directory (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type Description Library</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHSZYQR</td>
<td>CSECT (OCO) FEPI post for REQSESS processing</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSZYR1</td>
<td>CSECT (OCO) FEPI VTAM RECEIVE_ANY issuer</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSZYSC</td>
<td>CSECT (OCO) FEPI VTAM SCIP exit extension</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSZYSR</td>
<td>CSECT (OCO) FEPI feedback interpreter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSZZAG</td>
<td>CSECT (OCO) FEPI get RECEIVE_ANY request block</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSZZFR</td>
<td>CSECT (OCO) FEPI free RECEIVE_ANY request block</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSZZNG</td>
<td>CSECT (OCO) FEPI get session control request block</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSZZRG</td>
<td>CSECT (OCO) FEPI get RPL request block</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSZZCP</td>
<td>CSECT (OCO) FEPI SLU2 flow controller</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSZZDX</td>
<td>CSECT (OCO) FEPI SLU2 drain completion exit</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSZZID</td>
<td>CSECT (OCO) FEPI SLU2 send data processor</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSZZIX</td>
<td>CSECT (OCO) FEPI SLU2 send completion exit</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSZZOA</td>
<td>CSECT (OCO) FEPI SLU2 send response processor</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSZZOD</td>
<td>CSECT (OCO) FEPI SLU2 receive data processor</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSZZOR</td>
<td>CSECT (OCO) FEPI SLU2 receive processor</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSZZOX</td>
<td>CSECT (OCO) FEPI SLU2 receive specific completion exit</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSZZOY</td>
<td>CSECT (OCO) FEPI SLU2 receive specific action module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSZZPX</td>
<td>CSECT (OCO) FEPI SLU2 positive response drain exit</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSZZQS</td>
<td>CSECT (OCO) FEPI SLU2 REQSESS issuer</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSZZQX</td>
<td>CSECT (OCO) FEPI SLU2 REQSESS exit</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSZZSB</td>
<td>CSECT (OCO) FEPI SLU2 bind processor</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSZZSC</td>
<td>CSECT (OCO) FEPI SLU2 session controller</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSZZSD</td>
<td>CSECT (OCO) FEPI SLU2 SDT processor</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSZZSH</td>
<td>CSECT (OCO) FEPI SLU2 SHUTC processor</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSZZSQ</td>
<td>CSECT (OCO) FEPI SLU2 QC processor</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSZZSR</td>
<td>CSECT (OCO) FEPI SLU2 RESETSR processor</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSZZSX</td>
<td>CSECT (OCO) FEPI SLU2 OPNSEC processor</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHSZZTE</td>
<td>CSECT (OCO) FEPI SLU2 TERMSESS processor</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTACB</td>
<td>Macro Task abend control block</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHTACLE</td>
<td>DSECT TCT line entry prefix</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHTACP</td>
<td>CSECT Terminal abnormal condition program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHTAJP</td>
<td>CSECT Time adjustment program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHTBS</td>
<td>Macro Builder interface</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHTBSB</td>
<td>CSECT Add a node</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHTBSBP</td>
<td>CSECT Recursive part of DFHTBSB</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHTBSD</td>
<td>CSECT Delete node program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHTBSDP</td>
<td>CSECT Recursive part of DFHTBSD</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHTBSL</td>
<td>CSECT Create recovery record for node</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHTBSLP</td>
<td>CSECT Recursive part of DFHTBSL</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHTBSQ</td>
<td>CSECT Builder inquire process</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHTBSQP</td>
<td>CSECT Recursive part of DFHTBSQ</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHTBSR</td>
<td>CSECT Builder restore process</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHTBSRP</td>
<td>CSECT Recursive part of DFHTBSR</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHTBS</td>
<td>CSECT TBS syncpoint processor</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHTBST</td>
<td>DSECT TBS translate tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHTBSO</td>
<td>CSECT Table builder services program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHTC</td>
<td>Macro Terminal service request</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHTCA</td>
<td>Macro Task control area</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHTCADS</td>
<td>DSECT Task control area</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHTCAM</td>
<td>Source CICS-TCAM interface logic</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHTCLL</td>
<td>Source Common line control logic</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHTCOM</td>
<td>Source Input data length computation</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHTCSS</td>
<td>Source Start-stop event analysis</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHTCDEF</td>
<td>Symbolic Terminal control definitions</td>
<td>OS -</td>
</tr>
</tbody>
</table>
### Table 123. CICS modules directory (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHTCDPF</td>
<td>CSECT (OCO)</td>
<td>Terminal control prefix SDUMP module</td>
<td>03</td>
</tr>
<tr>
<td>DFHTCDOF</td>
<td>CSECT (OCO)</td>
<td>Terminal control SDUMP formatter</td>
<td>03</td>
</tr>
<tr>
<td>DFHTCORS</td>
<td>Source</td>
<td>Terminal storage routine</td>
<td>0S</td>
</tr>
<tr>
<td>DFHTCOP</td>
<td>CSECT</td>
<td>Terminal control program</td>
<td>0S 03</td>
</tr>
<tr>
<td>DFHTCPC</td>
<td>Macro</td>
<td>DFHZCP request</td>
<td>0S</td>
</tr>
<tr>
<td>DFHTCPM</td>
<td>Macro</td>
<td>Common ZCP functions</td>
<td>11</td>
</tr>
<tr>
<td>DFHTCPPL</td>
<td>Macro</td>
<td>LU6.2 limited resources service</td>
<td>0S</td>
</tr>
<tr>
<td>DFHTCPQR</td>
<td>Macro</td>
<td>Queued response notification</td>
<td>0S</td>
</tr>
<tr>
<td>DFHTCPRA</td>
<td>DSECT</td>
<td>Receive-any control element</td>
<td>0S</td>
</tr>
<tr>
<td>DFHTCPRT</td>
<td>Macro</td>
<td>DFHZCP RETURN macro</td>
<td>0S</td>
</tr>
<tr>
<td>DFHTCPRM</td>
<td>Macro</td>
<td>TCT generation - VTAM DSECTs</td>
<td>11</td>
</tr>
<tr>
<td>DFHTCPSV</td>
<td>Macro</td>
<td>DFHZCP SAVE macro</td>
<td>0S</td>
</tr>
<tr>
<td>DFHTCPZR</td>
<td>Macro</td>
<td>VTAM RPL extension for HPO</td>
<td>11</td>
</tr>
<tr>
<td>DFHTCQ</td>
<td>Macro</td>
<td>DFHZCP QUEUE macro</td>
<td>0S</td>
</tr>
<tr>
<td>DFHTCRP</td>
<td>CSECT</td>
<td>Terminal control recovery program</td>
<td>0S 03</td>
</tr>
<tr>
<td>DFHTCRPC</td>
<td>CSECT</td>
<td>XRF tracking interface for TCT contents</td>
<td>0S 03</td>
</tr>
<tr>
<td>DFHTCRPL</td>
<td>CSECT</td>
<td>Install TCT macro definitions</td>
<td>0S 03</td>
</tr>
<tr>
<td>DFHTCROPS</td>
<td>CSECT</td>
<td>XRF tracking interface for ZCP sessions</td>
<td>0S 03</td>
</tr>
<tr>
<td>DFHTCRCPU</td>
<td>CSECT</td>
<td>XRF tracking interface for SNTTEs</td>
<td>0S 03</td>
</tr>
<tr>
<td>DFHTCRCWE</td>
<td>DSECT</td>
<td>Remote install work element</td>
<td>0S</td>
</tr>
<tr>
<td>DFHTCSTAM</td>
<td>Source</td>
<td>Sequential terminal logic</td>
<td>0S</td>
</tr>
<tr>
<td>DFHTCSRV</td>
<td>Macro</td>
<td>DFHTC inner service macro</td>
<td>11</td>
</tr>
<tr>
<td>DFHTCSUM</td>
<td>CSECT</td>
<td>Terminal control dump summary program</td>
<td>0S</td>
</tr>
<tr>
<td>DFHTCT</td>
<td>Macro</td>
<td>Terminal control table</td>
<td>11</td>
</tr>
<tr>
<td>DFHTCTDY</td>
<td>CSECT</td>
<td>Terminal control table (dummy)</td>
<td>19 03</td>
</tr>
<tr>
<td>DFHTCFTN</td>
<td>Source</td>
<td>TCT TYPE=FINAL (VTAM)</td>
<td>11</td>
</tr>
<tr>
<td>DFHTCFTF</td>
<td>DSECT</td>
<td>TCT prefix</td>
<td>11</td>
</tr>
<tr>
<td>DFHTCGL</td>
<td>Source</td>
<td>Terminal control task initiation logic</td>
<td>0S</td>
</tr>
<tr>
<td>DFHTCTLC</td>
<td>Macro</td>
<td>TCT inner macro</td>
<td>11</td>
</tr>
<tr>
<td>DFHTCTLE</td>
<td>DSECT</td>
<td>TCT line entry</td>
<td>11</td>
</tr>
<tr>
<td>DFHTCTME</td>
<td>Macro</td>
<td>Generate TCT mode group entries</td>
<td>11</td>
</tr>
<tr>
<td>DFHTCTPR</td>
<td>Macro</td>
<td>TCTTE partition extension builder</td>
<td>11</td>
</tr>
<tr>
<td>DFHTCTPS</td>
<td>Macro</td>
<td>TCT inner macro</td>
<td>11</td>
</tr>
<tr>
<td>DFHTCTPX</td>
<td>Macro</td>
<td>TCT inner macro</td>
<td>11</td>
</tr>
<tr>
<td>DFHTCTRD</td>
<td>Macro</td>
<td>VTAM RDO command list builder</td>
<td>11</td>
</tr>
<tr>
<td>DFHTCETE</td>
<td>Macro</td>
<td>TCT definition macro</td>
<td>11</td>
</tr>
<tr>
<td>DFHTCTRN</td>
<td>Source</td>
<td>Terminal control translation tables</td>
<td>0S</td>
</tr>
<tr>
<td>DFHTCTSA</td>
<td>Macro</td>
<td>TCT inner macro</td>
<td>11</td>
</tr>
<tr>
<td>DFHTCTSB</td>
<td>Macro</td>
<td>TCT inner macro</td>
<td>11</td>
</tr>
<tr>
<td>DFHTCTSE</td>
<td>Macro</td>
<td>Generate ISC system entry</td>
<td>11</td>
</tr>
<tr>
<td>DFHTCTSK</td>
<td>Macro</td>
<td>Generate TCT skeleton entry</td>
<td>11</td>
</tr>
<tr>
<td>DFHTCTST</td>
<td>Macro</td>
<td>TCT inner macro</td>
<td>11</td>
</tr>
<tr>
<td>DFHTCTSV</td>
<td>Macro</td>
<td>TCT inner macro</td>
<td>11</td>
</tr>
<tr>
<td>DFHTCTTE</td>
<td>DSECT</td>
<td>TCT terminal entry</td>
<td>11</td>
</tr>
<tr>
<td>DFHTCTUA</td>
<td>Macro</td>
<td>TCT inner macro</td>
<td>11</td>
</tr>
<tr>
<td>DFHTCTUB</td>
<td>Macro</td>
<td>TCT inner macro</td>
<td>11</td>
</tr>
<tr>
<td>DFHTCTWA</td>
<td>DSECT</td>
<td>TCT transaction work area</td>
<td>11</td>
</tr>
<tr>
<td>DFHTCTWE</td>
<td>DSECT</td>
<td>TCT autodefine work element</td>
<td>0S</td>
</tr>
<tr>
<td>DFHTCTZE</td>
<td>Macro</td>
<td>TCTTE definition</td>
<td>11</td>
</tr>
<tr>
<td>DFHTCTZG</td>
<td>Sample</td>
<td>Terminal control table</td>
<td>19 03</td>
</tr>
<tr>
<td>DFHTCUDS</td>
<td>DSECT</td>
<td>COMMAREA passed to autoinstall exit</td>
<td>11</td>
</tr>
<tr>
<td>DFHTCUDS</td>
<td>DSECT</td>
<td>COMMAREA passed to autoinstall exit</td>
<td>C2 07</td>
</tr>
<tr>
<td>DFHTCUDS</td>
<td>DSECT</td>
<td>COMMAREA passed to autoinstall exit</td>
<td>P2 08</td>
</tr>
<tr>
<td>DFHTCVC29</td>
<td>DSECT</td>
<td>XRF session state data control vector</td>
<td>0S</td>
</tr>
<tr>
<td>DFHTCXX</td>
<td>Macro</td>
<td>TCA extension for LU6.2</td>
<td>11</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
<td>Library</td>
</tr>
<tr>
<td>-------------</td>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>DFHTCXDF</td>
<td>CSECT</td>
<td>DU domain - transaction dump formatter for terminal related areas</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHTD</td>
<td>Macro</td>
<td>Transient data service request</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHTDA</td>
<td>CSECT</td>
<td>Transient data request processor</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTDB</td>
<td>CSECT</td>
<td>Transient data request processor</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTDCI</td>
<td>DSECT</td>
<td>Transient data VSAM CI map</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHTDDUF</td>
<td>CSECT (OCO)</td>
<td>Transient data SDUMP formatter exit routine</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHTDEXL</td>
<td>CSECT</td>
<td>Transient data DCB exit list and DCB abend exit routine</td>
<td></td>
</tr>
<tr>
<td>DFHTDGDS</td>
<td>DSECT</td>
<td>Transaction dump global statistics</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHTDGDS</td>
<td>DSECT</td>
<td>Transaction dump global statistics</td>
<td>C2 07</td>
</tr>
<tr>
<td>DFHTDOA</td>
<td>DSECT</td>
<td>Transient data output area</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHTDOA</td>
<td>DSECT</td>
<td>Transient data output area</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHTDOCA</td>
<td>DSECT</td>
<td>TDOC parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHTDOCM</td>
<td>Macro</td>
<td>TDOC request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHTDOCT</td>
<td>CSECT</td>
<td>TDOC trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTDORDS</td>
<td>DSECT</td>
<td>Transaction dump statistics by dump code</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHTDORDS</td>
<td>DSECT</td>
<td>Transaction dump statistics by dump code</td>
<td>C2 07</td>
</tr>
<tr>
<td>DFHTDORDS</td>
<td>DSECT</td>
<td>Transaction dump statistics by dump code</td>
<td>P2 -</td>
</tr>
<tr>
<td>DFHTDRP</td>
<td>DSECT</td>
<td>Transient data recovery program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHTDSDS</td>
<td>DSECT</td>
<td>Transient data static storage</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHTDTDA</td>
<td>DSECT</td>
<td>TDTD parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHTTDM</td>
<td>CSECT</td>
<td>TTD request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHTDTDT</td>
<td>CSECT</td>
<td>TTD trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTDTM</td>
<td>CSECT</td>
<td>Transient data table management gate</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTDTMA</td>
<td>CSECT</td>
<td>TDTM parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHTDTMM</td>
<td>Macro</td>
<td>TDTM request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHTDTMM</td>
<td>DSECT</td>
<td>TDTM translate tables</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTDTM</td>
<td>DSECT</td>
<td>TDTM translate tables</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTDTRI</td>
<td>CSECT</td>
<td>Transient data trace interpreter</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHTDUEED</td>
<td>Macro</td>
<td>TD user exits EXEC argument list</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHTDTRX</td>
<td>CSECT</td>
<td>Transient data phase 1 initialization</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHTDMX</td>
<td>CSECT (OCO)</td>
<td>XM domain - TD facility management services</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHTDMX</td>
<td>DSECT</td>
<td>TDXM parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHTDMX</td>
<td>Macro</td>
<td>TDXM request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHTDMX</td>
<td>DSECT (OCO)</td>
<td>TDXM trace interpretation data</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHTEPA</td>
<td>Macro</td>
<td>TEP inner macro</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHTEPC</td>
<td>Macro</td>
<td>TEP inner macro</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHTEPCA</td>
<td>Macro</td>
<td>TEP communication area</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHTEPM</td>
<td>Macro</td>
<td>TEP module generator</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHTEPS</td>
<td>Macro</td>
<td>TEP inner macro</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHTEPT</td>
<td>Macro</td>
<td>TEP table generator</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHTERID</td>
<td>Symbolic</td>
<td>Terminal error definitions</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHTTEST</td>
<td>Macro</td>
<td>Domain call argument TEST macro</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHTFALA</td>
<td>DSECT</td>
<td>TFAL parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHTFALM</td>
<td>Macro</td>
<td>TFAL request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHTFA</td>
<td>CSECT (OCO)</td>
<td>TFAL trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTFBA</td>
<td>DSECT</td>
<td>TFBF parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHTFBFM</td>
<td>Macro</td>
<td>TFBF request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHTFBF</td>
<td>CSECT (OCO)</td>
<td>TFBF trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTFFIQ</td>
<td>CSECT (OCO)</td>
<td>Terminal facility manager inquire/set functions</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTFQA</td>
<td>DSECT</td>
<td>TFIQ parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHTFQIQ</td>
<td>DSECT</td>
<td>TFIQ requests (inline form)</td>
<td>OS -</td>
</tr>
</tbody>
</table>
**CICS directory**

Table 123. CICS modules directory (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHTFQM</td>
<td>DSECT</td>
<td>TFIQ requests</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHTFIQT</td>
<td>CSECT (OCO)</td>
<td>TFIQ trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTFP</td>
<td>CSECT</td>
<td>Transaction failure program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHTFRF</td>
<td>CSECT (OCO)</td>
<td>Terminal facility manager release function</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTFRFT</td>
<td>CSECT (OCO)</td>
<td>TFRF trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTFTRI</td>
<td>CSECT (OCO)</td>
<td>Terminal facility manager trace interpreter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTFXM</td>
<td>CSECT</td>
<td>TF XM transaction attach</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTIDM</td>
<td>CSECT (OCO)</td>
<td>TI domain - initialization/termination</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTIDUF</td>
<td>CSECT (OCO)</td>
<td>SDUMP formatter for TI domain</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTIEDS</td>
<td>DSECT</td>
<td>Task interface element</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHTIEM</td>
<td>CSECT</td>
<td>Resource manager interface TIE manager</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHTIOA</td>
<td>DSECT</td>
<td>Terminal input/output area</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHTIOA</td>
<td>DSECT</td>
<td>Terminal input/output area</td>
<td>C2 07</td>
</tr>
<tr>
<td>DFHTISRI</td>
<td>CSECT (OCO)</td>
<td>TI domain - services</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTISRA</td>
<td>DSECT</td>
<td>TISR parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHTISRM</td>
<td>Macro</td>
<td>TISR request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHTISRT</td>
<td>CSECT</td>
<td>TISR trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTITRI</td>
<td>CSECT (OCO)</td>
<td>Trace interpreter for TI domain</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTLT</td>
<td>Macro</td>
<td>Terminal list table</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHTM</td>
<td>Macro</td>
<td>Table manager interface</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHTMDUF</td>
<td>CSECT (OCO)</td>
<td>Table manager SDUMP formatter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTMP01</td>
<td>CSECT (OCO)</td>
<td>Table manager program - part 1</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTMP02</td>
<td>CSECT (OCO)</td>
<td>Table manager program - part 2</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTMTRI</td>
<td>CSECT (OCO)</td>
<td>Table manager program trace interpreter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTOACN</td>
<td>CSECT</td>
<td>Terminal object resolution (TOR) - add connection</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHTOAPT</td>
<td>CSECT</td>
<td>TOR - add pooled terminal</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTOASE</td>
<td>CSECT</td>
<td>TOR - add session</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHTOATM</td>
<td>CSECT</td>
<td>TOR - add (non-pooled) terminal</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTOATY</td>
<td>CSECT</td>
<td>TOR - add typeterm</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTOBPS</td>
<td>CSECT</td>
<td>TOR - create BPS and check attributes</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHTOCAN</td>
<td>CSECT</td>
<td>TOR - dynamic backout processing</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTOCMT</td>
<td>CSECT</td>
<td>TOR - syncpoint commit processing</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTOLCR</td>
<td>CSECT</td>
<td>TOR - end logical unit of complex - replacement</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTOLUI</td>
<td>CSECT</td>
<td>TOR - end logical unit of installation</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTOM</td>
<td>Macro</td>
<td>BMS terminal output</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHTON</td>
<td>CSECT</td>
<td>Terminal object resolution module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTONR</td>
<td>CSECT</td>
<td>Terminal object resolution recovery</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTONRT</td>
<td>DSECT</td>
<td>TONR translate tables</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTORP</td>
<td>CSECT</td>
<td>Terminal object recovery program</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTOR00</td>
<td>CSECT</td>
<td>Terminal object resolution program (DFHTOR)</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHTOUT1</td>
<td>CSECT</td>
<td>TOR - set operation utilities</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTOUT2</td>
<td>CSECT</td>
<td>TOR - map operation utilities</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTPE</td>
<td>DSECT</td>
<td>Terminal partition extension</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHTPP</td>
<td>CSECT</td>
<td>BMS terminal page processor</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHTPPA0</td>
<td>CSECT</td>
<td>BMS terminal page processor (standard)</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHTPPA10</td>
<td>CSECT</td>
<td>BMS terminal page processor (full)</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHTPO</td>
<td>CSECT</td>
<td>BMS terminal page cleanup program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHTQGDS</td>
<td>CSECT</td>
<td>Global statistics for Transient Data</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHTQGDS</td>
<td>CSECT</td>
<td>Global statistics for Transient Data</td>
<td>C2 07</td>
</tr>
<tr>
<td>DFHTQDRS</td>
<td>CSECT</td>
<td>Transient data queue statistics</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHTQDRS</td>
<td>CSECT</td>
<td>Transient data queue statistics</td>
<td>C2 07</td>
</tr>
</tbody>
</table>
### Table 123. CICS modules directory (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHTPR</td>
<td>CSECT</td>
<td>BMS terminal page retrieval program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHTPS</td>
<td>CSECT</td>
<td>BMS terminal page scheduling program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHTR</td>
<td>Macro</td>
<td>Trace service request</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHTRA</td>
<td>DSECT</td>
<td>TR domain - anchor block</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHTTRACE</td>
<td>Macro</td>
<td>Trace system macro</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHTRADS</td>
<td>DSECT</td>
<td>TR domain - parameter list to DFHTRAP</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHTRAO</td>
<td>DSECT</td>
<td>TR domain - auxiliary trace output</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHTRAP</td>
<td>CSECT</td>
<td>TR domain - FE global trap/trace exit</td>
<td>11 03</td>
</tr>
<tr>
<td>DFHTRBL</td>
<td>DSECT</td>
<td>TR domain - internal trace table block</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHTRCIF</td>
<td>CSECT</td>
<td>CZ Direct-to-CICS</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTRDM</td>
<td>CSECT</td>
<td>TR domain - initialization/termination</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHTRDS</td>
<td>DSECT</td>
<td>TR domain - control blocks</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHTRDU</td>
<td>DSECT</td>
<td>TR &amp; DU keyword copybook</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHTRDF</td>
<td>CSECT</td>
<td>(OCO) SDUMP formatter for TR domain</td>
<td>OS - 03</td>
</tr>
<tr>
<td>DFHTREN</td>
<td>DSECT</td>
<td>TR domain - trace entry</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHTREX</td>
<td>DSECT</td>
<td>Offline trace formatting control area</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHTRFCA</td>
<td>DSECT</td>
<td>Offline trace formatting - format data fields</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHTRFFE</td>
<td>CSECT</td>
<td>Offline trace formatting - format trace entry</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHTRFPB</td>
<td>CSECT</td>
<td>Offline trace formatting - process block</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHTRFPP</td>
<td>CSECT</td>
<td>Offline trace formatting - process selective print parameters</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHTRT</td>
<td>CSECT</td>
<td>Trace put routine for features</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHTRTA</td>
<td>CSECT</td>
<td>TRFT parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHTRTD</td>
<td>CSECT</td>
<td>TR feature trace entry header</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHTRTFM</td>
<td>Macro</td>
<td>TRFT macro</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHTRTT</td>
<td>CSECT</td>
<td>TRFT translate tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHTRTX</td>
<td>Macro</td>
<td>TRFT macro</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHTRTFY</td>
<td>Macro</td>
<td>TRFT call structured parameter list</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHTTRB</td>
<td>CSECT</td>
<td>Trace interpretation string builder</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHTRTP</td>
<td>CSECT</td>
<td>Trace control program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHTRPRA</td>
<td>CSECT</td>
<td>Auxiliary trace offline formatting</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHTRPGR</td>
<td>CSECT</td>
<td>GTF trace offline formatting</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHTRPT</td>
<td>CSECT</td>
<td>TR domain - trace put (all destinations)</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHTRPTA</td>
<td>DSECT</td>
<td>TRPT parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHTRPMT</td>
<td>Macro</td>
<td>TRPT request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHTRPPT</td>
<td>CSECT</td>
<td>TRPT trace interpretation data</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHTRPTX</td>
<td>Macro</td>
<td>TRPT request (XPI)</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHTRPFY</td>
<td>DSECT</td>
<td>TRPT parameter list (XPI)</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHTRPX</td>
<td>CSECT</td>
<td>TR domain - trace put (fast path)</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHTRSR</td>
<td>CSECT</td>
<td>TR domain - trace destination services</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHTRSRA</td>
<td>DSECT</td>
<td>TRSR parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHTRSRM</td>
<td>Macro</td>
<td>TRSR request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHTRSRT</td>
<td>CSECT</td>
<td>TRSR trace interpretation data</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHTRSU</td>
<td>CSECT</td>
<td>TR domain - subroutines</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHTRSUA</td>
<td>DSECT</td>
<td>TRSU parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHTRSUM</td>
<td>Macro</td>
<td>TRSU request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHTRSTU</td>
<td>CSECT</td>
<td>TRSU trace interpretation data</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHTRTRE</td>
<td>CSECT</td>
<td>Trace interpreter for TR domain</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHTRTST</td>
<td>Macro</td>
<td>TR domain - test if trace point active</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHTRUDS</td>
<td>DSECT</td>
<td>TRUE 24-bit parameter list save area</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHTRXDF</td>
<td>CSECT</td>
<td>DU domain - transaction dump formatter for internal trace table</td>
<td>OS 03</td>
</tr>
</tbody>
</table>
### CICS directory

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHTRZCP</td>
<td>CSECT</td>
<td>Terminal object builder</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHTRZIP</td>
<td>CSECT</td>
<td>Session object builder</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHTRZPP</td>
<td>CSECT</td>
<td>Pool object builder</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHTRXPP</td>
<td>CSECT</td>
<td>Connection object builder</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHTRYPP</td>
<td>CSECT</td>
<td>Typeterm object builder</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHTRZPP</td>
<td>CSECT</td>
<td>Terminal object matching</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHTS</td>
<td>Macro</td>
<td>Temporary-storage service request</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHTSAD</td>
<td>CSECT</td>
<td>TS Domain - TSAD Gate Function</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTSADT</td>
<td>CSECT</td>
<td></td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTSAM</td>
<td>CSECT</td>
<td>TS auxiliary manager functions subroutine</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTSAMT</td>
<td>DSECT</td>
<td>TSAM translate tables</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTSBR</td>
<td>CSECT</td>
<td>TS browse functions</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTSBRAT</td>
<td>CSECT</td>
<td>TSBR parameter list</td>
<td>03 -</td>
</tr>
<tr>
<td>DFHTSBRM</td>
<td>Macro</td>
<td>TSBR request</td>
<td>03 -</td>
</tr>
<tr>
<td>DFHTSBRT</td>
<td>DSECT</td>
<td>TSBR translate tables</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTSDM</td>
<td>CSECT</td>
<td>TS domain manager functions (initialize, quiesce, terminate)</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTSDQ</td>
<td>CSECT</td>
<td>Temporary Storage Delete Queue</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTSDUC</td>
<td>CSECT</td>
<td>Temporary-storage SDUMP analysis</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTSDFU</td>
<td>CSECT</td>
<td>Temporary-storage SDUMP formatter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTSDUS</td>
<td>CSECT</td>
<td>Temporary-storage SDUMP summary</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTSGDS</td>
<td>DSECT</td>
<td>Temporary-storage statistics DSECT (Assembler)</td>
<td>11 -</td>
</tr>
<tr>
<td></td>
<td>DSECT</td>
<td>Temporary-storage statistics DSECT (COBOL)</td>
<td>C2 07</td>
</tr>
<tr>
<td>DFHTSHD</td>
<td>Macro</td>
<td>Temporary-storage input/output area header</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHTSIOA</td>
<td>DSECT</td>
<td>Temporary-storage input/output area</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHTSICT</td>
<td>CSECT</td>
<td>TSIC translate tables</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTSITR</td>
<td>CSECT</td>
<td>TS trace interpretation</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTSMB</td>
<td>CSECT</td>
<td>DSMBMB Design</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTSMBT</td>
<td>CSECT</td>
<td></td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTSPT</td>
<td>CSECT</td>
<td>Temporary-storage control program</td>
<td>03 -</td>
</tr>
<tr>
<td>DFHTSPTA</td>
<td>CSECT</td>
<td>TSPT request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHTSPTM</td>
<td>Macro</td>
<td>TSPT request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHTSPTT</td>
<td>DSECT</td>
<td>TSPT translate tables</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTSQR</td>
<td>CSECT</td>
<td>TS mainline queue request functions</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTSQRT</td>
<td>DSECT</td>
<td>TSQR translate tables</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTSRM</td>
<td>CSECT</td>
<td>TS recovery manager functions</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTSSBT</td>
<td>DSECT</td>
<td>TSSB translate tables</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTSSH</td>
<td>CSECT</td>
<td>TS shared TS functions</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTSSHT</td>
<td>DSECT</td>
<td>TSSH translate tables</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTSSR</td>
<td>CSECT</td>
<td>TS service functions (inquire, set)</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTSSRT</td>
<td>DSECT</td>
<td>TSSR translate tables</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTSSST</td>
<td>CSECT</td>
<td>TS statistics functions</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTST</td>
<td>Macro</td>
<td>Temporary-storage table</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHTSTDS</td>
<td>DSECT</td>
<td>Temporary-storage table</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHTSTUEQ</td>
<td>CSECT</td>
<td>XTSESEQ and XTSEQ EXEC parameter lists</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHTSTUT</td>
<td>DSECT</td>
<td>TSUT abstract type internal control blocks</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHTSTUTI</td>
<td>Macro</td>
<td>TSUT abstract type inline functions</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHTSTWQ</td>
<td>CSECT</td>
<td>TS wait queue functions subroutine</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTSTWQT</td>
<td>DSECT</td>
<td>TSWQ translate tables</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHTTPDS</td>
<td>DSECT</td>
<td>BMS - terminal type parameter</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHTUL</td>
<td>DSECT</td>
<td>Standard-labeled tape user labels</td>
<td>- -</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
<td>Library</td>
</tr>
<tr>
<td>---------------</td>
<td>---------------</td>
<td>------------------------------------------------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>DFHTUTEN</td>
<td>Macro</td>
<td>Trace table generation macro</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHUCNV</td>
<td>Sample</td>
<td>CICS OS/2 user data conversion program</td>
<td>19 03</td>
</tr>
<tr>
<td>DFHUEDUF</td>
<td>CSECT (OCO)</td>
<td>User exit SDUMP formatter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHUEDFS</td>
<td>DSECT</td>
<td>File control user exit file/data set info</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHUEH</td>
<td>CSECT</td>
<td>User exit handler (AP domain)</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHUHJC</td>
<td>Source</td>
<td>User exit program invocation</td>
<td>-</td>
</tr>
<tr>
<td>DFHUHJWA</td>
<td>DSECT</td>
<td>User exit work areas</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHUHJEQ</td>
<td>CSECT</td>
<td>User exit inquire exitprogram function</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHUHJQT</td>
<td>CSECT</td>
<td>EIQT trace interpreter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHUHJUX</td>
<td>DSECT</td>
<td>User exit resource manager</td>
<td>-</td>
</tr>
<tr>
<td>DFHUHJXET</td>
<td>DSECT</td>
<td>User exit table entry</td>
<td>-</td>
</tr>
<tr>
<td>DFHUHJXETD</td>
<td>DSECT</td>
<td>User exit table header</td>
<td>-</td>
</tr>
<tr>
<td>DFHUHJXETM</td>
<td>Macro</td>
<td>User-exit-dependent code generator</td>
<td>-</td>
</tr>
<tr>
<td>DFHUHJXETPT</td>
<td>Macro</td>
<td>User exit point definition</td>
<td>-</td>
</tr>
<tr>
<td>DFHUHJXETB</td>
<td>DSECT</td>
<td>Assembler DSECT for User interface block</td>
<td>-</td>
</tr>
<tr>
<td>DFHUHJXETBIC</td>
<td>CSECT</td>
<td>C structure of the UIB</td>
<td>- 08</td>
</tr>
<tr>
<td>DFHUHJXETIBO</td>
<td>CSECT</td>
<td>Cobol structure of the UIB</td>
<td>C2 07</td>
</tr>
<tr>
<td>DFHUHJXETIBP</td>
<td>CSECT</td>
<td>PLI structure of the UIB</td>
<td>P2 17</td>
</tr>
<tr>
<td>DFHUHJXETPDS</td>
<td>Other</td>
<td>Cataloged procedure to update a temporary library during system generation</td>
<td>18 -</td>
</tr>
<tr>
<td>DFHUHURLDS</td>
<td>DSECT</td>
<td>BMS - user-supplied route list</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHUHURLDS</td>
<td>DSECT</td>
<td>BMS - user-supplied route list</td>
<td>C2 07</td>
</tr>
<tr>
<td>DFHUHURLDS</td>
<td>DSECT</td>
<td>BMS - user-supplied route list</td>
<td>D2 08</td>
</tr>
<tr>
<td>DFHUHUSBAD</td>
<td>CSECT (OCO)</td>
<td>US domain - Add, Delete and Inquire User</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHUHUSBADA</td>
<td>DSECT</td>
<td>USAD parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHUHUSBADM</td>
<td>Macro</td>
<td>USAD request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHUHUSBADT</td>
<td>CSECT (OCO)</td>
<td>USAD trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHUHUSBAGE</td>
<td>Macro</td>
<td>Usage pricing code generation macro</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHUHUSBAND</td>
<td>CSECT (OCO)</td>
<td>US domain - anchor block</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHUHUSBAP</td>
<td>CSECT</td>
<td>User backout program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHUHUSDE</td>
<td>CSECT</td>
<td>US domain - delete user DCE references</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHUHUSDNT</td>
<td>DSECT</td>
<td>USDE translate tables</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHUHUSDM</td>
<td>CSECT (OCO)</td>
<td>US domain - initialize, quiesce, and terminate domain functions</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHUHUSDUF</td>
<td>CSECT (OCO)</td>
<td>US domain - dump formatter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHUHUSFL</td>
<td>CSECT (OCO)</td>
<td>US domain - Flatten and unflatten user</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHUHUSFLA</td>
<td>DSECT</td>
<td>USFL parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHUHUSFLM</td>
<td>Macro</td>
<td>USFL request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHUHUSFLT</td>
<td>CSECT (OCO)</td>
<td>USFL trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHUHUSGDS</td>
<td>DSECT</td>
<td>US domain - global statistics</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHUHUSGDS</td>
<td>DSECT</td>
<td>US domain - global statistics</td>
<td>C2 07</td>
</tr>
<tr>
<td>DFHUHUSIS</td>
<td>CSECT (OCO)</td>
<td>US domain - inquire and set functions</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHUHUSISA</td>
<td>DSECT</td>
<td>USIS parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHUHUSIS</td>
<td>Macro</td>
<td>USIS request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHUHUSIST</td>
<td>CSECT (OCO)</td>
<td>USIS trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHUHUSST</td>
<td>CSECT (OCO)</td>
<td>US domain - statistics</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHUHUSTI</td>
<td>CSECT (OCO)</td>
<td>US domain - timeout handler</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHUHUSTA</td>
<td>DSECT</td>
<td>USTI parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHUHUSTIM</td>
<td>Macro</td>
<td>USTI request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHUHUSTIT</td>
<td>CSECT (OCO)</td>
<td>USTI trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHUHUSTI</td>
<td>CSECT (OCO)</td>
<td>US domain - trace formatter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHUHUXM</td>
<td>CSECT (OCO)</td>
<td>US domain - transaction support</td>
<td>- 03</td>
</tr>
</tbody>
</table>
### CICS directory

**Table 123. CICS modules directory (continued)**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHUSXMA</td>
<td>DSECT</td>
<td>USXM parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHUSXMI</td>
<td>Macro</td>
<td>USXM request (inline version of DFHUSXMM)</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHUSXMM</td>
<td>Macro</td>
<td>USXM request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHUSXMT</td>
<td>CSECT</td>
<td>USXM trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHT64</td>
<td>CSECT</td>
<td>RU Base64 encoding and decoding</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHV</td>
<td>Macro</td>
<td>Version/modification level generator</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHVSA</td>
<td>DSECT</td>
<td>VSAM work area</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHVTA</td>
<td>DSECT</td>
<td>NACP LIFO storage definition</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHWBA</td>
<td>CSECT</td>
<td>Web module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHWBADX</td>
<td>CSECT</td>
<td>Web module</td>
<td>19 03</td>
</tr>
<tr>
<td>DFHWBAHX</td>
<td>CSECT</td>
<td>Web module</td>
<td>- 19</td>
</tr>
<tr>
<td>DFHWBALX</td>
<td>CSECT</td>
<td>Web module</td>
<td>- 19</td>
</tr>
<tr>
<td>DFHWBAOX</td>
<td>CSECT</td>
<td>Web module</td>
<td>- 19</td>
</tr>
<tr>
<td>DFHWBAP</td>
<td>CSECT</td>
<td>WB Domain WBAP Gate Functions</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHWBAPF</td>
<td>CSECT</td>
<td>Web Module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHWBAPT</td>
<td>CSECT</td>
<td>Web Module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHWBAPQ</td>
<td>CSECT</td>
<td>Web module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHWBA1</td>
<td>CSECT</td>
<td>Web module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHWBA1D</td>
<td>CSECT</td>
<td>Web module</td>
<td>- 11</td>
</tr>
<tr>
<td>DFHWBA1H</td>
<td>CSECT</td>
<td>Web module</td>
<td>- 08</td>
</tr>
<tr>
<td>DFHWBA1L</td>
<td>CSECT</td>
<td>Web module</td>
<td>- 17</td>
</tr>
<tr>
<td>DFHWBA1O</td>
<td>CSECT</td>
<td>Web module</td>
<td>- 07</td>
</tr>
<tr>
<td>DFHWBBL1</td>
<td>CSECT</td>
<td>Business Logic interface program</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHWBLL</td>
<td>CSECT</td>
<td></td>
<td>- 17</td>
</tr>
<tr>
<td>DFHWBMS</td>
<td>CSECT</td>
<td>WB Web Interface BMS Support</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHWBCDD</td>
<td>CSECT</td>
<td>Web module</td>
<td>- 11</td>
</tr>
<tr>
<td>DFHWBCDH</td>
<td>CSECT</td>
<td>Web module</td>
<td>- 08</td>
</tr>
<tr>
<td>DFHWBCDL</td>
<td>CSECT</td>
<td>Web module</td>
<td>- 17</td>
</tr>
<tr>
<td>DFHWBCDO</td>
<td>CSECT</td>
<td>Web module</td>
<td>- 07</td>
</tr>
<tr>
<td>DFHWBCNV</td>
<td>Macro</td>
<td>WB CICS Web Interface codepage macro</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHWBCO1</td>
<td>CSECT</td>
<td>Web module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHWBCD</td>
<td>CSECT</td>
<td>Web module</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHWBDL0</td>
<td>CSECT</td>
<td>Autocall SCEEOBJ</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHWBDM</td>
<td>CSECT</td>
<td>Domain initialization</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHWBDUF</td>
<td>CSECT</td>
<td>Web module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHWBENV</td>
<td>CSECT</td>
<td>Web module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHWBEP</td>
<td>CSECT</td>
<td>Web error program</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHWBEPL</td>
<td>CSECT</td>
<td></td>
<td>- 17</td>
</tr>
<tr>
<td>DFHWBG</td>
<td>CSECT</td>
<td>WB Web Interface Garbage Collection</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHWIMG</td>
<td>CSECT</td>
<td>Web module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHWBIP</td>
<td>CSECT</td>
<td>Web module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHWBIPA</td>
<td>CSECT</td>
<td>Web module</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHWBIPM</td>
<td>Macro</td>
<td>DFHWBIP interface macro</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHWBPT</td>
<td>CSECT</td>
<td>Web module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHWBLA</td>
<td>CSECT</td>
<td>Web module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHWOUT</td>
<td>CSECT</td>
<td>Web module</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHWBPA</td>
<td>CSECT</td>
<td>Web module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHWBM</td>
<td>CSECT</td>
<td>Domain initialization</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHWBQMT</td>
<td>CSECT</td>
<td></td>
<td>- 03</td>
</tr>
<tr>
<td>DFHWBRP</td>
<td>CSECT</td>
<td>Web module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHWBSR</td>
<td>CSECT</td>
<td>WB Web Send/Receive</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHWBSRT</td>
<td>CSECT</td>
<td></td>
<td>- 03</td>
</tr>
<tr>
<td>DFHWBST</td>
<td>CSECT</td>
<td>Web module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHWBSTT</td>
<td>CSECT</td>
<td>Web module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHWBTC</td>
<td>CSECT</td>
<td>Web module</td>
<td>- 03</td>
</tr>
</tbody>
</table>
Table 123. CICS modules directory (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHWBTC0</td>
<td>CSECT</td>
<td>Web module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHWBTC1</td>
<td>CSECT</td>
<td>Web module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHWBTC2</td>
<td>CSECT</td>
<td>Web module</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHWBTC3</td>
<td>CSECT</td>
<td>Web module</td>
<td>- 08</td>
</tr>
<tr>
<td>DFHWBTC4</td>
<td>CSECT</td>
<td>Web module</td>
<td>- 17</td>
</tr>
<tr>
<td>DFHWBTC5</td>
<td>CSECT</td>
<td>Web module</td>
<td>- 07</td>
</tr>
<tr>
<td>DFHWBTC6</td>
<td>CSECT</td>
<td>Web module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHWBTC7</td>
<td>CSECT</td>
<td>Web module</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHWBTC8</td>
<td>CSECT</td>
<td>Web module</td>
<td>- 08</td>
</tr>
<tr>
<td>DFHWBTC9</td>
<td>CSECT</td>
<td>Web module</td>
<td>- 17</td>
</tr>
<tr>
<td>DFHWTLC1</td>
<td>CSECT</td>
<td>Web module</td>
<td>C2 -</td>
</tr>
<tr>
<td>DFHWTLC2</td>
<td>CSECT</td>
<td>Web module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHWTLC3</td>
<td>CSECT</td>
<td>Web module</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHWTLC4</td>
<td>CSECT</td>
<td>Web module</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHWTLC5</td>
<td>CSECT</td>
<td>Web module</td>
<td>- 08</td>
</tr>
<tr>
<td>DFHWTLC6</td>
<td>CSECT</td>
<td>Web module</td>
<td>- 17</td>
</tr>
<tr>
<td>DFHWTLC7</td>
<td>CSECT</td>
<td>Web module</td>
<td>- 07</td>
</tr>
<tr>
<td>DFHWTLC8</td>
<td>CSECT</td>
<td>Web Interface Unescaping Program</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHWTLC9</td>
<td>CSECT</td>
<td>Web Interface Unescaping parameter list</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHWBUNH</td>
<td>CSECT</td>
<td>Web Interface Unescaping program</td>
<td>- 08</td>
</tr>
<tr>
<td>DFHWBUNL</td>
<td>CSECT</td>
<td>Web Interface Unescaping program</td>
<td>- 17</td>
</tr>
<tr>
<td>DFHWBUNO</td>
<td>CSECT</td>
<td>Web Interface Unescaping program</td>
<td>- 07</td>
</tr>
<tr>
<td>DFHWBXM</td>
<td>CSECT</td>
<td>Web Interface Attach Client</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHWBXM1</td>
<td>CSECT</td>
<td>Web Interface Attach Client</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHWBXN</td>
<td>CSECT</td>
<td>Web Attach Processing</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHWCCS</td>
<td>CSECT</td>
<td>CAVM common services</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHWGDS</td>
<td>DSECT</td>
<td>CAVM global control block</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHWGNT</td>
<td>CSECT</td>
<td>CAVM entry point table for routines above</td>
<td>OS 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>16MB line</td>
<td></td>
</tr>
<tr>
<td>DFHWGSDS</td>
<td>DSECT</td>
<td>XRF static storage</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHWDSX</td>
<td>CSECT</td>
<td>XRF process dispatcher attach control</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHWDSY</td>
<td>CSECT</td>
<td>XRF process dispatcher initialization</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHWDSZ</td>
<td>CSECT</td>
<td>XRF process dispatcher</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHWDSZ</td>
<td>DSECT</td>
<td>CAVM dispatcher interface parameter block</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHWDSR</td>
<td>CSECT</td>
<td>PC/ABEND handler for XRF dispatcher</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHWDSW</td>
<td>CSECT</td>
<td>XRF process dispatcher wait services</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHWDSX</td>
<td>DSECT</td>
<td>CAVM file control block</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHWKX</td>
<td>CSECT</td>
<td>Warm keypoint program</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHWLF</td>
<td>Macro</td>
<td>XRF LIFO free storage request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHWLFF</td>
<td>CSECT</td>
<td>XRF LIFO free allocation service</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHWLFG</td>
<td>Macro</td>
<td>XRF LIFO get storage request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHWLG</td>
<td>Macro</td>
<td>XRF LIFO get allocation service</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHWMG</td>
<td>Macro</td>
<td>XRF LIFO get allocation service</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHWML</td>
<td>Macro</td>
<td>XRF message manager, signon initialization</td>
<td>OS 03</td>
</tr>
<tr>
<td></td>
<td>CSECT</td>
<td>XRF message manager, I/O services</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHWMM</td>
<td>CSECT</td>
<td>XRF message manager, data copying service</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHWMM1</td>
<td>CSECT</td>
<td>XRF message manager, PUTMSG process</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHWMM2</td>
<td>CSECT</td>
<td>XRF message manager, CICS TCB part of GETMSG processing</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHWMM3</td>
<td>CSECT</td>
<td>XRF message manager, message block services for GETMSG</td>
<td>OS 03</td>
</tr>
</tbody>
</table>

CICS directory
### Table 123. CICS modules directory (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFWWQP</td>
<td>CSECT</td>
<td>XRF message manager, CICS TCB part of PUTMSG processing</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFWWQS</td>
<td>CSECT</td>
<td>XRF message manager, work queue services</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFWWMD</td>
<td>CSECT</td>
<td>XRF message manager, message reader</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFWWMS</td>
<td>CSECT</td>
<td>XRF message manager, request interface</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFWWMS20</td>
<td>CSECT</td>
<td>XRF message manager, request router</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFWWMMR</td>
<td>CSECT</td>
<td>XRF message manager, output routine</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFWWNFDS</td>
<td>DSECT</td>
<td>CAVM NOTIFY exit parameter block</td>
<td>OS -</td>
</tr>
<tr>
<td>DFWWORDER</td>
<td>CSECT</td>
<td>WORDS function (used by DFWOBME)</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFWYOS</td>
<td>CSECT</td>
<td>XRF overseer startup module</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFWYOSA</td>
<td>CSECT</td>
<td>XRF overseer initialization module</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFWYOSB</td>
<td>CSECT</td>
<td>XRF overseer services module</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFWYOSM</td>
<td>Macro</td>
<td>XRF overseer interface definition</td>
<td>11 -</td>
</tr>
<tr>
<td>DFWYSDS</td>
<td>DSECT</td>
<td>CAVM surveillance status control block</td>
<td>OS -</td>
</tr>
<tr>
<td>DFWXCDS</td>
<td>DSECT</td>
<td>CAVM time-of-day difference control area</td>
<td>OS -</td>
</tr>
<tr>
<td>DFWXSMDS</td>
<td>DSECT</td>
<td>CAVM state management record</td>
<td>OS -</td>
</tr>
<tr>
<td>DFWXSDS</td>
<td>DSECT</td>
<td>XRF table of entry points in load module</td>
<td>OS -</td>
</tr>
<tr>
<td>DFWWSDS</td>
<td>DSECT</td>
<td>CAVM surveillance communication area</td>
<td>OS -</td>
</tr>
<tr>
<td>DFWWSSR</td>
<td>CSECT</td>
<td>CAVM state management request router and subtask entry point</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFWWSSD</td>
<td>DSECT</td>
<td>CAVM state management parameter block</td>
<td>OS -</td>
</tr>
<tr>
<td>DFWWSSN1</td>
<td>CSECT</td>
<td>CAVM state management signon initial entry point</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFWWSSN2</td>
<td>CSECT</td>
<td>CAVM state management signon request handler</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFWWSSN3</td>
<td>CSECT</td>
<td>CAVM state management data set initialization routine</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFWWSSOF</td>
<td>CSECT</td>
<td>CAVM state management sign-off request handler</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFWWSSR</td>
<td>CSECT</td>
<td>CAVM surveillance status reader</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFWWSSW</td>
<td>CSECT</td>
<td>CAVM surveillance status writer</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFWXSTDS</td>
<td>DSECT</td>
<td>XRF takeover parameter area</td>
<td>OS -</td>
</tr>
<tr>
<td>DFWXSTI</td>
<td>CSECT</td>
<td>CAVM surveillance tick generator and system status monitor</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFWXSTKV</td>
<td>CSECT</td>
<td>CAVM state management takeover request handler</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFWXSDS</td>
<td>DSECT</td>
<td>NOTIFY exit control block</td>
<td>OS -</td>
</tr>
<tr>
<td>DFWXSPI</td>
<td>CSECT</td>
<td>CAVM state management CAVM process initialization</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFWXSDS2</td>
<td>DSECT</td>
<td>Parameter list for DFWXSN2</td>
<td>OS -</td>
</tr>
<tr>
<td>DFWXSDS3</td>
<td>DSECT</td>
<td>Parameter list for DFWXSN3</td>
<td>OS -</td>
</tr>
<tr>
<td>DFWXTADS</td>
<td>DSECT</td>
<td>XRF takeover initiation argument block</td>
<td>OS -</td>
</tr>
<tr>
<td>DFWXITI</td>
<td>CSECT</td>
<td>XRF takeover initiation program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFWXITA</td>
<td>Source</td>
<td>XRF takeover initiation program - RST specific routines</td>
<td>OS -</td>
</tr>
<tr>
<td>DFWXITIC</td>
<td>Source</td>
<td>XRF takeover initiation program - CLT specific routines</td>
<td>OS -</td>
</tr>
<tr>
<td>DFWXITII</td>
<td>Source</td>
<td>XRF takeover initiation program - inquire job status</td>
<td>OS -</td>
</tr>
<tr>
<td>DFWXITIJ</td>
<td>Source</td>
<td>XRF takeover initiation program - job termination/wait</td>
<td>OS -</td>
</tr>
<tr>
<td>DFWXTO</td>
<td>Macro</td>
<td>Write to console operator</td>
<td>11 -</td>
</tr>
<tr>
<td>DFWXTOP</td>
<td>CSECT</td>
<td>XRF trace routine</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFWUXBMD</td>
<td>Macro</td>
<td>BMS User Exits Parameter List</td>
<td>11 -</td>
</tr>
<tr>
<td>DFWCXCALL</td>
<td>Macro</td>
<td>EXCI EXEC Interface</td>
<td>11 -</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
<td>Library</td>
</tr>
<tr>
<td>---------------</td>
<td>----------</td>
<td>--------------------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>DFHXCMP</td>
<td>CSECT</td>
<td>EXCI dump services</td>
<td>-</td>
</tr>
<tr>
<td>DFHXCX1P</td>
<td>CSECT</td>
<td>EXCI EXEC API handler</td>
<td>-</td>
</tr>
<tr>
<td>DFHXCXUR</td>
<td>CSECT</td>
<td>EXCI Get Unit of Recovery Tokens</td>
<td>-</td>
</tr>
<tr>
<td>DFHXCXO</td>
<td>Macro</td>
<td>EXCI EXEC options</td>
<td>11</td>
</tr>
<tr>
<td>DFHXCXOPT</td>
<td>DSECT</td>
<td>EXCI options table</td>
<td>19</td>
</tr>
<tr>
<td>DFHXCXP</td>
<td>CSECT</td>
<td>Transaction manager (part)</td>
<td>OS</td>
</tr>
<tr>
<td>DFHXCXPLD</td>
<td>Sample</td>
<td>EXCI CALL parameter list (Assembler)</td>
<td>11</td>
</tr>
<tr>
<td>DFHXCXPLH</td>
<td>Sample</td>
<td>EXCI CALL parameter list (C)</td>
<td>-</td>
</tr>
<tr>
<td>DFHXCXPLL</td>
<td>Sample</td>
<td>EXCI CALL parameter list (PL/I)</td>
<td>-</td>
</tr>
<tr>
<td>DFHXCXPLO</td>
<td>Sample</td>
<td>EXCI CALL parameter list (COBOL)</td>
<td>-</td>
</tr>
<tr>
<td>DFHXCXPRH</td>
<td>DSECT</td>
<td>EXCI program request handler</td>
<td>-</td>
</tr>
<tr>
<td>DFHXCRCRO</td>
<td>Sample</td>
<td>EXCI return codes (Assembler)</td>
<td>11</td>
</tr>
<tr>
<td>DFHXCRCR</td>
<td>Sample</td>
<td>EXCI return codes (C)</td>
<td>02</td>
</tr>
<tr>
<td>DFHXCRCO</td>
<td>Sample</td>
<td>EXCI return codes (PL/I)</td>
<td>-</td>
</tr>
<tr>
<td>DFHXCSTB</td>
<td>CSECT</td>
<td>EXCI stub</td>
<td>-</td>
</tr>
<tr>
<td>DFHXCSSVC</td>
<td>CSECT</td>
<td>EXCI SVC services</td>
<td>-</td>
</tr>
<tr>
<td>DFHXCTAB</td>
<td>CSECT</td>
<td>EXCI language table</td>
<td>-</td>
</tr>
<tr>
<td>DFHXCXTRA</td>
<td>CSECT</td>
<td>EXCI global trap program</td>
<td>11</td>
</tr>
<tr>
<td>DFHXCXTRD</td>
<td>DSECT</td>
<td>EXCI global trap program parameter list</td>
<td>-</td>
</tr>
<tr>
<td>DFHXCXTRI</td>
<td>CSECT</td>
<td>EXCI trace initialization termination, and recovery</td>
<td>-</td>
</tr>
<tr>
<td>DFHXCXTRP</td>
<td>CSECT</td>
<td>EXCI trace services</td>
<td>-</td>
</tr>
<tr>
<td>DFHXCXRM</td>
<td>CSECT</td>
<td>EXCI user-replaceable module</td>
<td>19</td>
</tr>
<tr>
<td>DFHXCXDM</td>
<td>Sample</td>
<td>Data Table User Exits Parameter List</td>
<td>11</td>
</tr>
<tr>
<td>DFHXCXDF</td>
<td>CSECT</td>
<td>DU domain - transaction dump formatter for headers and general information</td>
<td>OS</td>
</tr>
<tr>
<td>DFHXCDFL</td>
<td>Macro</td>
<td>DL/I function shipping</td>
<td>OS</td>
</tr>
<tr>
<td>DFHXFFC</td>
<td>Macro</td>
<td>FC function shipping</td>
<td>OS</td>
</tr>
<tr>
<td>DFHXFFED</td>
<td>Macro</td>
<td>Produce transformation program headings</td>
<td>OS</td>
</tr>
<tr>
<td>DFHXFIC</td>
<td>Macro</td>
<td>IC function shipping</td>
<td>OS</td>
</tr>
<tr>
<td>DFHXFIOA</td>
<td>DSECT</td>
<td>Transformer I/O area</td>
<td>OS</td>
</tr>
<tr>
<td>DFHXFJIC</td>
<td>Macro</td>
<td>JC function shipping</td>
<td>OS</td>
</tr>
<tr>
<td>DFHXFMOD</td>
<td>Macro</td>
<td>Produce data transformation programs</td>
<td>OS</td>
</tr>
<tr>
<td>DFHXFP</td>
<td>CSECT</td>
<td>Online data transformation program</td>
<td>OS</td>
</tr>
<tr>
<td>DFHXFPC</td>
<td>Macro</td>
<td>DFHXFMOD inner macro</td>
<td>OS</td>
</tr>
<tr>
<td>DFHXFQ</td>
<td>CSECT</td>
<td>Batch data transformation program</td>
<td>OS</td>
</tr>
<tr>
<td>DFHXFQU</td>
<td>Macro</td>
<td>TD and TS function shipping</td>
<td>OS</td>
</tr>
<tr>
<td>DFHXFRM</td>
<td>Macro</td>
<td>Function shipping recovery module</td>
<td>-</td>
</tr>
<tr>
<td>DFHXFSM</td>
<td>Macro</td>
<td>DFHXFMOD inner macro</td>
<td>OS</td>
</tr>
<tr>
<td>DFHXFSTG</td>
<td>Macro</td>
<td>XF control block and transformer</td>
<td>11</td>
</tr>
<tr>
<td>DFHXFX</td>
<td>CSECT</td>
<td>Optimized data transformation program</td>
<td>05</td>
</tr>
<tr>
<td>DFHXIS</td>
<td>Sample</td>
<td>XISCONA global user exit program</td>
<td>19</td>
</tr>
<tr>
<td>DFHXISDS</td>
<td>Sample</td>
<td>XISCONA data set information</td>
<td>19</td>
</tr>
<tr>
<td>DFHXLTL</td>
<td>Macro</td>
<td>Transaction list table</td>
<td>11</td>
</tr>
<tr>
<td>DFHXLTD</td>
<td>DSECT</td>
<td>Transaction list table</td>
<td>05</td>
</tr>
<tr>
<td>DFHXMAB</td>
<td>CSECT</td>
<td>XM domain - abend handler</td>
<td>-</td>
</tr>
<tr>
<td>DFHXMAB</td>
<td>CSECT</td>
<td>XM domain - abend handler</td>
<td>-</td>
</tr>
<tr>
<td>DFHXMAB</td>
<td>CSECT</td>
<td>XM domain - attach</td>
<td>-</td>
</tr>
<tr>
<td>DFHXMATA</td>
<td>Source</td>
<td>XMAT parameter list</td>
<td>OS</td>
</tr>
<tr>
<td>DFHXMATM</td>
<td>Source</td>
<td>XMAT request</td>
<td>OS</td>
</tr>
<tr>
<td>DFHXMATT</td>
<td>CSECT</td>
<td>XMAT trace interpretation data</td>
<td>-</td>
</tr>
<tr>
<td>DFHXMABD</td>
<td>CSECT</td>
<td>XM domain - browse</td>
<td>-</td>
</tr>
<tr>
<td>DFHXMABD</td>
<td>Source</td>
<td>XMAB parameter list</td>
<td>OS</td>
</tr>
<tr>
<td>DFHXMABD</td>
<td>Source</td>
<td>XMAB request</td>
<td>OS</td>
</tr>
<tr>
<td>DFHXMABDT</td>
<td>CSECT</td>
<td>XMAB trace interpretation data</td>
<td>-</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
<td>Library</td>
</tr>
<tr>
<td>------------</td>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>DFHXMCDS</td>
<td>DSECT</td>
<td>XM domain - TCLASS statistics</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHXMCDS</td>
<td>DSECT</td>
<td>XM domain - TCLASS statistics</td>
<td>C2 07</td>
</tr>
<tr>
<td>DFHXMCCL</td>
<td>CSECT (OCO)</td>
<td>XM domain - transaction class functions</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHXMCMA</td>
<td>Source</td>
<td>XMCL parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHXMCML</td>
<td>Source</td>
<td>XMCL request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHXMCNL</td>
<td>CSECT (OCO)</td>
<td>XMCL trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHXMCNX</td>
<td>Macro</td>
<td>XMCL request</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHXMCNY</td>
<td>DSECT</td>
<td>XMCL parameter list</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHXMDA</td>
<td>CSECT (OCO)</td>
<td>XM domain - delete installed transaction</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHXMDMA</td>
<td>Source</td>
<td>XMDD parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHXMDMM</td>
<td>Source</td>
<td>XMDD request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHXMDST</td>
<td>CSECT (OCO)</td>
<td>XMDD trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHXMDMD</td>
<td>CSECT (OCO)</td>
<td>XM domain - pre-initialize, initialize, and quiesce domain functions</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHXMDNA</td>
<td>Source</td>
<td>XMDN parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHXMDNT</td>
<td>CSECT</td>
<td>XMDN trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHXMDUA</td>
<td>CSECT (OCO)</td>
<td>Transaction manager SDUMP formatter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHXMXER</td>
<td>CSECT (OCO)</td>
<td>XM domain - XMER gate functions</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHXMXERA</td>
<td>Source</td>
<td>XMER parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHXMXERM</td>
<td>Source</td>
<td>XMER request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHXMXERT</td>
<td>CSECT</td>
<td>XMER trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHXMXFD</td>
<td>CSECT (OCO)</td>
<td>XM domain - XMFD gate functions</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHXMXFDA</td>
<td>Source</td>
<td>XMFD parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHXMXFDM</td>
<td>Macro</td>
<td>XMFD requests</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHXMXFDT</td>
<td>CSECT (OCO)</td>
<td>XMFD trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHXMXGDS</td>
<td>DSECT</td>
<td>XM domain - global statistics</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHXMGDS</td>
<td>DSECT</td>
<td>XM domain - global statistics</td>
<td>C2 07</td>
</tr>
<tr>
<td>DFHXMIQ</td>
<td>CSECT (OCO)</td>
<td>XM domain - XMIQ gate functions</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHXMIQA</td>
<td>Source</td>
<td>XMIQ parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHXMIQI</td>
<td>Source</td>
<td>XMIQ request (inline form of DFHXMIQM)</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHXIMIQM</td>
<td>Source</td>
<td>XMIQ requests</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHXIMIQT</td>
<td>CSECT (OCO)</td>
<td>XMIQ trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHXMIQX</td>
<td>Macro</td>
<td>XMIQ requests</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHXMIQY</td>
<td>DSECT</td>
<td>XMIQ parameter list</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHXMXLD</td>
<td>CSECT (OCO)</td>
<td>XM domain - XMLD gate functions</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHXMXLDA</td>
<td>Source</td>
<td>XMLD parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHXMXLDM</td>
<td>Source</td>
<td>XMLD requests</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHXMXLDT</td>
<td>CSECT</td>
<td>XMLD trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHXMNTA</td>
<td>DSECT</td>
<td>XMNT parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHXMNTT</td>
<td>CSECT</td>
<td>XMNT trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHXMPPA</td>
<td>DSECT</td>
<td>XMPP parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHXMPPIT</td>
<td>CSECT</td>
<td>XMPP trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHXMQC</td>
<td>CSECT (OCO)</td>
<td>XM domain - tclass functions subroutine</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHXMQCA</td>
<td>Source</td>
<td>XMQC parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHXMQCM</td>
<td>Source</td>
<td>XMQC request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHXMQCT</td>
<td>CSECT</td>
<td>XMQC trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHXMQD</td>
<td>CSECT (OCO)</td>
<td>XM domain - quiesce and delete transaction definitions functions subroutine</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHXMQDT</td>
<td>CSECT (OCO)</td>
<td>XMOD trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHXMRDS</td>
<td>DSECT</td>
<td>XM domain - transaction statistics</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHXMRDS</td>
<td>DSECT</td>
<td>XM domain - transaction statistics</td>
<td>C2 07</td>
</tr>
<tr>
<td>DFHXMRM</td>
<td>CSECT</td>
<td>XM domain Run Transaction Syncpoint Process.</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHXMRM1</td>
<td>CSECT</td>
<td></td>
<td>- 03</td>
</tr>
<tr>
<td>DFHXMRP</td>
<td>CSECT (OCO)</td>
<td>XM domain - definition recovery subroutine</td>
<td>- 03</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
<td>Library</td>
</tr>
<tr>
<td>------------</td>
<td>-----------</td>
<td>--------------------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>DFHXMRPT</td>
<td>CSECT (OCO)</td>
<td>XMRP trace interpretation data</td>
<td>03</td>
</tr>
<tr>
<td>DFHXMRSD</td>
<td>DSECT (OCO)</td>
<td>XM domain - communications area for transaction restart (Assembler)</td>
<td>11</td>
</tr>
<tr>
<td>DFHXMRSH</td>
<td>DSECT (OCO)</td>
<td>XM domain - communications area for transaction restart (C/370)</td>
<td>08</td>
</tr>
<tr>
<td>DFHXMRSL</td>
<td>DSECT (OCO)</td>
<td>XM domain - communications area for transaction restart (PL/I)</td>
<td>17</td>
</tr>
<tr>
<td>DFHXMRSO</td>
<td>DSECT (OCO)</td>
<td>XM domain - communications area for transaction restart (COBOL)</td>
<td>07</td>
</tr>
<tr>
<td>DFHXMRU</td>
<td>CSECT</td>
<td>XMRU CDURUN and Gate Module</td>
<td>03</td>
</tr>
<tr>
<td>DFHXMRUT</td>
<td>CSECT</td>
<td></td>
<td>03</td>
</tr>
<tr>
<td>DFHXMSG</td>
<td>CSECT</td>
<td>Default XRF recovery message</td>
<td>05 03</td>
</tr>
<tr>
<td>DFHXMSR</td>
<td>CSECT (OCO)</td>
<td>XM domain - XMSR gate functions</td>
<td>03</td>
</tr>
<tr>
<td>DFHXMSRA</td>
<td>Source</td>
<td>XMSR parameter list</td>
<td>05</td>
</tr>
<tr>
<td>DFHXMSRM</td>
<td>Source</td>
<td>XMSR request</td>
<td>05</td>
</tr>
<tr>
<td>DFHXMSRT</td>
<td>CSECT (OCO)</td>
<td>XMSR trace interpretation data</td>
<td>05</td>
</tr>
<tr>
<td>DFHXMSRX</td>
<td>Macro</td>
<td>XMSR request</td>
<td>05</td>
</tr>
<tr>
<td>DFHXMSUA</td>
<td>DSECT</td>
<td>XMSU parameter list</td>
<td>05</td>
</tr>
<tr>
<td>DFHXMSUM</td>
<td>Macro</td>
<td>XMSU request</td>
<td>05</td>
</tr>
<tr>
<td>DFHXMSUT</td>
<td>CSECT</td>
<td>XMSU trace interpretation data</td>
<td>05 03</td>
</tr>
<tr>
<td>DFHXMTA</td>
<td>CSECT (OCO)</td>
<td>XM domain - task reply gate</td>
<td>03</td>
</tr>
<tr>
<td>DFHXMTA</td>
<td>CSECT (OCO)</td>
<td>XM domain - trace initialization, termination, and recovery</td>
<td>03</td>
</tr>
<tr>
<td>DFHXMTRM</td>
<td>Macro</td>
<td>Obtain 3 character task number from TCA of task issuing trace put</td>
<td>05</td>
</tr>
<tr>
<td>DFHXMXD</td>
<td>CSECT (OCO)</td>
<td>XM domain - XMDX gate functions</td>
<td>03</td>
</tr>
<tr>
<td>DFHXMXDA</td>
<td>Source</td>
<td>XMDX parameter list</td>
<td>05</td>
</tr>
<tr>
<td>DFHXMXDD</td>
<td>Source</td>
<td>XMDX transaction definition instance parameter list</td>
<td>05</td>
</tr>
<tr>
<td>DFHXMXDI</td>
<td>Source</td>
<td>XMDX request (inline form of DFHXMXDM)</td>
<td>05</td>
</tr>
<tr>
<td>DFHXMXDT</td>
<td>CSECT (OCO)</td>
<td>XMDX trace interpretation data</td>
<td>05</td>
</tr>
<tr>
<td>DFHXMXDX</td>
<td>Source</td>
<td>XMDX request</td>
<td>11</td>
</tr>
<tr>
<td>DFHXMXDY</td>
<td>DSECT</td>
<td>XMDX parameter list</td>
<td>11</td>
</tr>
<tr>
<td>DFHXMXME</td>
<td>CSECT (OCO)</td>
<td>XM domain - XMKE gate functions</td>
<td>03</td>
</tr>
<tr>
<td>DFHXMXEA</td>
<td>Source</td>
<td>XMKE parameter list</td>
<td>05</td>
</tr>
<tr>
<td>DFHXMXEM</td>
<td>Source</td>
<td>XMKE request</td>
<td>05</td>
</tr>
<tr>
<td>DFHXMXET</td>
<td>CSECT (OCO)</td>
<td>XMKE trace interpretation data</td>
<td>03</td>
</tr>
<tr>
<td>DFHXMXM</td>
<td>CSECT</td>
<td>Run Transaction XM Attach Client</td>
<td>03</td>
</tr>
<tr>
<td>DFHXMXND</td>
<td>CSECT (OCO)</td>
<td>XM domain - transaction storage</td>
<td>05</td>
</tr>
<tr>
<td>DFHXOPU@</td>
<td>CSECT</td>
<td></td>
<td>03</td>
</tr>
<tr>
<td>DFHXOPUS</td>
<td>Sample</td>
<td>Sample IIOP URM (C Version)</td>
<td>19</td>
</tr>
<tr>
<td>DFHXQBF</td>
<td>CSECT</td>
<td>XQ queue server buffer pool routines</td>
<td>03</td>
</tr>
<tr>
<td>DFHXQCF</td>
<td>CSECT</td>
<td>XQ queue server coupling facility I/O</td>
<td>03</td>
</tr>
<tr>
<td>DFHXQCN</td>
<td>CSECT</td>
<td>XQ queue server connect/disconnect</td>
<td>03</td>
</tr>
<tr>
<td>DFHXQDF</td>
<td>CSECT</td>
<td>XQ TS queue pool server definitions</td>
<td>03</td>
</tr>
<tr>
<td>DFHXQEN</td>
<td>CSECT</td>
<td>XQ ENF event interface</td>
<td>03</td>
</tr>
<tr>
<td>DFHXQIF</td>
<td>CSECT</td>
<td>XQ queue server interface module</td>
<td>03</td>
</tr>
<tr>
<td>DFHXQIQ</td>
<td>CSECT</td>
<td>XQ queue server inquire module</td>
<td>03</td>
</tr>
<tr>
<td>DFHXQMN</td>
<td>CSECT</td>
<td>XQ queue server mainline</td>
<td>03</td>
</tr>
<tr>
<td>DFHXQMS</td>
<td>CSECT</td>
<td>XQ queue pool server messages</td>
<td>03</td>
</tr>
<tr>
<td>DFHXQQP</td>
<td>CSECT</td>
<td>XQ queue server command processing</td>
<td>03</td>
</tr>
<tr>
<td>DFHXQPR</td>
<td>CSECT</td>
<td>XQ queue server parameter processing</td>
<td>03</td>
</tr>
<tr>
<td>DFHXQRL</td>
<td>CSECT</td>
<td>XQ queue server reload routine</td>
<td>03</td>
</tr>
</tbody>
</table>
### Table 123. CICS modules directory (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHXQRO</td>
<td>CSECT</td>
<td>XQ queue server request routine</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHXORS</td>
<td>CSECT</td>
<td>XQ ARM Restart Support</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHXQST</td>
<td>CSECT</td>
<td>XQ queue server statistics</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHXQS1D</td>
<td>CSECT</td>
<td>XQ list structure statistics record</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHXQS2D</td>
<td>CSECT</td>
<td>XQ queue buffer statistics record</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHXQS3D</td>
<td>CSECT</td>
<td>XQ main storage statistics record</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHXQUL</td>
<td>CSECT</td>
<td>XQ queue server unload routine</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHXR</td>
<td>Macro</td>
<td>XRF code generation macro</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHXRA</td>
<td>CSECT</td>
<td>XRF request processing program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHXRB</td>
<td>CSECT</td>
<td>XRF NOTIFY exit program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHXRC</td>
<td>CSECT</td>
<td>XRF inquire status exit program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHXRCP</td>
<td>CSECT</td>
<td>XRF console communication program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHXRDUF</td>
<td>CSECT (OCO)</td>
<td>XRF SDUMP formatter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHXRE</td>
<td>CSECT</td>
<td>XRF startup program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHXRF</td>
<td>CSECT</td>
<td>XRF CAVM sign-off interface</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHXRHDS</td>
<td>DSECT</td>
<td>XRF health data definition</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHXRDOCL</td>
<td>Other</td>
<td>Used by DFHCRST cataloged procedure</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHXRSP</td>
<td>CSECT</td>
<td>XRF surveillance program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHXRXDF</td>
<td>CSECT</td>
<td>DU domain - transaction dump formatter for XRF</td>
<td>OS 03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>XRF related areas</td>
<td></td>
</tr>
<tr>
<td>DFHXSAD</td>
<td>CSECT (OCO)</td>
<td>XS domain - XSAD gate functions</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHXSADA</td>
<td>Source</td>
<td>XSAD parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHXSADM</td>
<td>Source</td>
<td>XSAD request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHXSADT</td>
<td>CSECT (OCO)</td>
<td>XSAD trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHXSDM</td>
<td>CSECT (OCO)</td>
<td>XS domain - initialize, quiesce, terminate domain functions</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHXSDUF</td>
<td>CSECT (OCO)</td>
<td>XS domain - SDUMP formatter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHXSEAI</td>
<td>CSECT</td>
<td>Early verification stub program</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHXSEJ</td>
<td>CSECT</td>
<td>Security Interfaces for EJB</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHXSEJT</td>
<td>CSECT</td>
<td>- 03</td>
<td></td>
</tr>
<tr>
<td>DFHXSEV</td>
<td>CSECT (OCO)</td>
<td>XS domain - early verification support</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHXSFMA</td>
<td>Source</td>
<td>XSFL parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHXSFML</td>
<td>Source</td>
<td>XSFL request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHXSFLT</td>
<td>CSECT (OCO)</td>
<td>XSFL trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHXSIDT</td>
<td>CSECT (OCO)</td>
<td>XS domain - trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHXSIS</td>
<td>CSECT (OCO)</td>
<td>XS domain - XSIS gate functions</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHXSISA</td>
<td>Source</td>
<td>XSIS parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHXSISIM</td>
<td>Source</td>
<td>XSIS request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHXSIST</td>
<td>CSECT (OCO)</td>
<td>XSIS trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHXSLU</td>
<td>CSECT (OCO)</td>
<td>XS domain - XSLU gate functions</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHXSLUA</td>
<td>Source</td>
<td>XSLU parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHXSLUM</td>
<td>Source</td>
<td>XSLU request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHXSLUT</td>
<td>CSECT (OCO)</td>
<td>XSLU trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHXSPUB</td>
<td>DSECT (OCO)</td>
<td>XS domain - public storage fields</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHXSPW</td>
<td>CSECT (OCO)</td>
<td>XS domain - XSPW gate functions</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHXSPWA</td>
<td>Source</td>
<td>XSPW parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHXSPWM</td>
<td>Source</td>
<td>XSPW request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHXSPWT</td>
<td>CSECT (OCO)</td>
<td>XSPW trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHXSRC</td>
<td>CSECT (OCO)</td>
<td>XS domain - XSRC gate functions</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHXSRCM</td>
<td>Source</td>
<td>XSRC parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHXSRCI</td>
<td>Source</td>
<td>XSRC request (inline form of DFHXSRCM)</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHXSRCM</td>
<td>Macro</td>
<td>XSRC requests</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHXSRCT</td>
<td>CSECT (OCO)</td>
<td>XSRC trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHXSSA</td>
<td>CSECT (OCO)</td>
<td>XS domain - supervisor request router</td>
<td>- 03</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
<td>Library</td>
</tr>
<tr>
<td>------------</td>
<td>-------</td>
<td>------------------------------------------------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>DFHXSSAT</td>
<td>CSECT</td>
<td>XSSA trace interpretation data</td>
<td>03</td>
</tr>
<tr>
<td>DFHXSSBT</td>
<td>CSECT</td>
<td>XSSB trace interpretation data</td>
<td>03</td>
</tr>
<tr>
<td>DFHXSSC</td>
<td>CSECT</td>
<td>XS domain - resource checking functions</td>
<td>03</td>
</tr>
<tr>
<td>DFHXSSD</td>
<td>CSECT</td>
<td>XS domain - create passticket function</td>
<td>03</td>
</tr>
<tr>
<td>DFHXSSDT</td>
<td>CSECT</td>
<td>XSSD trace interpretation data</td>
<td>03</td>
</tr>
<tr>
<td>DFHXSSSE</td>
<td>CSECT</td>
<td>Security Supervisor Phase E Cert. Management</td>
<td>03</td>
</tr>
<tr>
<td>DFHXSSSET</td>
<td>CSECT</td>
<td></td>
<td>03</td>
</tr>
<tr>
<td>DFHXSSI</td>
<td>CSECT</td>
<td>XS domain - storage initialization</td>
<td>03</td>
</tr>
<tr>
<td>DFHXSSIT</td>
<td>CSECT</td>
<td>XSSI trace interpretation data</td>
<td>03</td>
</tr>
<tr>
<td>DFHXSTRI</td>
<td>CSECT</td>
<td>XS domain - trace initialization, termination, and recovery</td>
<td>03</td>
</tr>
<tr>
<td>DFHXSTRI</td>
<td>Macro</td>
<td>Installation data for ESM exits</td>
<td>11</td>
</tr>
<tr>
<td>DFHXSTM</td>
<td>CSECT</td>
<td>XRF message manager for security manager</td>
<td>03</td>
</tr>
<tr>
<td>DFHXSSMM</td>
<td>CSECT</td>
<td>XSM parameter list</td>
<td>03</td>
</tr>
<tr>
<td>DFHXSSMM</td>
<td>Macro</td>
<td>XSM request</td>
<td>03</td>
</tr>
<tr>
<td>DFHXSSM</td>
<td>CSECT</td>
<td>XS domain - XM domain interface</td>
<td>03</td>
</tr>
<tr>
<td>DFHXSSM</td>
<td>Macro</td>
<td>XSM parameter list</td>
<td>03</td>
</tr>
<tr>
<td>DFHXSSMM</td>
<td>Macro</td>
<td>XSM requests</td>
<td>03</td>
</tr>
<tr>
<td>DFHXSSMT</td>
<td>CSECT</td>
<td>XSM trace interpretation data</td>
<td>03</td>
</tr>
<tr>
<td>DFHXST</td>
<td>Macro</td>
<td>DFHXST internal table generator</td>
<td>03</td>
</tr>
<tr>
<td>DFHXST</td>
<td>Macro</td>
<td>BMS internal macro</td>
<td>11</td>
</tr>
<tr>
<td>DFHXSTCI</td>
<td>CSECT</td>
<td>XRF terminal switching</td>
<td>03</td>
</tr>
<tr>
<td>DFHXSTENF</td>
<td>Sample</td>
<td>XICTENF/XALTENF global user exit program</td>
<td>19</td>
</tr>
<tr>
<td>DFHXSTEPT</td>
<td>CSECT</td>
<td>User-replaceable terminal error tables</td>
<td>19</td>
</tr>
<tr>
<td>DFHXTPD</td>
<td>DSECT</td>
<td>XTP internal control blocks</td>
<td>03</td>
</tr>
<tr>
<td>DFHXSTG</td>
<td>Macro</td>
<td>XTP parameter list</td>
<td>03</td>
</tr>
<tr>
<td>DFHXST</td>
<td>Source</td>
<td>XTP data transformation argument</td>
<td>03</td>
</tr>
<tr>
<td>DFHXST</td>
<td>Macro</td>
<td>DFHXST inner macro</td>
<td>03</td>
</tr>
<tr>
<td>DFHXZIDS</td>
<td>DSECT</td>
<td>XZIQUE exit data set information</td>
<td>11</td>
</tr>
<tr>
<td>DFHYBTPL</td>
<td>Other</td>
<td>Cataloged procedure to translate, compile, and link-edit Language</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>Other</td>
<td>Environment/370 PL/I application programs that use EXEC DLI and will run in a batch or CICS shared database region</td>
<td></td>
</tr>
<tr>
<td>DFHYBTVL</td>
<td>Other</td>
<td>Cataloged procedure to translate, compile, and link-edit Language</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>Other</td>
<td>Environment/370 VS COBOL II application programs that use EXEC DLI and will run in a batch or CICS shared database region</td>
<td></td>
</tr>
<tr>
<td>DFHYITDL</td>
<td>Other</td>
<td>Cataloged procedure to translate, compile, and link-edit Language</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>Other</td>
<td>Environment/370 C application programs</td>
<td></td>
</tr>
<tr>
<td>DFHYITEL</td>
<td>Other</td>
<td>Cataloged procedure to translate, compile, and link-edit C++ application programs using the LE/370 compiler</td>
<td></td>
</tr>
<tr>
<td>DFHYITPL</td>
<td>Other</td>
<td>Cataloged procedure to translate, compile, and link-edit Language</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>Other</td>
<td>Environment/370 PL/I application programs</td>
<td></td>
</tr>
</tbody>
</table>
### CICS directory

Table 123. CICS modules directory (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHYITVL</td>
<td>Other</td>
<td>Cataloged procedure to translate, compile, and link-edit Language Environment/370 VS COBOL application programs</td>
<td>18 -</td>
</tr>
<tr>
<td>DFHYXTDL</td>
<td>Other</td>
<td>Cataloged procedure to translate, compile, and link-edit Language Environment/370 C application programs that are to use the external CICS interface</td>
<td>18 -</td>
</tr>
<tr>
<td>DFHYXTEL</td>
<td>Other</td>
<td>Cataloged procedure to translate (EXCI), compile, and link-edit C++ application programs using the LE/370 compiler</td>
<td>18 -</td>
</tr>
<tr>
<td>DFHYXTPL</td>
<td>Other</td>
<td>Cataloged procedure to translate, compile, and link-edit Language Environment/370 PL/I application programs that are to use the external CICS interface</td>
<td>18 -</td>
</tr>
<tr>
<td>DFHYXTVL</td>
<td>Other</td>
<td>Cataloged procedure to translate, compile, and link-edit Language Environment/370 VS COBOL application programs that are to use the external CICS interface</td>
<td>18 -</td>
</tr>
<tr>
<td>DFHZABD</td>
<td>CSECT</td>
<td>No VTAM support abend handler</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHZACT</td>
<td>CSECT</td>
<td>Activate scan</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHZAIT</td>
<td>CSECT</td>
<td>Attach initialization table</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHZAND</td>
<td>CSECT</td>
<td>Abend control block</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHZAPB</td>
<td>Sample</td>
<td>3770 application program</td>
<td>19 -</td>
</tr>
<tr>
<td>DFHZARER</td>
<td>CSECT</td>
<td>LU6.2 protocol error and exception handler</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHZARL</td>
<td>CSECT</td>
<td>LU6.2 application request logic</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHZARM</td>
<td>CSECT</td>
<td>LU6.2 migration logic</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHZARR</td>
<td>CSECT</td>
<td>Application request handler</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHZARRQ</td>
<td>CSECT</td>
<td>LU6.2 application receive request logic</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHZARRA</td>
<td>CSECT</td>
<td>LU6.2 application receive buffer support</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHZARRRC</td>
<td>CSECT</td>
<td>LU6.2 classify what next to receive</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHZARRRF</td>
<td>CSECT</td>
<td>LU6.2 receive FMH7 and ER1</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHZASX</td>
<td>CSECT</td>
<td>DFASY exit</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHZATA</td>
<td>CSECT</td>
<td>Autoinstall program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHZATA2</td>
<td>CSECT</td>
<td>ZCINST Autoinstall Program - Console</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHZATD</td>
<td>CSECT</td>
<td>Autoinstall delete program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHZATDX</td>
<td>CSECT</td>
<td>User-replaceable autoinstall exit</td>
<td>19 03</td>
</tr>
<tr>
<td>DFHZATDY</td>
<td>CSECT</td>
<td>User-replaceable autoinstall exit with APPC</td>
<td>19 03</td>
</tr>
<tr>
<td>DFHZATI</td>
<td>CSECT</td>
<td>Automatic task initiation</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHZATMD</td>
<td>CSECT</td>
<td>Automatic terminal remote definition program</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHZATMF</td>
<td>CSECT</td>
<td>Mass flag program for time-out delete</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHZATR</td>
<td>CSECT</td>
<td>Autoinstall restart program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHZATS</td>
<td>CSECT</td>
<td>Remote autoinstall/delete program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHZATT</td>
<td>CSECT</td>
<td>Task attach</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHZBAN</td>
<td>CSECT</td>
<td>Terminal control bind analysis</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHZBKT</td>
<td>CSECT</td>
<td>LU6.2 bracket state machine</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHZBLX</td>
<td>CSECT</td>
<td>VTAM SCIP exit LU6.2 bind handling</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHZBSM</td>
<td>Macro</td>
<td>LU6.2 bracket state macro</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHZCA</td>
<td>CSECT</td>
<td>VTAM working set module</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHZCB</td>
<td>CSECT</td>
<td>VTAM working set module</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHZCC</td>
<td>CSECT</td>
<td>VTAM working set module</td>
<td>OS 03</td>
</tr>
</tbody>
</table>

1510 CICS Transaction Server: CICS Diagnosis Reference
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHZCGRP</td>
<td>CSECT</td>
<td>Attach CGRP task (for DFHZGRP)</td>
<td>-</td>
</tr>
<tr>
<td>DFHZCHM</td>
<td>Macro</td>
<td>LU6.2 chain state macro</td>
<td>OS</td>
</tr>
<tr>
<td>DFHZCHS</td>
<td>CSECT</td>
<td>LU6.2 chain state machine</td>
<td>OS</td>
</tr>
<tr>
<td>DFHZCLS</td>
<td>CSECT</td>
<td>CLSDST</td>
<td>OS</td>
</tr>
<tr>
<td>DFHZCLX</td>
<td>CSECT</td>
<td>CLSDST exit</td>
<td>OS</td>
</tr>
<tr>
<td>DFHZCNA</td>
<td>CSECT</td>
<td>System console activity control</td>
<td>OS</td>
</tr>
<tr>
<td>DFHZCNR</td>
<td>CSECT</td>
<td>System console application request</td>
<td>OS</td>
</tr>
<tr>
<td>DFHZCNT</td>
<td>CSECT</td>
<td>LU6.2 contention state machine</td>
<td>OS</td>
</tr>
<tr>
<td>DFHZCNM</td>
<td>Macro</td>
<td>MRO application state setting</td>
<td>OS</td>
</tr>
<tr>
<td>DFHZCN1</td>
<td>CSECT</td>
<td>CICS Client CCIN Transaction</td>
<td>OS</td>
</tr>
<tr>
<td>DFHZCN2</td>
<td>CSECT</td>
<td>CICS Client CCIN ZC domain subroutine</td>
<td>OS</td>
</tr>
<tr>
<td>DFHZCN2T</td>
<td>DSECT</td>
<td>ZCN2 translate tables</td>
<td>OS</td>
</tr>
<tr>
<td>DFHZCTR1</td>
<td>CSECT</td>
<td>ZC CICS Client trace interpretation</td>
<td>OS</td>
</tr>
<tr>
<td>DFHZCVR</td>
<td>CSECT</td>
<td>Terminal control open VTAM retry</td>
<td>OS</td>
</tr>
<tr>
<td>DFHZCVP</td>
<td>CSECT</td>
<td>Terminal management program</td>
<td>OS</td>
</tr>
<tr>
<td>DFHZCPB</td>
<td>Macro</td>
<td>Bracket control</td>
<td>OS</td>
</tr>
<tr>
<td>DFHZDSP</td>
<td>CSECT</td>
<td>Dispatcher</td>
<td>OS</td>
</tr>
<tr>
<td>DFHZDST</td>
<td>CSECT</td>
<td>SNA-ASCII translator</td>
<td>OS</td>
</tr>
<tr>
<td>DFHZDTDX</td>
<td>Sample</td>
<td>Autoinstall user exit - C/370</td>
<td>OS</td>
</tr>
<tr>
<td>DFHZDWM</td>
<td>CSECT</td>
<td>Error message writer</td>
<td>OS</td>
</tr>
<tr>
<td>DFHZEPRD</td>
<td>DSECT</td>
<td>TCP/ZCP module entry address list</td>
<td>OS</td>
</tr>
<tr>
<td>DFHZEMW</td>
<td>CSECT</td>
<td>ZCP equates</td>
<td>OS</td>
</tr>
<tr>
<td>DFHZERH</td>
<td>CSECT</td>
<td>LU6.2 error program</td>
<td>OS</td>
</tr>
<tr>
<td>DFHZERM</td>
<td>Macro</td>
<td>ZCP error-handling macro</td>
<td>OS</td>
</tr>
<tr>
<td>DFHZEVR</td>
<td>Macro</td>
<td>ZC VTAM exit GTF trace macro</td>
<td>OS</td>
</tr>
<tr>
<td>DFHZEVI</td>
<td>CSECT</td>
<td>LU6.2 security encryption program part 1</td>
<td>OS</td>
</tr>
</tbody>
</table>
CICS directory

Table 123. CICS modules directory (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHZGV2</td>
<td>CSECT</td>
<td>LU6.2 security encryption program part 2</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHZGFE</td>
<td>CSECT</td>
<td>FREEMAIN request</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHZGAI</td>
<td>CSECT (OCO)</td>
<td>APPC autoinstall - create APPC clones</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHZGAIM</td>
<td>Source</td>
<td>ZGAI parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHZGAIT</td>
<td>CSECT</td>
<td>ZGAI trace interpretation data</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHZGMB</td>
<td>CSECT (OCO)</td>
<td>APPC manipulate bitmap</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHZGIMA</td>
<td>Source</td>
<td>ZGMB parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHZGIM</td>
<td>Source</td>
<td>ZGMB request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHZGBT</td>
<td>CSECT (OCO)</td>
<td>ZGMB trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHZGCA</td>
<td>CSECT (OCO)</td>
<td>LU6.2 CNOS actioning</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHZGCAA</td>
<td>Source</td>
<td>ZGCA parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHZGCAM</td>
<td>Source</td>
<td>ZGCA request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHZGCAT</td>
<td>CSECT (OCO)</td>
<td>ZGCA trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHZGCC</td>
<td>CSECT (OCO)</td>
<td>Catalog CNOS services</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHZGCCA</td>
<td>Source</td>
<td>ZGCC parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHZGCM</td>
<td>Source</td>
<td>ZGCC request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHZGCT</td>
<td>CSECT (OCO)</td>
<td>ZGCC trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHZGCH</td>
<td>CSECT</td>
<td>ZC VTAM change macro domain subroutine</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHZGCHA</td>
<td>CSECT</td>
<td>ZGCH parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHZGCHM</td>
<td>Macro</td>
<td>ZGCH request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHZGCHT</td>
<td>DSECT</td>
<td>ZGCH translate tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHZGCN</td>
<td>CSECT (OCO)</td>
<td>LU6.2 CNOS negotiation</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHZGCNA</td>
<td>Source</td>
<td>ZGCN parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHZGCNM</td>
<td>Source</td>
<td>ZGCN request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHZGCT</td>
<td>CSECT (OCO)</td>
<td>ZGCN trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHZGDAA</td>
<td>Source</td>
<td>ZGDA parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHZGDAM</td>
<td>Macro</td>
<td>ZGDA requests</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHZGDAT</td>
<td>CSECT (OCO)</td>
<td>ZGDA trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHZGDCC</td>
<td>CSECT</td>
<td>Terminal control subroutine constants</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHZGET</td>
<td>CSECT</td>
<td>GETMAIN request</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHZGIN</td>
<td>CSECT</td>
<td>ZC VTAM INQUIRE domain subroutine</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHZGINA</td>
<td>CSECT</td>
<td>ZGIN parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHZGINM</td>
<td>Macro</td>
<td>ZGIN request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHZGINT</td>
<td>DSECT</td>
<td>ZGIN translate tables</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHZGPC</td>
<td>CSECT (OCO)</td>
<td>LU6.2 recover CNOS values for modegroups</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHZGPCA</td>
<td>Source</td>
<td>ZGPC parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHZGPCM</td>
<td>Source</td>
<td>ZGPC request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHZGPCT</td>
<td>CSECT (OCO)</td>
<td>ZGPC trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHZGPDR</td>
<td>CSECT (OCO)</td>
<td>VTAM persistent sessions resource handler</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHZGPRA</td>
<td>Source</td>
<td>ZGPR parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHZGPR</td>
<td>Source</td>
<td>ZGPR request (inline form of DFHZGPRM)</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHZGPRM</td>
<td>Source</td>
<td>ZGPR request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHZGPRQ</td>
<td>CSECT (OCO)</td>
<td>ZGPR trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHZGPRP</td>
<td>CSECT (OCO)</td>
<td>VTAM persistent sessions initialization</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHZGPRQ</td>
<td>Source</td>
<td>ZGPR parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHZGPRD</td>
<td>Source</td>
<td>ZGPR control blocks</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHZGPRM</td>
<td>Source</td>
<td>ZGPR request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHZGPRQ</td>
<td>CSECT (OCO)</td>
<td>ZGPR trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHZGSL</td>
<td>CSECT (OCO)</td>
<td>VTAM persistent sessions set logon</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHZGSLA</td>
<td>Source</td>
<td>ZGSL parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHZGSLM</td>
<td>Source</td>
<td>ZGSL request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHZGSLT</td>
<td>CSECT (OCO)</td>
<td>ZGSL trace interpretation data</td>
<td>- 03</td>
</tr>
</tbody>
</table>
## Table 123. CICS modules directory (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHZGTA</td>
<td>CSECT</td>
<td>ZC TMP table alter gate</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHZGTAA</td>
<td>CSECT</td>
<td>ZGTA parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHZGTAM</td>
<td>Macro</td>
<td>ZGTA request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHZGTAT</td>
<td>DSECT</td>
<td>ZGTA translate tables</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHZGTAI</td>
<td>CSECT</td>
<td>ZC TMP table inquire gate</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHZGTIC</td>
<td>CSECT</td>
<td>ZGTA create copybook</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHZGTIM</td>
<td>Macro</td>
<td>ZGTA request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHZGTTT</td>
<td>DSECT</td>
<td>ZGTA translate tables</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHZGTRA</td>
<td>DSECT</td>
<td>ZGTR interface parameter area</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHZGTRM</td>
<td>Macro</td>
<td>DFHZGTR interface macro</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHZGTRT</td>
<td>DSECT</td>
<td>ZGTR translate tables</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHZGUB</td>
<td>CSECT (OCO)</td>
<td>VTAM persistent sessions terminate</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHZGUBA</td>
<td>Source</td>
<td>ZGUB parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHZGUBM</td>
<td>Source</td>
<td>ZGUB request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHZGUBT</td>
<td>CSECT (OCO)</td>
<td>ZGUB trace interpretation data</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHZGURD</td>
<td>CSECT (OCO)</td>
<td>VTAM persistent sessions URD table</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHZGXAA</td>
<td>DSECT</td>
<td>ZEGA parameter list</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHZGXAM</td>
<td>Macro</td>
<td>ZEGA requests</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHZGXAT</td>
<td>CSECT</td>
<td>ZEGA trace interpretation data</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHZHPCH</td>
<td>Macro</td>
<td>Generate authorized path CHECK or CHECK macro</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHZHPDS</td>
<td>DSECT</td>
<td>ZCP call plist for initialization of SRB facility (HPO)</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHZHPRV</td>
<td>Macro</td>
<td>Generate authorized path RECEIVE or RECEIVE macro</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHZHPRX</td>
<td>CSECT</td>
<td>Authorized path SRB mode VTAM EXE</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHZHPSD</td>
<td>Macro</td>
<td>Generate authorized path SEND or SEND macro</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHZPSR</td>
<td>CSECT</td>
<td>Authorized path SRB requests</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHZISP</td>
<td>Source</td>
<td>Terminal control initialization</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHZIS1</td>
<td>CSECT</td>
<td>Allocate/free/point</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHZIS2</td>
<td>CSECT</td>
<td>Prepare/SPR/commit/abend</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHZLEX</td>
<td>CSECT</td>
<td>LERAD exit</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHZLIX</td>
<td>CSECT</td>
<td>Logon exit</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHZLOC</td>
<td>CSECT</td>
<td>Locate</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHZLRP</td>
<td>CSECT</td>
<td>Logical record presentation</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHZLS1</td>
<td>CSECT</td>
<td>LUG.2 CNOS request transaction program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHZLSIM</td>
<td>Macro</td>
<td>LUG.2 CNOS request</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHZLTX</td>
<td>CSECT</td>
<td>LOSTERM exit</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHZMIM</td>
<td>Macro</td>
<td>NACP sense code table generation macro</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHZNAC</td>
<td>CSECT</td>
<td>Node abnormal condition program (NACP)</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHZNCA</td>
<td>CSECT</td>
<td>NACP message table generator</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHZNCE</td>
<td>CSECT</td>
<td>NACP interface to NEP</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHZNCM</td>
<td>Macro</td>
<td>NACP message table generation macro</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHZNCC</td>
<td>CSECT</td>
<td>Sense code analysis</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHZNCSV</td>
<td>CSECT</td>
<td>VTAM return code analysis</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHZNPEI</td>
<td>Macro</td>
<td>NEP interface generator</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHZNPEX</td>
<td>Source</td>
<td>Translated command-level default NEP</td>
<td>19 -</td>
</tr>
<tr>
<td>DFHZNPE0</td>
<td>CSECT</td>
<td>User-replaceable node error program</td>
<td>19 03</td>
</tr>
<tr>
<td>DFHZNSET</td>
<td>Other</td>
<td>SMP/E zone setter (used by cataloged procedures)</td>
<td>11 -</td>
</tr>
<tr>
<td>DFHZNSEP</td>
<td>CSECT</td>
<td>VTAM services procedure error exit</td>
<td>OS 03</td>
</tr>
</tbody>
</table>
### CICS directory

#### Table 123. CICS modules directory (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFWZOPA</td>
<td>CSECT</td>
<td>Dynamic VTAM open</td>
<td>03</td>
</tr>
<tr>
<td>DFWZOPN</td>
<td>CSECT</td>
<td>OPNDST</td>
<td>03</td>
</tr>
<tr>
<td>DFWZOPX</td>
<td>CSECT</td>
<td>OPNDST exit</td>
<td>03</td>
</tr>
<tr>
<td>DFWZPTDX</td>
<td>Sample</td>
<td>Autoinstall user exit - PL/I</td>
<td>- 19</td>
</tr>
<tr>
<td>DFWZQUE</td>
<td>CSECT</td>
<td>Attach chain and queue subroutine</td>
<td>03</td>
</tr>
<tr>
<td>DFWZRAC</td>
<td>CSECT</td>
<td>Receive-any completion</td>
<td>03</td>
</tr>
<tr>
<td>DFWZRAQ</td>
<td>CSECT</td>
<td>Read ahead queuing</td>
<td>03</td>
</tr>
<tr>
<td>DFWZRAR</td>
<td>CSECT</td>
<td>Read ahead retrieval</td>
<td>03</td>
</tr>
<tr>
<td>DFWZRAS</td>
<td>CSECT</td>
<td>Receive-any slowdown processing</td>
<td>03</td>
</tr>
<tr>
<td>DFWZRBD</td>
<td>DSECT</td>
<td>LU6.2 application receive set buffer hdr</td>
<td>03</td>
</tr>
<tr>
<td>DFWZRLP</td>
<td>CSECT</td>
<td>LU6.2 post-VTAM receive logic</td>
<td>03</td>
</tr>
<tr>
<td>DFWZRLX</td>
<td>CSECT</td>
<td>LU6.2 receive exit program</td>
<td>03</td>
</tr>
<tr>
<td>DFWZRPL</td>
<td>Source</td>
<td>TC build receive-any RPLs</td>
<td>03</td>
</tr>
<tr>
<td>DFWZROM</td>
<td>Macro</td>
<td>Add element to RPL completion queue</td>
<td>03</td>
</tr>
<tr>
<td>DFWZRRX</td>
<td>CSECT</td>
<td>Release request exit</td>
<td>03</td>
</tr>
<tr>
<td>DFWZRSVP</td>
<td>CSECT</td>
<td>Resync send program</td>
<td>03</td>
</tr>
<tr>
<td>DFWZRSST</td>
<td>CSECT</td>
<td>RESETSR</td>
<td>03</td>
</tr>
<tr>
<td>DFWZRSV1</td>
<td>CSECT</td>
<td>VTAM LU6.1 resynchronization</td>
<td>03</td>
</tr>
<tr>
<td>DFWZRSV2</td>
<td>CSECT</td>
<td>VTAM LU6.1 resynchronization</td>
<td>03</td>
</tr>
<tr>
<td>DFWZRSV3</td>
<td>CSECT</td>
<td>VTAM LU6.1 resynchronization</td>
<td>03</td>
</tr>
<tr>
<td>DFWZRSVY</td>
<td>CSECT</td>
<td>VTAM LU6.1 resynchronization</td>
<td>03</td>
</tr>
<tr>
<td>DFWZRSV5</td>
<td>CSECT</td>
<td>VTAM LU6.1 resynchronization</td>
<td>03</td>
</tr>
<tr>
<td>DFWZRSY6</td>
<td>CSECT</td>
<td>VTAM LU6.1 resynchronization</td>
<td>03</td>
</tr>
<tr>
<td>DFWZRTRI</td>
<td>CSECT</td>
<td>VTAM LU6.1 resynchronization trace</td>
<td>03</td>
</tr>
<tr>
<td>DFWZRSV1</td>
<td>CSECT</td>
<td>VTAM LU6.1 resynchronization</td>
<td>03</td>
</tr>
<tr>
<td>DFWZRSV2</td>
<td>CSECT</td>
<td>VTAM LU6.1 resynchronization</td>
<td>03</td>
</tr>
<tr>
<td>DFWZRSV3</td>
<td>CSECT</td>
<td>VTAM LU6.1 resynchronization</td>
<td>03</td>
</tr>
<tr>
<td>DFWZRSVY</td>
<td>CSECT</td>
<td>VTAM LU6.1 resynchronization</td>
<td>03</td>
</tr>
<tr>
<td>DFWZRSV5</td>
<td>CSECT</td>
<td>VTAM LU6.1 resynchronization</td>
<td>03</td>
</tr>
<tr>
<td>DFWZRSY6</td>
<td>CSECT</td>
<td>VTAM LU6.1 resynchronization</td>
<td>03</td>
</tr>
<tr>
<td>DFWZRTRI</td>
<td>CSECT</td>
<td>VTAM LU6.1 resynchronization trace</td>
<td>03</td>
</tr>
<tr>
<td>DFWZRSV1</td>
<td>CSECT</td>
<td>VTAM LU6.1 resynchronization</td>
<td>03</td>
</tr>
<tr>
<td>DFWZRSV2</td>
<td>CSECT</td>
<td>VTAM LU6.1 resynchronization</td>
<td>03</td>
</tr>
<tr>
<td>DFWZRSV3</td>
<td>CSECT</td>
<td>VTAM LU6.1 resynchronization</td>
<td>03</td>
</tr>
<tr>
<td>DFWZRSVY</td>
<td>CSECT</td>
<td>VTAM LU6.1 resynchronization</td>
<td>03</td>
</tr>
<tr>
<td>DFWZRSV5</td>
<td>CSECT</td>
<td>VTAM LU6.1 resynchronization</td>
<td>03</td>
</tr>
<tr>
<td>DFWZRSY6</td>
<td>CSECT</td>
<td>VTAM LU6.1 resynchronization</td>
<td>03</td>
</tr>
<tr>
<td>DFWZRTRI</td>
<td>CSECT</td>
<td>VTAM LU6.1 resynchronization trace</td>
<td>03</td>
</tr>
<tr>
<td>DFWZRLV</td>
<td>CSECT</td>
<td>LU6.2 pre-VTAM receive logic</td>
<td>03</td>
</tr>
<tr>
<td>DFWZRSV</td>
<td>CSECT</td>
<td>Receive specific</td>
<td>03</td>
</tr>
<tr>
<td>DFWZRXS</td>
<td>CSECT</td>
<td>Receive specific exit</td>
<td>03</td>
</tr>
<tr>
<td>DFWZRSAX</td>
<td>CSECT</td>
<td>Send DFASY exit</td>
<td>03</td>
</tr>
<tr>
<td>DFWZSCX</td>
<td>CSECT</td>
<td>Session control input exit</td>
<td>03</td>
</tr>
<tr>
<td>DFWZSDA</td>
<td>CSECT</td>
<td>Send asynchronous command</td>
<td>03</td>
</tr>
<tr>
<td>DFWZSLD</td>
<td>CSECT</td>
<td>LU6.2 send logic</td>
<td>03</td>
</tr>
<tr>
<td>DFWZSDR</td>
<td>CSECT</td>
<td>Send response</td>
<td>03</td>
</tr>
<tr>
<td>DFWZSDS</td>
<td>CSECT</td>
<td>Send DFSYN</td>
<td>03</td>
</tr>
<tr>
<td>DFWZSDX</td>
<td>CSECT</td>
<td>Send DFSYN data exit</td>
<td>03</td>
</tr>
<tr>
<td>DFWZSES</td>
<td>CSECT</td>
<td>SESSIONC</td>
<td>03</td>
</tr>
<tr>
<td>DFWZSEX</td>
<td>CSECT</td>
<td>SESSIONC exit</td>
<td>03</td>
</tr>
<tr>
<td>DFWZSHU</td>
<td>CSECT</td>
<td>Checks shutdown status for VTAM terminals</td>
<td>03</td>
</tr>
<tr>
<td>DFWZSIM</td>
<td>CSECT</td>
<td>SIMLOGON</td>
<td>03</td>
</tr>
<tr>
<td>DFWZSXI</td>
<td>CSECT</td>
<td>SIMLOGON exit</td>
<td>03</td>
</tr>
<tr>
<td>DFWZSKR</td>
<td>CSECT</td>
<td>Command response</td>
<td>03</td>
</tr>
<tr>
<td>DFWZSLDS</td>
<td>Symbolic</td>
<td>Send logon status data structure</td>
<td>11</td>
</tr>
<tr>
<td>DFWZSLZ</td>
<td>CSECT</td>
<td>Set logon start</td>
<td>03</td>
</tr>
<tr>
<td>DFWZSLX</td>
<td>CSECT</td>
<td>LU6.2 send exit program</td>
<td>03</td>
</tr>
<tr>
<td>DFWZSSX</td>
<td>CSECT</td>
<td>Send DFSYN exit</td>
<td>03</td>
</tr>
<tr>
<td>DFWZSTAM</td>
<td>Macro</td>
<td>DFHZSTAP interface</td>
<td>03</td>
</tr>
<tr>
<td>DFWZSTAP</td>
<td>CSECT</td>
<td>Conversation state determination</td>
<td>03</td>
</tr>
<tr>
<td>DFWZSTU</td>
<td>CSECT</td>
<td>Terminal control status change</td>
<td>03</td>
</tr>
<tr>
<td>DFWZSUP</td>
<td>CSECT</td>
<td>Startup task</td>
<td>03</td>
</tr>
<tr>
<td>DFWZSYN</td>
<td>CSECT</td>
<td>VTAM recovery module</td>
<td>03</td>
</tr>
<tr>
<td>DFWZSYX</td>
<td>CSECT</td>
<td>SYNAD exit</td>
<td>03</td>
</tr>
<tr>
<td>DFWZS102</td>
<td>DSECT</td>
<td>ZC SUBPOOL_TOKENs table</td>
<td>03</td>
</tr>
<tr>
<td>DFWZTAX</td>
<td>CSECT</td>
<td>Turnaround exit</td>
<td>03</td>
</tr>
<tr>
<td>DFWZTPX</td>
<td>CSECT</td>
<td>TPEND exit</td>
<td>03</td>
</tr>
<tr>
<td>DFWZTR</td>
<td>Macro</td>
<td>ZCP trace macro</td>
<td>03</td>
</tr>
</tbody>
</table>
### Table 123. CICS modules directory (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHZTRA</td>
<td>CSECT</td>
<td>VTAM trace module</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHZTSP</td>
<td>CSECT</td>
<td>Terminal sharing program</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHZUCT</td>
<td>CSECT</td>
<td>Uppercase translate</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHZUUX</td>
<td>CSECT</td>
<td>User input exit</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHZUSR</td>
<td>CSECT</td>
<td>LU6.2 conversation state machine</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHZUSRM</td>
<td>Macro</td>
<td>LU6.2 conversation state macro</td>
<td>-</td>
</tr>
<tr>
<td>DFHZXCU</td>
<td>CSECT</td>
<td>VTAM XRF catch-up transaction</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHZXDFS</td>
<td>CSECT</td>
<td>XRF ZCP queue SDUMP formatter</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHZXPS</td>
<td>CSECT</td>
<td>VTAM persistent sessions APPC recovery</td>
<td>- 03</td>
</tr>
<tr>
<td>DFHZXQO</td>
<td>CSECT</td>
<td>XRF ZCP tracking queue organizer</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHZXQOS</td>
<td>Symbolic</td>
<td>DFHZXQO internal control blocks</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHZXRC</td>
<td>CSECT</td>
<td>XRF and Persistent sessions state data analysis</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHZXREO</td>
<td>CSECT</td>
<td>VTAM reconnect transaction</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHZXRPL</td>
<td>Macro</td>
<td>Clear RPL</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHZXRT</td>
<td>CSECT</td>
<td>Transaction routing - LU6.2 command processor, AOR</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHZXS</td>
<td>Macro</td>
<td>Interface to DFHZXST</td>
<td>OS -</td>
</tr>
<tr>
<td>DFHZXST</td>
<td>CSECT</td>
<td>XRF ZCP session-state tracking</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFHZXSTS</td>
<td>CSECT</td>
<td>SETLOGON routine</td>
<td>OS 03</td>
</tr>
<tr>
<td>DFH0AZBC</td>
<td>Sample</td>
<td>FEPI sample: CICS back-end application</td>
<td>19 -</td>
</tr>
<tr>
<td>DFH0AZBI</td>
<td>Sample</td>
<td>FEPI sample: IMS back-end application</td>
<td>19 -</td>
</tr>
<tr>
<td>DFH0AZPA</td>
<td>Sample</td>
<td>FEPI sample: SLU P pseudo-conversational program (Assembler)</td>
<td>19 -</td>
</tr>
<tr>
<td>DFH0AZPS</td>
<td>Sample</td>
<td>FEPI sample: SLU P one-out one-in program (Assembler)</td>
<td>19 -</td>
</tr>
<tr>
<td>DFH0AZQS</td>
<td>Sample</td>
<td>FEPI sample: STSN processing</td>
<td>19 -</td>
</tr>
<tr>
<td>DFH0AZTD</td>
<td>Sample</td>
<td>FEPI sample: 3270 data stream pass through</td>
<td>19 -</td>
</tr>
<tr>
<td>DFH0AZXS</td>
<td>Sample</td>
<td>FEPI sample: setup program (Assembler)</td>
<td>19 -</td>
</tr>
<tr>
<td>DFH0BAT1</td>
<td>Sample</td>
<td>Batch enabling sample BAT1 -disable transactions coordinator</td>
<td>C3 -</td>
</tr>
<tr>
<td>DFH0BAT2</td>
<td>Sample</td>
<td>Batch enabling sample BAT2 -inquire retained locks coordinator</td>
<td>C3 -</td>
</tr>
<tr>
<td>DFH0BAT3</td>
<td>Sample</td>
<td>Batch enabling sample BAT3 -force retained locks coordinator</td>
<td>C3 -</td>
</tr>
<tr>
<td>DFH0BAT4</td>
<td>Sample</td>
<td>Batch enabling sample BAT1 -disable transactions program</td>
<td>C3 -</td>
</tr>
<tr>
<td>DFH0BAT5</td>
<td>Sample</td>
<td>Batch enabling sample BAT2 -inquire retained locks program</td>
<td>C3 -</td>
</tr>
<tr>
<td>DFH0BAT6</td>
<td>Sample</td>
<td>Batch enabling sample BAT3 -force indoubt UOWs program</td>
<td>C3 -</td>
</tr>
<tr>
<td>DFH0BAT7</td>
<td>Sample</td>
<td>Batch enabling sample BAT2 -retry backout failures program</td>
<td>C3 -</td>
</tr>
<tr>
<td>DFH0BAT8</td>
<td>Sample</td>
<td>Batch enabling sample BAT3 -forcibly release locks program</td>
<td>C3 -</td>
</tr>
<tr>
<td>DFH0BCA</td>
<td>Sample</td>
<td>CUA communication area layout - COBOL</td>
<td>C3 -</td>
</tr>
<tr>
<td>DFH0BCCR</td>
<td>Sample</td>
<td>CUA customer record layout - COBOL</td>
<td>C3 -</td>
</tr>
<tr>
<td>DFH0BC11</td>
<td>Sample</td>
<td>Batch enabling sample BAT1 -disable transactions TS queue</td>
<td>C3 -</td>
</tr>
<tr>
<td>DFH0BC12</td>
<td>Sample</td>
<td>Batch enabling sample BAT1 -disable transactions commarea</td>
<td>C3 -</td>
</tr>
<tr>
<td>DFH0BC21</td>
<td>Sample</td>
<td>Batch enabling sample BAT2 -inquire retained locks TS queue</td>
<td>C3 -</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
<td>Library</td>
</tr>
<tr>
<td>---------------</td>
<td>------------</td>
<td>-------------------------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>DFH0BC22</td>
<td>Sample</td>
<td>Batch enabling sample BAT2 - inquire retained locks commarea</td>
<td>C3</td>
</tr>
<tr>
<td>DFH0BC23</td>
<td>Sample</td>
<td>Batch enabling sample BAT2 - inquire retained locks map texts</td>
<td>C3</td>
</tr>
<tr>
<td>DFH0BC31</td>
<td>Sample</td>
<td>Batch enabling sample BAT3 - force retained locks TS queue</td>
<td>C3</td>
</tr>
<tr>
<td>DFH0BC32</td>
<td>Sample</td>
<td>Batch enabling sample BAT3 - force retained locks commarea</td>
<td>C3</td>
</tr>
<tr>
<td>DFH0BFKT</td>
<td>Sample</td>
<td>CUA variable function key layout - COBOL</td>
<td>C3</td>
</tr>
<tr>
<td>DFH0BFPD</td>
<td>Sample</td>
<td>CUA redefinition of file pull-down - COBOL</td>
<td>C3</td>
</tr>
<tr>
<td>DFH0BHP</td>
<td>Sample</td>
<td>CUA redefinition of help pop-up - COBOL</td>
<td>C3</td>
</tr>
<tr>
<td>DFH0BHPD</td>
<td>Sample</td>
<td>CUA redefinition of help pull-down - COBOL</td>
<td>C3</td>
</tr>
<tr>
<td>DFH0BHR</td>
<td>Sample</td>
<td>CUA help text TS queue layout - COBOL</td>
<td>C3</td>
</tr>
<tr>
<td>DFH0BHT</td>
<td>Sample</td>
<td>CUA help file key table - COBOL</td>
<td>C3</td>
</tr>
<tr>
<td>DFH0BLST</td>
<td>Sample</td>
<td>CUA redefinition of list base panel - COBOL</td>
<td>C3</td>
</tr>
<tr>
<td>DFH0BMSG</td>
<td>Sample</td>
<td>CUA application message table - COBOL</td>
<td>C3</td>
</tr>
<tr>
<td>DFH0BM1</td>
<td>Sample</td>
<td>Batch enabling sample BAT1 - disable transactions BMS mapset</td>
<td>19</td>
</tr>
<tr>
<td>DFH0BM10</td>
<td>Sample</td>
<td>Batch enabling sample BAT1 - disable transactions BMS mapset</td>
<td>C3</td>
</tr>
<tr>
<td>DFH0BM2</td>
<td>Sample</td>
<td>Batch enabling sample BAT2 - inquire retained locks BMS mapset</td>
<td>19</td>
</tr>
<tr>
<td>DFH0BM20</td>
<td>Sample</td>
<td>Batch enabling sample BAT2 - inquire retained locks BMS mapset</td>
<td>C3</td>
</tr>
<tr>
<td>DFH0BM3</td>
<td>Sample</td>
<td>Batch enabling sample BAT3 - force retained locks BMS mapset</td>
<td>19</td>
</tr>
<tr>
<td>DFH0BM30</td>
<td>Sample</td>
<td>Batch enabling sample BAT3 - force retained locks BMS mapset</td>
<td>C3</td>
</tr>
<tr>
<td>DFH0BRT</td>
<td>Sample</td>
<td>CUA program routing control table - COBOL</td>
<td>C3</td>
</tr>
<tr>
<td>DFH0BTSQ</td>
<td>Sample</td>
<td>CUA TS queue details layout - COBOL</td>
<td>C3</td>
</tr>
<tr>
<td>DFH0BZCA</td>
<td>Sample</td>
<td>FEPI sample: system definition and customization (Assembler)</td>
<td>19</td>
</tr>
<tr>
<td>DFH0BZCC</td>
<td>Sample</td>
<td>FEPI sample: system definition and customization (C/370)</td>
<td>D3</td>
</tr>
<tr>
<td>DFH0BZCO</td>
<td>Sample</td>
<td>FEPI sample: system definition and customization (COBOL)</td>
<td>C3</td>
</tr>
<tr>
<td>DFH0BZCP</td>
<td>Sample</td>
<td>FEPI sample: system definition and customization (PL/I)</td>
<td>P3</td>
</tr>
<tr>
<td>DFH0BZMA</td>
<td>Sample</td>
<td>FEPI sample: messages &amp; text (Assembler)</td>
<td>19</td>
</tr>
<tr>
<td>DFH0BZMC</td>
<td>Sample</td>
<td>FEPI sample: messages &amp; text (C/370)</td>
<td>D3</td>
</tr>
<tr>
<td>DFH0BZMD</td>
<td>Sample</td>
<td>FEPI sample: messages &amp; text (COBOL)</td>
<td>C3</td>
</tr>
<tr>
<td>DFH0BZMP</td>
<td>Sample</td>
<td>FEPI sample: messages &amp; text (PL/I)</td>
<td>P3</td>
</tr>
<tr>
<td>DFH0BZ10</td>
<td>Sample</td>
<td>FEPI sample: front-end terminal map (COBOL)</td>
<td>C3</td>
</tr>
<tr>
<td>DFH0BZ20</td>
<td>Sample</td>
<td>FEPI sample: front-end terminal map (COBOL)</td>
<td>C3</td>
</tr>
<tr>
<td>DFH0BZ3A</td>
<td>Sample</td>
<td>FEPI sample: front-end terminal map (COBOL)</td>
<td>19</td>
</tr>
<tr>
<td>DFH0BZ40</td>
<td>Sample</td>
<td>FEPI sample: front-end terminal map (COBOL)</td>
<td>C3</td>
</tr>
<tr>
<td>DFH0BZ50</td>
<td>Sample</td>
<td>FEPI sample: front-end terminal map (COBOL)</td>
<td>C3</td>
</tr>
<tr>
<td>DFH0BZ6C</td>
<td>Sample</td>
<td>FEPI sample: front-end terminal map (C/370)</td>
<td>D3</td>
</tr>
<tr>
<td>DFH0BZ7P</td>
<td>Sample</td>
<td>FEPI sample: front-end terminal map (PL/I)</td>
<td>P3</td>
</tr>
<tr>
<td>DFH0BZ8A</td>
<td>Sample</td>
<td>FEPI sample: front-end terminal map</td>
<td>19</td>
</tr>
<tr>
<td>DFH0BZ9A</td>
<td>Sample</td>
<td>FEPI sample: front-end terminal map</td>
<td>19</td>
</tr>
<tr>
<td>DFH0CALL</td>
<td>Sample</td>
<td>Inquiry/update - COBOL</td>
<td>C3</td>
</tr>
<tr>
<td>DFH0CBAC</td>
<td>Sample</td>
<td>Sample CICS BTS 3270 Transaction Client</td>
<td>C3</td>
</tr>
</tbody>
</table>
### Table 123. CICS modules directory (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFH0CBAE</td>
<td>Sample</td>
<td>Sample Bridge Exit Routine</td>
<td>C3</td>
</tr>
<tr>
<td>DFH0CBAI</td>
<td>Sample</td>
<td>Sample input routine for BTS 3270 txn</td>
<td>C3</td>
</tr>
<tr>
<td>DFH0CBAO</td>
<td>Sample</td>
<td>Sample output routine for BTS 3270 txn</td>
<td>C3</td>
</tr>
<tr>
<td>DFH0CBDC</td>
<td>Sample</td>
<td>CSD backup program - COBOL</td>
<td>C3</td>
</tr>
<tr>
<td>DFH0CBRD</td>
<td>Sample</td>
<td>Sample bridge exit common area</td>
<td>C3</td>
</tr>
<tr>
<td>DFH0CBRE</td>
<td>Sample</td>
<td>Sample bridge exit</td>
<td>C3</td>
</tr>
<tr>
<td>DFH0CBRF</td>
<td>Sample</td>
<td>Sample bridge formatter</td>
<td>C3</td>
</tr>
<tr>
<td>DFH0CBRU</td>
<td>Sample</td>
<td>Sample bridge exit user area</td>
<td>C3</td>
</tr>
<tr>
<td>DFH0CBRW</td>
<td>Sample</td>
<td>Browse - COBOL</td>
<td>C3</td>
</tr>
<tr>
<td>DFH0CCOM</td>
<td>Sample</td>
<td>Order entry queue print - COBOL</td>
<td>C3</td>
</tr>
<tr>
<td>DFH0CESD</td>
<td>Sample</td>
<td>Shutdown assist program - COBOL</td>
<td>C3</td>
</tr>
<tr>
<td>DFH0CFLIL</td>
<td>Sample</td>
<td>Customer file (FILEA) record layout - COBOL</td>
<td>C3</td>
</tr>
<tr>
<td>DFH0CLOG</td>
<td>Sample</td>
<td>Audit trail (log) record layout - COBOL</td>
<td>C3</td>
</tr>
<tr>
<td>DFH0CLB6</td>
<td>Sample</td>
<td>Order entry queue record layout - COBOL</td>
<td>C3</td>
</tr>
<tr>
<td>DFH0CMA</td>
<td>Sample</td>
<td>Operator instructions map set - COBOL</td>
<td>19</td>
</tr>
<tr>
<td>DFH0CMB</td>
<td>Sample</td>
<td>Customer details map set - COBOL</td>
<td>19</td>
</tr>
<tr>
<td>DFH0CMC</td>
<td>Sample</td>
<td>File browse map set - COBOL</td>
<td>19</td>
</tr>
<tr>
<td>DFH0CMD</td>
<td>Sample</td>
<td>Low balance inquiry map set - COBOL</td>
<td>19</td>
</tr>
<tr>
<td>DFH0CMK</td>
<td>Sample</td>
<td>Order entry map set - COBOL</td>
<td>19</td>
</tr>
<tr>
<td>DFH0CML</td>
<td>Sample</td>
<td>Order report map set - COBOL</td>
<td>19</td>
</tr>
<tr>
<td>DFH0CMNU</td>
<td>Sample</td>
<td>Operator instructions - COBOL</td>
<td>C3</td>
</tr>
<tr>
<td>DFH0CONT</td>
<td>Sample</td>
<td>sample program for CICS BTS</td>
<td>C3</td>
</tr>
<tr>
<td>DFH0CMP</td>
<td>Sample</td>
<td>Keystroke overlap/look-aside query - map set - COBOL</td>
<td>19</td>
</tr>
<tr>
<td>DFH0CPKO</td>
<td>Sample</td>
<td>Keystroke overlap - COBOL</td>
<td>19</td>
</tr>
<tr>
<td>DFH0CPLA</td>
<td>Sample</td>
<td>Look-aside query - COBOL</td>
<td>19</td>
</tr>
<tr>
<td>DFH0CREN</td>
<td>Sample</td>
<td>Order entry - COBOL</td>
<td>19</td>
</tr>
<tr>
<td>DFH0CREP</td>
<td>Sample</td>
<td>Low balance inquiry - COBOL</td>
<td>19</td>
</tr>
<tr>
<td>DFH0CFLM</td>
<td>Sample</td>
<td>CSD cross-reference program - COBOL</td>
<td>19</td>
</tr>
<tr>
<td>DFH0CXXC</td>
<td>Sample</td>
<td>Keystroke overlap - COBOL</td>
<td>19</td>
</tr>
<tr>
<td>DFH0CZTK</td>
<td>Sample</td>
<td>FEPI sample: keystroke CONVERSE program (C/370)</td>
<td>03</td>
</tr>
<tr>
<td>DFH0CZXS</td>
<td>Sample</td>
<td>FEPI sample: setup program (C/370)</td>
<td>03</td>
</tr>
<tr>
<td>DFH0DCU5</td>
<td>Sample</td>
<td>CUA customer details file contents</td>
<td>05</td>
</tr>
<tr>
<td>DFH0DEL1</td>
<td>Sample</td>
<td>Sample program for CICS BTS</td>
<td>19</td>
</tr>
<tr>
<td>DFH0DHLP</td>
<td>Sample</td>
<td>CUA help file contents</td>
<td>04</td>
</tr>
<tr>
<td>DFH0DHTX</td>
<td>Sample</td>
<td>Sample EXITPGM Template</td>
<td></td>
</tr>
<tr>
<td>DFH0DLCC</td>
<td>Sample</td>
<td>CICS-DL/I program (CALL) - COBOL</td>
<td>19</td>
</tr>
<tr>
<td>DFH0DLCE</td>
<td>Sample</td>
<td>CICS-DL/I program (EXEC) - COBOL</td>
<td>19</td>
</tr>
<tr>
<td>DFH0DFORC</td>
<td>Sample</td>
<td>DB2 formatting program - COBOL</td>
<td>19</td>
</tr>
<tr>
<td>DFH0DMAP</td>
<td>Sample</td>
<td>Sample goodnight program map set</td>
<td></td>
</tr>
<tr>
<td>DFH0DGNIT</td>
<td>Sample</td>
<td>Sample goodnight transaction</td>
<td>19</td>
</tr>
<tr>
<td>DFH0DIV1</td>
<td>Sample</td>
<td>Sample program for CICS BTS</td>
<td>19</td>
</tr>
<tr>
<td>DFH0IZRI</td>
<td>Sample</td>
<td>FEPI sample: RDO data for back-end IMS</td>
<td>19</td>
</tr>
<tr>
<td>DFH0IZRQ</td>
<td>Sample</td>
<td>FEPI sample: RDM data for front-end CICS</td>
<td>19</td>
</tr>
<tr>
<td>DFH0JCS</td>
<td>Other</td>
<td>JCL to create CUA customer details file</td>
<td>02</td>
</tr>
<tr>
<td>DFH0JHLP</td>
<td>Other</td>
<td>JCL to create CUA help file</td>
<td>02</td>
</tr>
<tr>
<td>DFH0MAB</td>
<td>Sample</td>
<td>CUA abend handling - map set - COBOL</td>
<td>19</td>
</tr>
<tr>
<td>DFH0MABT</td>
<td>Sample</td>
<td>CUA about the sample application pop-up - map set - COBOL</td>
<td>19</td>
</tr>
<tr>
<td>DFH0MBRW</td>
<td>Sample</td>
<td>CUA browse customer details, base panel - map set - COBOL</td>
<td>19</td>
</tr>
<tr>
<td>DFH0MDDEL</td>
<td>Sample</td>
<td>CUA delete a customer record, base panel - map set - COBOL</td>
<td>19</td>
</tr>
<tr>
<td>DFH0MFPO</td>
<td>Sample</td>
<td>CUA file pull-down - map set - COBOL</td>
<td>19</td>
</tr>
<tr>
<td>DFH0MFLP</td>
<td>Sample</td>
<td>CUA help stub full-screen pop-up - map set - COBOL</td>
<td>19</td>
</tr>
<tr>
<td>DFH0MH</td>
<td>Sample</td>
<td>CUA contextual help pop-up - map set - COBOL</td>
<td>19</td>
</tr>
<tr>
<td>DFH0MHP</td>
<td>Sample</td>
<td>CUA help pull-down - map set - COBOL</td>
<td>19</td>
</tr>
<tr>
<td>DFH0MMLST</td>
<td>Sample</td>
<td>CUA list processing, base panel - map set - COBOL</td>
<td>19</td>
</tr>
<tr>
<td>DFH0MNEW</td>
<td>Sample</td>
<td>CUA new customer record, base panel - map set - COBOL</td>
<td>19</td>
</tr>
</tbody>
</table>
### CICS directory

**Table 123. CICS modules directory (continued)**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFH0MOPN</td>
<td>Sample</td>
<td>CUA file open pop-up - map set - COBOL</td>
<td>19 -</td>
</tr>
<tr>
<td>DFH0MPRT</td>
<td>Sample</td>
<td>CUA print pop-up - map set - COBOL</td>
<td>19 -</td>
</tr>
<tr>
<td>DFH0MSAS</td>
<td>Sample</td>
<td>CUA save changed customer record pop-up - map set - COBOL</td>
<td>19 -</td>
</tr>
<tr>
<td>DFH0MT1</td>
<td>Sample</td>
<td>CUA primary panel for sample application - map set - COBOL</td>
<td>19 -</td>
</tr>
<tr>
<td>DFH0MUPD</td>
<td>Sample</td>
<td>CUA update customer details, base panel - map set - COBOL</td>
<td>19 -</td>
</tr>
<tr>
<td>DFH0MZ1</td>
<td>Sample</td>
<td>FEPI sample: keystroke CONVERSE map (COBOL)</td>
<td>19 -</td>
</tr>
<tr>
<td>DFH0MZ2</td>
<td>Sample</td>
<td>FEPI sample: send/start &amp; receive map (COBOL)</td>
<td>19 -</td>
</tr>
<tr>
<td>DFH0MZ3</td>
<td>Sample</td>
<td>FEPI sample: map for back-end CICS application (Assembler)</td>
<td>19 -</td>
</tr>
<tr>
<td>DFH0MZ4</td>
<td>Sample</td>
<td>FEPI sample: SLU P one-out one-in map (Assembler)</td>
<td>19 -</td>
</tr>
<tr>
<td>DFH0MZ5</td>
<td>Sample</td>
<td>FEPI sample: SLU P pseudo-conversational map (COBOL)</td>
<td>19 -</td>
</tr>
<tr>
<td>DFH0MZ6</td>
<td>Sample</td>
<td>FEPI sample: keystroke CONVERSE map (C/370)</td>
<td>19 -</td>
</tr>
<tr>
<td>DFH0MZ7</td>
<td>Sample</td>
<td>FEPI sample: keystroke CONVERSE map (PL/I)</td>
<td>19 -</td>
</tr>
<tr>
<td>DFH0MZ8</td>
<td>Sample</td>
<td>FEPI sample: SLU P one-out one-in map (Assembler)</td>
<td>19 -</td>
</tr>
<tr>
<td>DFH0MZ9</td>
<td>Sample</td>
<td>FEPI sample: SLU P pseudo-conversational map (Assembler)</td>
<td>19 -</td>
</tr>
<tr>
<td>DFH0PAYC</td>
<td>Sample</td>
<td>Sample program for CICS BTS</td>
<td>19 -</td>
</tr>
<tr>
<td>DFH0PAYM</td>
<td>Sample</td>
<td>Sample program for CICS BTS</td>
<td>19 -</td>
</tr>
<tr>
<td>DFH0PAYO</td>
<td>Sample</td>
<td>Sample program for CICS BTS</td>
<td>19 -</td>
</tr>
<tr>
<td>DFH0PAY1</td>
<td>Sample</td>
<td>Sample program for CICS BTS</td>
<td>19 -</td>
</tr>
<tr>
<td>DFH0PS</td>
<td>Sample</td>
<td>Keystroke overlap/look-aside query - partition set - COBOL</td>
<td>19 -</td>
</tr>
<tr>
<td>DFH0PZTK</td>
<td>Sample</td>
<td>FEPI sample: keystroke CONVERSE program (PL/I)</td>
<td>19 -</td>
</tr>
<tr>
<td>DFH0RED1</td>
<td>Sample</td>
<td>Sample program for CICS BTS</td>
<td>19 -</td>
</tr>
<tr>
<td>DFH0REM1</td>
<td>Sample</td>
<td>Sample program for CICS BTS</td>
<td>19 -</td>
</tr>
<tr>
<td>DFH0SALC</td>
<td>Sample</td>
<td>Sample program for CICS BTS</td>
<td>19 -</td>
</tr>
<tr>
<td>DFH0SALM</td>
<td>Sample</td>
<td>Sample program for CICS BTS</td>
<td>19 -</td>
</tr>
<tr>
<td>DFH0SAL0</td>
<td>Sample</td>
<td>Sample program for CICS BTS</td>
<td>19 -</td>
</tr>
<tr>
<td>DFH0SAL1</td>
<td>Sample</td>
<td>Sample program for CICS BTS</td>
<td>19 -</td>
</tr>
<tr>
<td>DFH0SAL2</td>
<td>Sample</td>
<td>Sample program for CICS BTS</td>
<td>19 -</td>
</tr>
<tr>
<td>DFH0SET</td>
<td>Sample</td>
<td>Menu map for sample application</td>
<td>19 -</td>
</tr>
<tr>
<td>DFH0SINX</td>
<td>Sample</td>
<td>Rebuild primer index from master file</td>
<td>19 -</td>
</tr>
<tr>
<td>DFH0SIXR</td>
<td>Sample</td>
<td>Name index record for sample application</td>
<td>19 -</td>
</tr>
<tr>
<td>DFH0SREC</td>
<td>Sample</td>
<td>Account file record for sample application</td>
<td>19 -</td>
</tr>
<tr>
<td>DFH0STAT</td>
<td>Sample</td>
<td>Collect and print statistics - COBOL</td>
<td>C3 03</td>
</tr>
<tr>
<td>DFH0STCM</td>
<td>Sample</td>
<td>Statistics sample (DFH0STAT) Commarea</td>
<td>19 -</td>
</tr>
<tr>
<td>DFH0STLK</td>
<td>Sample</td>
<td>Collect and print stats map set - COBOL</td>
<td>C3 03</td>
</tr>
<tr>
<td>DFH0STM</td>
<td>Sample</td>
<td>Collect and print stats map set - CICS</td>
<td>19 03</td>
</tr>
<tr>
<td>DFH0STMD</td>
<td>Sample</td>
<td>Sample program for CICS BTS</td>
<td>19 -</td>
</tr>
<tr>
<td>DFH0STMU</td>
<td>Sample</td>
<td>Sample program for CICS BTS</td>
<td>19 -</td>
</tr>
<tr>
<td>DFH0STOC</td>
<td>Sample</td>
<td>Sample program for CICS BTS</td>
<td>19 -</td>
</tr>
<tr>
<td>DFH0STPR</td>
<td>Sample</td>
<td>Sample program for CICS BTS</td>
<td>19 03</td>
</tr>
<tr>
<td>DFH0STS</td>
<td>Sample</td>
<td>Statistics sample mapset - report selection</td>
<td>19 03</td>
</tr>
<tr>
<td>DFH0STSD</td>
<td>Sample</td>
<td>Sample program for CICS BTS</td>
<td>19 -</td>
</tr>
<tr>
<td>DFH0STSU</td>
<td>Sample</td>
<td>Sample program for CICS BTS</td>
<td>19 -</td>
</tr>
<tr>
<td>DFH0STSY</td>
<td>Sample</td>
<td>Sample program for CICS BTS</td>
<td>19 03</td>
</tr>
<tr>
<td>DFH0STTP</td>
<td>Sample</td>
<td>Sample program for CICS BTS</td>
<td>19 03</td>
</tr>
<tr>
<td>DFH0S00</td>
<td>Sample</td>
<td>Online account menu sample program</td>
<td>C3 -</td>
</tr>
</tbody>
</table>
### Table 123. CICS modules directory (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFH0S01</td>
<td>Sample</td>
<td>File inquire for sample application</td>
<td>C3 -</td>
</tr>
<tr>
<td>DFH0S02</td>
<td>Sample</td>
<td>File update for sample application</td>
<td>C3 -</td>
</tr>
<tr>
<td>DFH0S03</td>
<td>Sample</td>
<td>Print customer record for sample application</td>
<td>C3 -</td>
</tr>
<tr>
<td>DFH0S04</td>
<td>Sample</td>
<td>Error routine for sample application</td>
<td>C3 -</td>
</tr>
<tr>
<td>DFH0VAB</td>
<td>Sample</td>
<td>CUA abend handler - COBOL</td>
<td>C3 -</td>
</tr>
<tr>
<td>DFH0VABT</td>
<td>Sample</td>
<td>CUA about pop-up handler - COBOL</td>
<td>C3 -</td>
</tr>
<tr>
<td>DFH0VBRW</td>
<td>Sample</td>
<td>CUA browse customer details processing - COBOL</td>
<td>C3 -</td>
</tr>
<tr>
<td>DFH0VDEL</td>
<td>Sample</td>
<td>CUA delete customer details processing - COBOL</td>
<td>C3 -</td>
</tr>
<tr>
<td>DFH0VDQ</td>
<td>Sample</td>
<td>CUA temporary-storage cleanup - COBOL</td>
<td>C3 -</td>
</tr>
<tr>
<td>DFH0VHLP</td>
<td>Sample</td>
<td>CUA help pop-up handler - COBOL</td>
<td>C3 -</td>
</tr>
<tr>
<td>DFH0VHP</td>
<td>Sample</td>
<td>CUA contextual help pop-up handler - COBOL</td>
<td>C3 -</td>
</tr>
<tr>
<td>DFH0VLO</td>
<td>Sample</td>
<td>CUA help file handler - COBOL</td>
<td>C3 -</td>
</tr>
<tr>
<td>DFH0VLST</td>
<td>Sample</td>
<td>CUA list panel handler - COBOL</td>
<td>C3 -</td>
</tr>
<tr>
<td>DFH0VNEW</td>
<td>Sample</td>
<td>CUA new customer panel processing - COBOL</td>
<td>C3 -</td>
</tr>
<tr>
<td>DFH0VOL</td>
<td>Sample</td>
<td>CUA overlay handler - COBOL</td>
<td>C3 -</td>
</tr>
<tr>
<td>DFH0VQPN</td>
<td>Sample</td>
<td>CUA file open pop-up handler - COBOL</td>
<td>C3 -</td>
</tr>
<tr>
<td>DFH0VPRT</td>
<td>Sample</td>
<td>CUA print pop-up handler - COBOL</td>
<td>C3 -</td>
</tr>
<tr>
<td>DFH0VRIQ</td>
<td>Sample</td>
<td>CUA customer detail file handler - COBOL</td>
<td>C3 -</td>
</tr>
<tr>
<td>DFH0VSAS</td>
<td>Sample</td>
<td>CUA save customer details pop-up handler - COBOL</td>
<td>C3 -</td>
</tr>
<tr>
<td>DFH0VTR</td>
<td>Sample</td>
<td>CUA table router - COBOL</td>
<td>C3 -</td>
</tr>
<tr>
<td>DFH0VTR1</td>
<td>Sample</td>
<td>CUA primary panel processing - COBOL</td>
<td>C3 -</td>
</tr>
<tr>
<td>DFH0VUPD</td>
<td>Sample</td>
<td>CUA update customer record processing - COBOL</td>
<td>C3 -</td>
</tr>
<tr>
<td>DFH0VZPA</td>
<td>Sample</td>
<td>FEPI sample: SLU P pseudo-conversational program (COBOL)</td>
<td>C3 -</td>
</tr>
<tr>
<td>DFH0VZPS</td>
<td>Sample</td>
<td>FEPI sample: SLU P one-out one-in program (COBOL)</td>
<td>C3 -</td>
</tr>
<tr>
<td>DFH0VQS</td>
<td>Sample</td>
<td>FEPI sample: STSN handler (COBOL)</td>
<td>C3 -</td>
</tr>
<tr>
<td>DFH0VZTD</td>
<td>Sample</td>
<td>FEPI sample: 3270 data stream pass through (COBOL)</td>
<td>C3 -</td>
</tr>
<tr>
<td>DFH0VZTK</td>
<td>Sample</td>
<td>FEPI sample: keystroke CONVERSE program (COBOL)</td>
<td>C3 -</td>
</tr>
<tr>
<td>DFH0VZTR</td>
<td>Sample</td>
<td>FEPI sample: screen image RECEIVE/EXTRACT FIELD (COBOL)</td>
<td>C3 -</td>
</tr>
<tr>
<td>DFH0VZTS</td>
<td>Sample</td>
<td>FEPI sample: screen image SEND/START (COBOL)</td>
<td>C3 -</td>
</tr>
<tr>
<td>DFH0VZUC</td>
<td>Sample</td>
<td>FEPI sample: begin session handler (COBOL)</td>
<td>C3 -</td>
</tr>
<tr>
<td>DFH0VZUU</td>
<td>Sample</td>
<td>FEPI sample: end session handler (COBOL)</td>
<td>C3 -</td>
</tr>
<tr>
<td>DFH0VUX</td>
<td>Sample</td>
<td>FEPI sample: monitor unsolicited data handler (COBOL)</td>
<td>C3 -</td>
</tr>
<tr>
<td>DFH0VZXS</td>
<td>Sample</td>
<td>FEPI sample: setup program (COBOL)</td>
<td>C3 -</td>
</tr>
<tr>
<td>DFH0WBCA</td>
<td>Sample</td>
<td>Sample Client Authentication Program</td>
<td>- 19</td>
</tr>
<tr>
<td>DFH29BO</td>
<td>Symbolic</td>
<td>Special characters for 2980</td>
<td>C2 07</td>
</tr>
<tr>
<td>DFH30SS</td>
<td>Sample</td>
<td>Dyn alloc - JCL to build sample program</td>
<td>02 -</td>
</tr>
<tr>
<td>DFH99CC</td>
<td>Sample</td>
<td>Dyn alloc - character and numeric string conversion</td>
<td>19 03</td>
</tr>
<tr>
<td>DFH99BD</td>
<td>Other</td>
<td>Dyn alloc - JCL to build sample program</td>
<td>- 03</td>
</tr>
</tbody>
</table>
## CICS directory

### Table 123. CICS modules directory (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFH99DY</td>
<td>Sample</td>
<td>Dyn alloc - issue SVC and analyze</td>
<td>19 03</td>
</tr>
<tr>
<td>DFH99FP</td>
<td>Sample</td>
<td>Dyn alloc - process function keyword</td>
<td>19 03</td>
</tr>
<tr>
<td>DFH99GI</td>
<td>Sample</td>
<td>Dyn alloc - format display and get input</td>
<td>19 03</td>
</tr>
<tr>
<td>DFH99KC</td>
<td>Sample</td>
<td>Dyn alloc - keyword value conversion</td>
<td>19 03</td>
</tr>
<tr>
<td>DFH99KH</td>
<td>Sample</td>
<td>Dyn alloc - list keywords for help</td>
<td>19 03</td>
</tr>
<tr>
<td>DFH99KO</td>
<td>Sample</td>
<td>Dyn alloc - process operator keywords</td>
<td>19 03</td>
</tr>
<tr>
<td>DFH99KR</td>
<td>Sample</td>
<td>Dyn alloc - convert returned value to keyword</td>
<td>19 03</td>
</tr>
<tr>
<td>DFH99LK</td>
<td>Sample</td>
<td>Dyn alloc - search key set for given token</td>
<td>19 03</td>
</tr>
<tr>
<td>DFH99M</td>
<td>Sample</td>
<td>Dyn alloc - macro</td>
<td>11 -</td>
</tr>
<tr>
<td>DFH99MAC</td>
<td>Sample</td>
<td>Dyn alloc - macro</td>
<td>19 -</td>
</tr>
<tr>
<td>DFH99ML</td>
<td>Sample</td>
<td>Dyn alloc - build message text from token list</td>
<td>19 03</td>
</tr>
<tr>
<td>DFH99MM</td>
<td>Sample</td>
<td>Dyn alloc - main control program</td>
<td>19 03</td>
</tr>
<tr>
<td>DFH99MP</td>
<td>Sample</td>
<td>Dyn alloc - message filing routine</td>
<td>19 03</td>
</tr>
<tr>
<td>DFH99MT</td>
<td>Sample</td>
<td>Dyn alloc - match abbreviation with keyword</td>
<td>19 03</td>
</tr>
<tr>
<td>DFH99RP</td>
<td>Sample</td>
<td>Dyn alloc - process returned values</td>
<td>19 03</td>
</tr>
<tr>
<td>DFH99SVC</td>
<td>Sample</td>
<td>Dyn alloc - SVC services</td>
<td>19 -</td>
</tr>
<tr>
<td>DFH99T</td>
<td>Sample</td>
<td>Dyn alloc - table of keywords</td>
<td>19 03</td>
</tr>
<tr>
<td>DFH99TK</td>
<td>Sample</td>
<td>Dyn alloc - tokenize input command</td>
<td>19 03</td>
</tr>
<tr>
<td>DFH99TX</td>
<td>Sample</td>
<td>Dyn alloc - text display routine</td>
<td>19 03</td>
</tr>
<tr>
<td>DFH99VH</td>
<td>Sample</td>
<td>Dyn alloc - list description for help</td>
<td>19 03</td>
</tr>
<tr>
<td>DFHGAALL</td>
<td>Sample</td>
<td>Inquiry/update</td>
<td>19 03</td>
</tr>
<tr>
<td>DFHGABRW</td>
<td>Sample</td>
<td>Browse</td>
<td>19 03</td>
</tr>
<tr>
<td>DFHGAODM</td>
<td>Sample</td>
<td>Order entry queue print</td>
<td>19 03</td>
</tr>
<tr>
<td>DFHGADSP</td>
<td>Sample</td>
<td>XRF overseer - display status</td>
<td>19 03</td>
</tr>
<tr>
<td>DFHGAFL</td>
<td>Sample</td>
<td>Customer file (FILEA) record layout</td>
<td>19 -</td>
</tr>
<tr>
<td>DFHGAGA</td>
<td>Sample</td>
<td>Generated version of DFHUGAMA</td>
<td>19 03</td>
</tr>
<tr>
<td>DFHGAGB</td>
<td>Sample</td>
<td>Generated version of DFHUGAMB</td>
<td>19 03</td>
</tr>
<tr>
<td>DFHGAGC</td>
<td>Sample</td>
<td>Generated version of DFHUGAMC</td>
<td>19 03</td>
</tr>
<tr>
<td>DFHGAGCB</td>
<td>Sample</td>
<td>XRF overseer - set up RPL</td>
<td>19 03</td>
</tr>
<tr>
<td>DFHGAGD</td>
<td>Sample</td>
<td>Generated version of DFHUGAMD</td>
<td>19 03</td>
</tr>
<tr>
<td>DFHGAGK</td>
<td>Sample</td>
<td>Generated version of DFHUGAMK</td>
<td>19 03</td>
</tr>
<tr>
<td>DFHGAGL</td>
<td>Sample</td>
<td>Generated version of DFHUGAML</td>
<td>19 03</td>
</tr>
<tr>
<td>DFHGALOG</td>
<td>Sample</td>
<td>Audit trail (log) record layout</td>
<td>19 -</td>
</tr>
<tr>
<td>DFHGAL86</td>
<td>Sample</td>
<td>Order entry queue record layout</td>
<td>19 -</td>
</tr>
<tr>
<td>DFHUGAMA</td>
<td>Sample</td>
<td>Operator instructions map set</td>
<td>19 -</td>
</tr>
<tr>
<td>DFHUGAMAU</td>
<td>Sample</td>
<td>Operator instructions map set</td>
<td>19 -</td>
</tr>
<tr>
<td>DFHUGAMB</td>
<td>Sample</td>
<td>Customer details map set</td>
<td>19 -</td>
</tr>
<tr>
<td>DFHUGAMBU</td>
<td>Sample</td>
<td>Customer details map set</td>
<td>19 -</td>
</tr>
<tr>
<td>DFHGAMC</td>
<td>Sample</td>
<td>File browse map set</td>
<td>19 -</td>
</tr>
<tr>
<td>DFHGAMCU</td>
<td>Sample</td>
<td>File browse map set</td>
<td>19 -</td>
</tr>
<tr>
<td>DFHGAMD</td>
<td>Sample</td>
<td>Low balance inquiry map set</td>
<td>19 -</td>
</tr>
<tr>
<td>DFHGAMDU</td>
<td>Sample</td>
<td>Web Interface BMS screen emulation</td>
<td>19 -</td>
</tr>
<tr>
<td>DFHGAMK</td>
<td>Sample</td>
<td>Order entry map set</td>
<td>19 -</td>
</tr>
<tr>
<td>DFHGAMKU</td>
<td>Sample</td>
<td>Order entry map set</td>
<td>19 -</td>
</tr>
<tr>
<td>DFHGAML</td>
<td>Sample</td>
<td>Order report map set</td>
<td>19 -</td>
</tr>
<tr>
<td>DFHGAMNU</td>
<td>Sample</td>
<td>Operator instructions</td>
<td>19 03</td>
</tr>
<tr>
<td>DFHGAREN</td>
<td>Sample</td>
<td>Order entry</td>
<td>19 03</td>
</tr>
<tr>
<td>DFHGAREP</td>
<td>Sample</td>
<td>Low balance inquiry</td>
<td>19 03</td>
</tr>
<tr>
<td>DFHGARES</td>
<td>Sample</td>
<td>XRF overseer - restart failed region</td>
<td>19 03</td>
</tr>
<tr>
<td>DFHGATXC</td>
<td>Sample</td>
<td>EXCI batch client program (Assembler)</td>
<td>19 03</td>
</tr>
<tr>
<td>DFHGAXCC</td>
<td>Sample</td>
<td>EXCI batch client program (Assembler)</td>
<td>19 03</td>
</tr>
<tr>
<td>DFHGAXCS</td>
<td>Sample</td>
<td>EXCI batch server program (Assembler)</td>
<td>19 03</td>
</tr>
</tbody>
</table>

1520 CICS Transaction Server: CICS Diagnosis Reference
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFGHAXRO</td>
<td>Sample</td>
<td>XRF overseer program</td>
<td>19 03</td>
</tr>
<tr>
<td>DFGHAXVS</td>
<td>Sample</td>
<td>EXCI sample server</td>
<td>19 03</td>
</tr>
<tr>
<td>DFGHBMXT</td>
<td>Sample</td>
<td>Sample BMS global user exit</td>
<td>19 -</td>
</tr>
<tr>
<td>DFGHBATCH</td>
<td>Sample</td>
<td>Batch test data for DFHIVPBT</td>
<td>19 -</td>
</tr>
<tr>
<td>DFGHGCAT1</td>
<td>Sample</td>
<td>CLIST to create RACF profiles for CICS category 1 transactions</td>
<td>19 -</td>
</tr>
<tr>
<td>DFGHGCAT2</td>
<td>Sample</td>
<td>CLIST to create RACF profiles for CICS category 2 transactions</td>
<td>19 -</td>
</tr>
<tr>
<td>DFGHGCESD</td>
<td>Sample</td>
<td>Shutdown assist program</td>
<td>P3 -</td>
</tr>
<tr>
<td>DFGHCRFA</td>
<td>Sample</td>
<td>CSD cross-reference program</td>
<td>19 03</td>
</tr>
<tr>
<td>DFGHCRFP</td>
<td>Sample</td>
<td>CSD cross-reference program - PL/I</td>
<td>P3 -</td>
</tr>
<tr>
<td>DFGHCSDU</td>
<td>Sample</td>
<td>RDO offline utilities</td>
<td>19 -</td>
</tr>
<tr>
<td>DFGHCUS1</td>
<td>Sample</td>
<td>CSDUP invocation from TSO environment</td>
<td>19 03</td>
</tr>
<tr>
<td>DFGHDFALL</td>
<td>Sample</td>
<td>Inquiry/update - C/370</td>
<td>D3 -</td>
</tr>
<tr>
<td>DFGHDBAN</td>
<td>Sample</td>
<td>Batch test data for DFHIVPDB (Assembler)</td>
<td>19 -</td>
</tr>
<tr>
<td>DFGHDBCB</td>
<td>Sample</td>
<td>Batch test data for DFHIVPDB (Cobol)</td>
<td>19 -</td>
</tr>
<tr>
<td>DFGHDBPL</td>
<td>Sample</td>
<td>Batch test data for DFHIVPDB (PL/I)</td>
<td>19 -</td>
</tr>
<tr>
<td>DFGHDBRM</td>
<td>Sample</td>
<td>Browse - C/370</td>
<td>D3 -</td>
</tr>
<tr>
<td>DFGHDBRT2</td>
<td>Sample</td>
<td>DB2 table definitions for DFGHDFORx</td>
<td>19 -</td>
</tr>
<tr>
<td>DFGHDCOM</td>
<td>Sample</td>
<td>Order entry queue print - C/370</td>
<td>D3 -</td>
</tr>
<tr>
<td>DFGHDCDO</td>
<td>Sample</td>
<td>DCT SDSCI entries</td>
<td>19 -</td>
</tr>
<tr>
<td>DFGHDCET</td>
<td>Sample</td>
<td>DCT entries for basic facilities</td>
<td>19 -</td>
</tr>
<tr>
<td>DFGHDCST</td>
<td>Sample</td>
<td>DCT entries for sample applications</td>
<td>19 -</td>
</tr>
<tr>
<td>DFGHDFIL</td>
<td>Sample</td>
<td>Customer file (FILEA) record layout -C/370</td>
<td>D3 -</td>
</tr>
<tr>
<td>DFGHDLAC</td>
<td>Sample</td>
<td>CICS-DL/I program using CALL interface</td>
<td>19 03</td>
</tr>
<tr>
<td>DFGHDLAE</td>
<td>Sample</td>
<td>CICS-DL/I program using EXEC DLI</td>
<td>19 03</td>
</tr>
<tr>
<td>DFGHDLPC</td>
<td>Sample</td>
<td>CICS-DL/I program (CALL) - PL/I</td>
<td>P3 -</td>
</tr>
<tr>
<td>DFGHDLPE</td>
<td>Sample</td>
<td>CICS-DL/I program (EXEC) - PL/I</td>
<td>P3 -</td>
</tr>
<tr>
<td>DFGHDLB6</td>
<td>Sample</td>
<td>Order entry queue record layout - C/370</td>
<td>D3 -</td>
</tr>
<tr>
<td>DFGHDMA</td>
<td>Sample</td>
<td>Operator instructions map set - C/370</td>
<td>19 -</td>
</tr>
<tr>
<td>DFGHDDB</td>
<td>Sample</td>
<td>Customer details map set - C/370</td>
<td>19 -</td>
</tr>
<tr>
<td>DFGHDMS</td>
<td>Sample</td>
<td>File browse map set - C/370</td>
<td>19 -</td>
</tr>
<tr>
<td>DFGHDMO</td>
<td>Sample</td>
<td>Low balance inquiry map set - C/370</td>
<td>19 -</td>
</tr>
<tr>
<td>DFGHDMK</td>
<td>Sample</td>
<td>Order entry map set - C/370</td>
<td>19 -</td>
</tr>
<tr>
<td>DFGHDMN</td>
<td>Sample</td>
<td>Order report map set - C/370</td>
<td>19 -</td>
</tr>
<tr>
<td>DFGHDMNU</td>
<td>Sample</td>
<td>Operator instructions - C/370</td>
<td>D3 -</td>
</tr>
<tr>
<td>DFGHDREN</td>
<td>Sample</td>
<td>Order entry - C/370</td>
<td>D3 -</td>
</tr>
<tr>
<td>DFGHDERP</td>
<td>Sample</td>
<td>Low balance inquiry - C/370</td>
<td>D3 -</td>
</tr>
<tr>
<td>DFGHDTLJC</td>
<td>Sample</td>
<td>Shared Data Tables XDTLC exit program</td>
<td>19 -</td>
</tr>
<tr>
<td>DFGHDTRD</td>
<td>Sample</td>
<td>Shared data tables XOTAD exit program</td>
<td>19 -</td>
</tr>
<tr>
<td>DFGHDXCC</td>
<td>Sample</td>
<td>Batch Client Program (C/370)</td>
<td>D3 -</td>
</tr>
<tr>
<td>DFGHDXVC</td>
<td>Sample</td>
<td>EXCI client program - Java environment</td>
<td>19 03</td>
</tr>
<tr>
<td>DFGHFAIN</td>
<td>Sample</td>
<td>Data for batch load of FILEA</td>
<td>19 -</td>
</tr>
<tr>
<td>DFGHFCBF</td>
<td>Sample</td>
<td>Sample XFCBFFAIL exit program</td>
<td>19 -</td>
</tr>
<tr>
<td>DFGHFCBV</td>
<td>Sample</td>
<td>Sample XFCBOVER exit program</td>
<td>19 -</td>
</tr>
<tr>
<td>DFGHFCFLO</td>
<td>Sample</td>
<td>Sample XFCFLDEL exit program</td>
<td>19 -</td>
</tr>
<tr>
<td>DFGHFOFARA</td>
<td>Sample</td>
<td>DB2 formatting program</td>
<td>19 03</td>
</tr>
<tr>
<td>DFGHFOFRP</td>
<td>Sample</td>
<td>DB2 formatting program - PL/I</td>
<td>P3 -</td>
</tr>
<tr>
<td>DFGHGMAP</td>
<td>Sample</td>
<td>Sample goodnight transaction BMS map</td>
<td>19 -</td>
</tr>
<tr>
<td>DFGHICCN</td>
<td>Sample</td>
<td>Call to CPSM to issue cancel command</td>
<td>19 -</td>
</tr>
<tr>
<td>DFGHICIC</td>
<td>Sample</td>
<td>CICS-CICS or CICS-IMS conversation</td>
<td>19 03</td>
</tr>
<tr>
<td>DFGHIFBL</td>
<td>Sample</td>
<td>Remote file browse - local processing</td>
<td>19 03</td>
</tr>
<tr>
<td>DFGHIFBR</td>
<td>Sample</td>
<td>Remote file browse - remote processing</td>
<td>19 03</td>
</tr>
</tbody>
</table>
CICS directory

<table>
<thead>
<tr>
<th>Name</th>
<th>Sample</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHGiGB</td>
<td>Sample</td>
<td>Generated version of DFHGiMB</td>
<td>19 03</td>
</tr>
<tr>
<td>DFHGiGC</td>
<td>Sample</td>
<td>Generated version of DFHGiMC</td>
<td>19 03</td>
</tr>
<tr>
<td>DFHGiGS</td>
<td>Sample</td>
<td>Generated version of DFHGiMS</td>
<td>19 03</td>
</tr>
<tr>
<td>DFHGiGX</td>
<td>Sample</td>
<td>Generated version of DFHGiMX</td>
<td>19 03</td>
</tr>
<tr>
<td>DFHGiG1</td>
<td>Sample</td>
<td>Generated version of DFHGiM1</td>
<td>19 03</td>
</tr>
<tr>
<td>DFHGiG2</td>
<td>Sample</td>
<td>Generated version of DFHGiM2</td>
<td>19 03</td>
</tr>
<tr>
<td>DFHGiIAT</td>
<td>Sample</td>
<td>IIOP banking sample app.to C Account</td>
<td>19 -</td>
</tr>
<tr>
<td>DFHGiIBI</td>
<td>Sample</td>
<td>IIOP banking sample app.to C Init.</td>
<td>19 -</td>
</tr>
<tr>
<td>DFHGiIBQ</td>
<td>Sample</td>
<td>IIOP banking sample app.to C Query</td>
<td>19 -</td>
</tr>
<tr>
<td>DFHGiICC</td>
<td>Sample</td>
<td>IIOP banking sample app.to C Credit Check</td>
<td>19 -</td>
</tr>
<tr>
<td>DFHGiIICH</td>
<td>Sample</td>
<td>IIOP banking sample app.to C Cr.Chk commarea</td>
<td>19 -</td>
</tr>
<tr>
<td>DFHGiIIMA</td>
<td>Sample</td>
<td>IIOP banking sample app.to C BMS Map</td>
<td>19 -</td>
</tr>
<tr>
<td>DFHGiIIMR</td>
<td>Sample</td>
<td>IIOP banking sample app.to C Comm_struct</td>
<td>19 -</td>
</tr>
<tr>
<td>DFHGiIMB</td>
<td>Sample</td>
<td>Remote file browse - map set</td>
<td>19 -</td>
</tr>
<tr>
<td>DFHGiIMC</td>
<td>Sample</td>
<td>CICS-CICS or CICS-IMS conversation - map set</td>
<td>19 -</td>
</tr>
<tr>
<td>DFHGiIMS</td>
<td>Sample</td>
<td>CICS-IMS conversation/demand paged output - map set</td>
<td>19 -</td>
</tr>
<tr>
<td>DFHGiIMSN</td>
<td>Sample</td>
<td>CICS-IMS conversation</td>
<td>19 03</td>
</tr>
<tr>
<td>DFHGiIMSO</td>
<td>Sample</td>
<td>CICS-IMS demand paged output</td>
<td>19 03</td>
</tr>
<tr>
<td>DFHGiIMX</td>
<td>Sample</td>
<td>Local to remote temporary-storage queue transfer - map set</td>
<td>19 -</td>
</tr>
<tr>
<td>DFHGiIM1</td>
<td>Sample</td>
<td>TS record retrieval - map set 1</td>
<td>19 -</td>
</tr>
<tr>
<td>DFHGiIM2</td>
<td>Sample</td>
<td>TS record retrieval - map set 2</td>
<td>19 -</td>
</tr>
<tr>
<td>DFHGiIQRD</td>
<td>Sample</td>
<td>TS record retrieval - local display</td>
<td>19 03</td>
</tr>
<tr>
<td>DFHGiIQL</td>
<td>Sample</td>
<td>TS record retrieval - local request</td>
<td>19 03</td>
</tr>
<tr>
<td>DFHGiIQR</td>
<td>Sample</td>
<td>TS record retrieval - remote request</td>
<td>19 03</td>
</tr>
<tr>
<td>DFHGiIQXL</td>
<td>Sample</td>
<td>Local to remote temporary-storage queue transfer - local processing</td>
<td>19 03</td>
</tr>
<tr>
<td>DFHGiIQXR</td>
<td>Sample</td>
<td>Local to remote temporary-storage queue transfer - remote processing</td>
<td>19 03</td>
</tr>
<tr>
<td>DFHGiJSAM</td>
<td>Sample</td>
<td>Java Sample Linker in C</td>
<td>19 -</td>
</tr>
<tr>
<td>DFHGiLLCA</td>
<td>Sample</td>
<td>Java Sample COMMAREA checker in C</td>
<td>19 -</td>
</tr>
<tr>
<td>DFHGiLDSP</td>
<td>Sample</td>
<td>Create FILEA data file</td>
<td>19 03</td>
</tr>
<tr>
<td>DFHGiLGLS</td>
<td>Sample</td>
<td>Sample GLUE program for XLGSTRM</td>
<td>19 -</td>
</tr>
<tr>
<td>DFHGiMC</td>
<td>Sample</td>
<td>MCT entry for DBCTL</td>
<td>19 -</td>
</tr>
<tr>
<td>DFHGiMOLS</td>
<td>Sample</td>
<td>Offline processor of monitoring data</td>
<td>19 03</td>
</tr>
<tr>
<td>DFHGiOFAR</td>
<td>Sample</td>
<td>19 -</td>
<td></td>
</tr>
<tr>
<td>DFHGiPALL</td>
<td>Sample</td>
<td>Inquiry/update - PL/I</td>
<td>P3 -</td>
</tr>
<tr>
<td>DFHGiPB</td>
<td>Sample</td>
<td>Browse - PL/I</td>
<td>P3 -</td>
</tr>
<tr>
<td>DFHGiPCX</td>
<td>Sample</td>
<td>XPCFTCH global user exit program</td>
<td>19 03</td>
</tr>
<tr>
<td>DFHGiPCA</td>
<td>Sample</td>
<td>Global work area for DFHGiPCX</td>
<td>19 -</td>
</tr>
<tr>
<td>DFHGiPCE</td>
<td>Sample</td>
<td>Order entry queue print - PL/I</td>
<td>P3 -</td>
</tr>
<tr>
<td>DFHGiPCP</td>
<td>Sample</td>
<td>Enabling program for DFHGiPCX and DFHGiZCAT</td>
<td>19 03</td>
</tr>
<tr>
<td>DFHGiPCE</td>
<td>Sample</td>
<td>DFHGiPCX global user exit invocation</td>
<td>19 03</td>
</tr>
<tr>
<td>DFHGiECTA</td>
<td>Sample</td>
<td>XPCTA user exit program</td>
<td>19 -</td>
</tr>
<tr>
<td>DFHGiPPE</td>
<td>Sample</td>
<td>Dummy main program for PL/I programs - using CSD offline extract function</td>
<td>P3 -</td>
</tr>
<tr>
<td>DFHGiPFL</td>
<td>Sample</td>
<td>Customer file (FILEA) record layout - PL/I</td>
<td>P3 -</td>
</tr>
<tr>
<td>DFHGiPLOG</td>
<td>Sample</td>
<td>Audit trail (log) record layout - PL/I</td>
<td>P3 -</td>
</tr>
<tr>
<td>DFHGiPL86</td>
<td>Sample</td>
<td>Order entry queue record layout - PL/I</td>
<td>P3 -</td>
</tr>
<tr>
<td>DFHGiPM</td>
<td>Sample</td>
<td>Operator instructions map set - PL/I</td>
<td>19 -</td>
</tr>
<tr>
<td>DFHGiPM</td>
<td>Sample</td>
<td>Customer details map set - PL/I</td>
<td>19 -</td>
</tr>
<tr>
<td>DFHGiPM</td>
<td>Sample</td>
<td>File browse map set - PL/I</td>
<td>19 -</td>
</tr>
<tr>
<td>DFHGiPM</td>
<td>Sample</td>
<td>Low balance inquiry map set - PL/I</td>
<td>19 -</td>
</tr>
<tr>
<td>DFHGiPM</td>
<td>Sample</td>
<td>Order entry map set - PL/I</td>
<td>19 -</td>
</tr>
</tbody>
</table>
Table 123. CICS modules directory (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFGPML</td>
<td>Sample</td>
<td>Order report map set - PL/I</td>
<td>19 -</td>
</tr>
<tr>
<td>DFGPMMNU</td>
<td>Sample</td>
<td>Operator instructions - PL/I</td>
<td>P3 -</td>
</tr>
<tr>
<td>DFGPMP</td>
<td>Sample</td>
<td>Keystroke overlap/look-aside query - map set - PL/I</td>
<td>19 -</td>
</tr>
<tr>
<td>DFGPPOKO</td>
<td>Sample</td>
<td>Keystroke overlap - PL/I</td>
<td>P3 -</td>
</tr>
<tr>
<td>DFGPPLA</td>
<td>Sample</td>
<td>Look-aside query - PL/I</td>
<td>P3 -</td>
</tr>
<tr>
<td>DFGPREN</td>
<td>Sample</td>
<td>Order entry - PL/I</td>
<td>P3 -</td>
</tr>
<tr>
<td>DFGPREP</td>
<td>Sample</td>
<td>Low balance inquiry PL/I</td>
<td>P3 -</td>
</tr>
<tr>
<td>DFGPS</td>
<td>Sample</td>
<td>Keystroke overlap/look-aside query - partition set - PL/I</td>
<td>19 -</td>
</tr>
<tr>
<td>DFGPXXXCC</td>
<td>Sample</td>
<td>Batch client program (PL/I)</td>
<td>P3 -</td>
</tr>
<tr>
<td>DFGRACF</td>
<td>Sample</td>
<td>RACF class descriptor table</td>
<td>19 -</td>
</tr>
<tr>
<td>DFGRITNG</td>
<td>Sample</td>
<td>Build KEYRING profiles in RACF</td>
<td>19 -</td>
</tr>
<tr>
<td>DFGSIPA</td>
<td>Other</td>
<td>System initialization parameters for use with AOR and default SIT</td>
<td>19 -</td>
</tr>
<tr>
<td>DFGSIPD</td>
<td>Other</td>
<td>System initialization parameters for use with DOR and default SIT</td>
<td>19 -</td>
</tr>
<tr>
<td>DFGSIPPT</td>
<td>Other</td>
<td>System initialization parameters for use with TOR and default SIT</td>
<td>19 -</td>
</tr>
<tr>
<td>DFGSIP1</td>
<td>Other</td>
<td>System initialization parameters for use by DFHIVPOL (online IVP)</td>
<td>19 -</td>
</tr>
<tr>
<td>DFGSIP2</td>
<td>Other</td>
<td>System initialization parameters for use by DFHIVPBT (batch IVP)</td>
<td>19 -</td>
</tr>
<tr>
<td>DFGSIP5</td>
<td>Other</td>
<td>System initialization parameters for use by DFHIVPDB (DBCTL IVP)</td>
<td>19 -</td>
</tr>
<tr>
<td>DFGSINPW</td>
<td>Sample</td>
<td>Password expiration mgement for Windows/NT</td>
<td>19 -</td>
</tr>
<tr>
<td>DFGSINP2</td>
<td>Sample</td>
<td>Password expiration mgement for OS/2 Warp</td>
<td>19 -</td>
</tr>
<tr>
<td>DFGSOSLT</td>
<td>Sample</td>
<td>Input for DB2 table load utility</td>
<td>19 -</td>
</tr>
<tr>
<td>DFGSTAS</td>
<td>Sample</td>
<td>DFGHOSTAT storage statistics subroutine</td>
<td>19 03</td>
</tr>
<tr>
<td>DFGSTCN</td>
<td>Sample</td>
<td>DFGHOSTAT time calculations subroutine</td>
<td>19 03</td>
</tr>
<tr>
<td>DFGSTEED</td>
<td>Sample</td>
<td>Stagger end-of-day time for statistics</td>
<td>19 03</td>
</tr>
<tr>
<td>DFGSTER</td>
<td>Sample</td>
<td>PLT program to print recovery statistics on CICS emergency restart</td>
<td>19 03</td>
</tr>
<tr>
<td>DFGSSTTB</td>
<td>Sample</td>
<td>Statistics sample user exit ID table</td>
<td>19 03</td>
</tr>
<tr>
<td>DFGSXXP1</td>
<td>Sample</td>
<td>Suppress message by number (user exit)</td>
<td>19 03</td>
</tr>
<tr>
<td>DFGSXXP2</td>
<td>Sample</td>
<td>Suppress message by destination route code</td>
<td>19 03</td>
</tr>
<tr>
<td>DFGSXXP3</td>
<td>Sample</td>
<td>Suppress message by transient data queue</td>
<td>19 03</td>
</tr>
<tr>
<td>DFGSXXP4</td>
<td>Sample</td>
<td>Reroute console message to transient data queue</td>
<td>19 03</td>
</tr>
<tr>
<td>DFGSXXP5</td>
<td>Sample</td>
<td>Reroute message from one transient data queue to another</td>
<td>19 03</td>
</tr>
<tr>
<td>DFGSXXP6</td>
<td>Sample</td>
<td>Reroute message from transient data queue to list of consoles</td>
<td>19 03</td>
</tr>
<tr>
<td>DFGTCTS</td>
<td>Sample</td>
<td>TCT entries for sequential (CRLP) terminals</td>
<td>19 -</td>
</tr>
<tr>
<td>DFGTDWT</td>
<td>Sample</td>
<td>Transient data write to terminal</td>
<td>19 03</td>
</tr>
<tr>
<td>DFGULPA</td>
<td>Other</td>
<td>Placeholder for DFGULPA</td>
<td>19 -</td>
</tr>
<tr>
<td>DFGUMOD</td>
<td>Other</td>
<td>SMP/E USERMOD to move LPA-eligible CICS modules into LPA library</td>
<td>19 -</td>
</tr>
<tr>
<td>DFGWBAU</td>
<td>Sample</td>
<td>Web module</td>
<td>19 03</td>
</tr>
<tr>
<td>DFGWBSA</td>
<td>Sample</td>
<td>Web module</td>
<td>19 03</td>
</tr>
<tr>
<td>DFGWBSB</td>
<td>Sample</td>
<td>Web module</td>
<td>19 03</td>
</tr>
<tr>
<td>DFGWBCS</td>
<td>Sample</td>
<td>Web module</td>
<td>19 03</td>
</tr>
<tr>
<td>DFGWBSN</td>
<td>Sample</td>
<td>Web module</td>
<td>19 03</td>
</tr>
<tr>
<td>DFGWBSR</td>
<td>Sample</td>
<td>Web module</td>
<td>19 03</td>
</tr>
</tbody>
</table>
## CICS directory

### Table 123. CICS modules directory (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHGWBST</td>
<td>Sample</td>
<td>Web module</td>
<td>19 03</td>
</tr>
<tr>
<td>DFHGWBI1A</td>
<td>Sample</td>
<td>Web module</td>
<td>19 03</td>
</tr>
<tr>
<td>DFHGWBI1C</td>
<td>Sample</td>
<td>Web module</td>
<td>D3 -</td>
</tr>
<tr>
<td>DFHGWDRQ</td>
<td>Sample</td>
<td>Web module</td>
<td>19 03</td>
</tr>
<tr>
<td>DFHGWNQE</td>
<td>Sample</td>
<td>Exec ENQ/DEQ Sample XNQREQ Exit</td>
<td>19 03</td>
</tr>
<tr>
<td>DFHWXRDS</td>
<td>Sample</td>
<td>XRF overseer control blocks</td>
<td>19 -</td>
</tr>
<tr>
<td>DFHWXTSE</td>
<td>Sample</td>
<td>XSERQ global user exit program</td>
<td>19 03</td>
</tr>
<tr>
<td>DFHWXZIQ</td>
<td>Sample</td>
<td>Sample XZIQUE global user exit program</td>
<td>19 03</td>
</tr>
<tr>
<td>DFHWZCAT</td>
<td>Sample</td>
<td>Sample XZCAT global user exit program</td>
<td>19 03</td>
</tr>
<tr>
<td>DFHWZCGA</td>
<td>Sample</td>
<td>Global work area for DFHWZCAT</td>
<td>19 -</td>
</tr>
<tr>
<td>DFJGUMOD</td>
<td>Sample</td>
<td>Place holder for DFJGUMOD</td>
<td>19 -</td>
</tr>
<tr>
<td>DLIUIB</td>
<td>DSECT</td>
<td>DL/I user interface block</td>
<td>C2 -</td>
</tr>
<tr>
<td>DLIUIB</td>
<td>DSECT</td>
<td>DL/I user interface block</td>
<td>P2 -</td>
</tr>
<tr>
<td>DLIUIB</td>
<td>DSECT</td>
<td>DL/I user interface block</td>
<td>D3 -</td>
</tr>
<tr>
<td>DLIUIB</td>
<td>Macro</td>
<td>DL/I user interface block</td>
<td>12 -</td>
</tr>
<tr>
<td>DFH99SVC</td>
<td>CSECT</td>
<td>Dyn alloc - SVC services</td>
<td>- 03</td>
</tr>
<tr>
<td>DSNCPRA</td>
<td>Macro</td>
<td>CICS-D02 connect dynamic plan selection</td>
<td>12 -</td>
</tr>
<tr>
<td>DSNCPRA</td>
<td>Macro</td>
<td>CICS-D02 connect dynamic plan selection (Assembler)</td>
<td>C2 -</td>
</tr>
<tr>
<td>DSNCPRA</td>
<td>Macro</td>
<td>CICS-D02 connect dynamic plan selection (COBOL)</td>
<td>P2 -</td>
</tr>
<tr>
<td>DSNCPRA</td>
<td>Macro</td>
<td>CICS-D02 connect dynamic plan selection (PL/1)</td>
<td></td>
</tr>
<tr>
<td>DSNCRCT</td>
<td>Macro</td>
<td>CICS-D02 connect RCT macro</td>
<td>12 -</td>
</tr>
<tr>
<td>DSNCRCT</td>
<td>CSECT</td>
<td>CICS-D02 connect dynamic plan selection</td>
<td>19 03</td>
</tr>
<tr>
<td>MEUKEYS</td>
<td>CSECT</td>
<td>MEU key definitions</td>
<td>17 -</td>
</tr>
<tr>
<td>MEULANG</td>
<td>CSECT</td>
<td>MEU language table</td>
<td>11 -</td>
</tr>
<tr>
<td>MEU00</td>
<td>CSECT</td>
<td>MEU MEU00x message set</td>
<td>13 -</td>
</tr>
<tr>
<td>MEU01</td>
<td>CSECT</td>
<td>MEU MEU01x message set</td>
<td>13 -</td>
</tr>
<tr>
<td>MEU02</td>
<td>CSECT</td>
<td>MEU MEU02x message set</td>
<td>13 -</td>
</tr>
<tr>
<td>MEU03</td>
<td>CSECT</td>
<td>MEU MEU03x message set</td>
<td>13 -</td>
</tr>
<tr>
<td>MEU04</td>
<td>CSECT</td>
<td>MEU MEU04x message set</td>
<td>13 -</td>
</tr>
<tr>
<td>MEU05</td>
<td>CSECT</td>
<td>MEU MEU05x message set</td>
<td>13 -</td>
</tr>
<tr>
<td>SRRC</td>
<td>Symbolic</td>
<td>SAA resource recovery pseudonyms for C</td>
<td>D3 -</td>
</tr>
<tr>
<td>SRRCOBOL</td>
<td>Symbolic</td>
<td>SAA resource recovery pseudonyms for COBOL</td>
<td>C2 -</td>
</tr>
<tr>
<td>SRRHASM</td>
<td>Symbolic</td>
<td>SAA resource recovery pseudonyms for assembler-language</td>
<td>12 -</td>
</tr>
<tr>
<td>SRRPLI</td>
<td>Symbolic</td>
<td>SAA pseudonym file for PL/I</td>
<td>P2 -</td>
</tr>
</tbody>
</table>
Chapter 113. CICS executable modules

The following list shows, for each module:
1. The name of the module
2. Its entry points
3. Callers of the module
4. A brief description of the module
5. Where the module returns to. This information is omitted where the module returns to its caller (the normal situation).

In general, this list is restricted to non-OCO modules. In the few cases where OCO modules are included, no design details are given.

DFHACP

Entry points: DFHACPNA

Called by: DFHAPRM, DFHAPXME

Description: The abnormal condition program writes a message to the terminal and to the CSMT destination if a transaction abends or cannot be started. Subject to tests on the type of terminal, DFHACP invokes DFHMGP to output the message. It calls DFHPEP and, depending on the result, may disable the transaction. For each error, there is an entry in a table which contains the number of the message to be written to the principal facility (terminal) and the number of the message to be written to CSMT. If, in either case, there is no message, zero is entered.

The main subroutines of DFHACP are:

ACPCALMG - Use DFHMGP to output a message
ACPCPWLPEP - Invoke DFHPEP
ACPFMT - Identify message for terminal
TERMERR - Terminal error.

DFHALP

Entry points: DFHALPNA

Called by: DFHAPRM, DFHSII1

Description: The terminal allocation program contains the logic to allocate TCTTE resources to requesting transactions. The request operates in a multiple exchange between the requesting transaction and terminal control. DFHALP passes a SCHEDULE request to terminal control as an ATI terminal control, then responds with an AVAIL command. The requests are represented by AIDs (AID chain manipulations being performed by calls to DFHALP). For LU6.2, DFHALP issues a terminal control allocate mode name macro.

DFHAMP

Entry points: DFHAMPNA

Called by: DFHEIP, DFHSII1

Description: The allocation management program is invoked by the CEDA transaction. It analyzes commands and calls the definition file management program, DFHMP, to process changes to records in the CSD. For the INSTALL command, DFHAMP also calls program manager, transaction manager, and DFHSPP. DFHPUP is called to convert data between address list format and the CSD record format.

DFHAPJC

Entry points: DFHAPJCN

Called by: User

Description: The AP domain journal control gate service module handles WRITE_JOURNAL_DATA calls made by the user exit’s XPI. It gets a TCA if the task doesn’t currently have one, and also a JCA. If the task already has a JCA, this is stacked. It then copies the...
parameter list passed in the domain call, to the JCA, and then issues one of four journal writes, depending on the request. Finally the return code from the JC write is copied into the domain parameter list, and the JCA and TCA are released if they were obtained by DFHAPJC.

### DFHAPSIP

**Entry points:** DFHSIPNA  
**Called by:** DFHAPDM  
**Description:** The main AP domain initialization program provides DFHWTO support and common subroutines used by DFHSIA1 through DFHSIJ1. In sequence, DFHAPSIP performs the following functions:  
- Defines the AP domain subpools  
- Acquires the SIT address  
- Passes control to the DFHSIA1, DFHSIB1, and so on.

The main subroutines of DFHAPSIP are:  
- CHKRLVLR - Check release level  
- OVERLSUP - Overlay supervisor  
- SIGETCOR - Storage allocation  
- SILOADR - Program loader  
- SIPCONS - Console WRITE.

### DFHAPST

**Entry points:** DFHAPST  
**Called by:** DFHEIP, DFHSTST  
**Description:** The supervisory statistics program within the AP domain accepts a request for and then supervises the copying/resetting of statistics counters in the AP domain by calling the appropriate DFHSTxx modules to access the counters.

This module is called when:  
- Statistics domain is collecting INTERVAL statistics and calls this module to pass it copies of and to reset all statistics in AP domain. This module then sequentially calls all of the DFHSTxx modules to do the copying and resetting.  
- A CEMT PERFORM STATISTICS command results in a call to the statistics domain which then makes an appropriate call to this module to pass it copies of the requested statistics. This module then calls the DFHSTxx modules required to do the copying.  
- An EXEC CICS COLLECT STATISTICS command results in a call to this module which then calls the DFHSTxx module required to pass copies of the statistics back to the application program.

Thus, this module is called only by the statistics domain or by DFHEIP.

This module provides two functions:  
- **COLLECT_STATISTICS** collects statistics for all resources in the AP domain and calls the statistics domain to write them out to the SMF data set.

### COLLECT_RESOURCE_STATS

collects statistics for the named resource type (optionally qualified by the resource identifier) and either copies them to a buffer available through the API, or causes them to be written to the SMF data set.

### DFHAPTD

**Entry points:** DFHAPTD  
**Called by:** DFHETD, DFHTDA, DFHTDB, ME domain  
**Description:** DFHAPTD handles DFHTDTDM macro requests; as such, it provides the transient data gate into the AP domain. DFHTDTDM macro requests are routed from DFHAPTD to DFHTDP using the corresponding DFHTD CTYPE requests.

### DFHAPTI

**Entry points:** DFHAPTI  
**Called by:** the timer domain to handle NOTIFY calls for the application domain.  
**Description:** The DFHAPTO module looks at the token passed by the timer domain and resumes either the DFHAPTI or DFHAPTIX module, as appropriate.

### DFHAPTIX

**Entry points:** DFHAPTIX  
**Called by:** runs as a system task attached by the DFHSIII module.  
**Description:** The DFHAPTIX module is part of the interval control mechanism. When it first gets control, it suspends itself to wait for an interval control ICE to expire. Interval control uses the timer domain to handle time intervals. When the timer domain detects the expiry of an interval control related interval, it calls the DFHAPTO module, which in turn resumes the DFHAPTIX module. The DFHAPTIX module then makes an “expiry analysis” call to the DFHCICP module, which processes any expired ICEs. On return, the DFHAPTIX module suspends itself again to wait for the next ICE to expire.
calls the DFHAPTI module, which resumes the DFHAPTIX module, which in turn calls the DFHICP module to do midnight processing.

### DFHASV

**Entry points:** DFHASVNA  
**Called by:** DFHCSCV  
**Description:** DFHASV is one of the modules that run under the CICS type 3 SVC. On entry to DFHASV, register 0 contains one of the following request codes:  
0 - Paging request  
8 - SRB termination  
9 - HPO initialization  
24 - Monitoring services  
64 - Authorize general purpose subtask TCB  
80 - Issue SDUMP  
136 - Bind AP domain.

### DFHBSIB3

**Entry points:** DFHBSIB3  
**Called by:** DFHTB5xx  
**Description:** DFHBSIB3 adds BMS 3270 support to a TCT table entry.

### DFHBSIZ1

**Entry points:** DFHBSIZ1  
**Called by:** DFHTB5xx  
**Description:** DFHBSIZ1 adds SCS support to a TCT table entry.

### DFHBSIZ3

**Entry points:** DFHBSIZ3  
**Called by:** DFHTB5xx  
**Description:** DFHBSIZ3 adds DFHZCP 3270 support to a TCT table entry.

### DFHBSMIR

**Entry points:** DFHBSMIR  
**Called by:** DFHTB5xx  
**Description:** DFHBSMIR builds a TCT table entry for a session.

### DFHBSMPP

**Entry points:** DFHBSMPP  
**Called by:** DFHTB5xx  
**Description:** DFHBSMPP builds a TCT table entry for a pipeline pool entry.

### DFHBSM61

**Entry points:** DFHBSM61  
**Called by:** DFHTB5xx  
**Description:** DFHBSM61 builds sessions for an LU6.2 mode group.

### DFHBSM62

**Entry points:** DFHBSM62  
**Called by:** DFHTB5xx  
**Description:** DFHBSM62 builds the mode entry for an LU6.2 mode group.

### DFHBS5

**Entry points:** DFHBS5  
**Called by:** DFHTB5xx  
**Description:** DFHBS5 adds a new connection (system entry) to a CICS system.

### DFHBSA

**Entry points:** DFHBSA  
**Called by:** DFHTB5xx  
**Description:** DFHBSA initializes DFHKCP support in a new TCT system entry.

### DFHBSS

**Entry points:** DFHBSS  
**Called by:** DFHTB5xx  
**Description:** DFHBSS initializes the statistics counters in a new TCT system entry.

### DFHBSSS

**Entry points:** DFHBSSS  
**Called by:** DFHTB5xx  
**Description:** DFHBSSS builds security support for a new TCT system entry.

### DFHBSSZ

**Entry points:** DFHBSSZ  
**Called by:** DFHTB5xx  
**Description:** DFHBSSZ builds VTAM interface support for a new TCT system entry.
DFHBSSZB
Entry points: DFHBSSZB
Called by: DFHTBSxx
Description: DFHBSSZB adds a new batch interregion connection to a CICS system.

DFHBSSZG
Entry points: DFHBSSZG
Called by: DFHTBSxx
Description: DFHBSSZG adds a new advanced program-to-program communication (APPC) single-session connection to a CICS system.

DFHBSSZI
Entry points: DFHBSSZI
Called by: DFHTBSxx
Description: DFHBSSZI adds an indirect terminal control system table entry to a CICS system.

DFHBSSZL
Entry points: DFHBSSZL
Called by: DFHTBSxx
Description: DFHBSSZL adds a local terminal control system table entry to a CICS system.

DFHBSSZM
Entry points: DFHBSSZM
Called by: DFHTBSxx
Description: DFHBSSZM introduces a new connection (system) to ZCP.

DFHBSSZP
Entry points: DFHBSSZP
Called by: DFHTBSxx
Description: DFHBSSZP builds an advanced program-to-program communication (APPC) parallel-session connection to a CICS system.

DFHBSSZR
Entry points: DFHBSSZR
Called by: DFHTBSxx
Description: DFHBSSZR builds an MRO session entry.

DFHBSSZS
Entry points: DFHBSSZS
Called by: DFHTBSxx
Description: DFHBSSZS builds an advanced program-to-program communication (APPC) session entry.

DFHBSSZ6
Entry points: DFHBSSZ6
Called by: DFHTBSxx
Description: DFHBSSZ6 builds an LU6.1 connection entry.

DFHBST
Entry points: DFHBST
Called by: DFHTBSxx
Description: DFHBST performs TCTTE initialization common to terminals, pipeline pool entries, and sessions for IRC and ISC.

DFHBSTB
Entry points: DFHBSTB
Called by: DFHTBSxx
Description: DFHBSTB adds support for BMS to a new TCT terminal or session entry.

DFHBSTBL
Entry points: DFHBSTBL
Called by: DFHTBSxx
Description: DFHBSTBL adds support for logical device components (LDCs).

DFHBSTB3
Entry points: DFHBSTB3
Called by: DFHTBSxx
Description: DFHBSTB3 adds partition support to a new TCT terminal or session entry.

DFHBSTC
Entry points: DFHBSTC
Called by: DFHTBSxx
Description: DFHBSTC performs those operations that are executed after the installation of a terminal.
DFHBSTD
Entry points: DFHBSTD
Called by: DFHTBSxx
Description: DFHBSTD adds data interchange program (DFHDIP) support for a new TCT table entry.

DFHBSTE
Entry points: DFHBSTE
Called by: DFHTBSxx
Description: DFHBSTE adds EXEC diagnostic facility (EDF) support for a new TCT table entry.

DFHBSTH
Entry points: DFHBSTH
Called by: DFHTBSxx
Description: DFHBSTH initializes EXEC interface fields for a new TCT table entry.

DFHBSTI
Entry points: DFHBSTI
Called by: DFHTBSxx
Description: DFHBSTI adds interval control program (DFHICP) support for a new TCT table entry.

DFHBSTM
Entry points: DFHBSTM
Called by: DFHTBSxx
Description: DFHBSTM adds message generation program (DFHMGP) support for a new TCT table entry.

DFHBSTO
Entry points: DFHBSTO
Called by: DFHTBSxx
Description: DFHBSTO is the spooler builder.

DFHBSTP3
Entry points: DFHBSTP3
Called by: DFHTBSxx
Description: DFHBST adds 3270-copy support for a new TCT table entry.

---

Chapter 113. CICS executable modules
<table>
<thead>
<tr>
<th>Entry Points</th>
<th>Called by</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFHBSTZH</td>
<td>DFHTBSxx</td>
<td>DFHBSTZH adds an interregion (IRC) batch session to a CICS system.</td>
</tr>
<tr>
<td>DFHBSTZL</td>
<td>DFHTBSxx</td>
<td>DFHBSTZL adds logical device code support to a new TCT terminal or session entry.</td>
</tr>
<tr>
<td>DFHBSTZO</td>
<td>DFHTBSxx</td>
<td>DFHBSTZO adds an MVS console to a CICS system.</td>
</tr>
<tr>
<td>DFHBSTZP</td>
<td>DFHTBSxx</td>
<td>DFHBSTZP adds a pipeline pool entry to a CICS system.</td>
</tr>
<tr>
<td>DFHBSTZR</td>
<td>DFHTBSxx</td>
<td>DFHBSTZR adds an interregion (IRC) session to a CICS system.</td>
</tr>
<tr>
<td>DFHBSTZS</td>
<td>DFHTBSxx</td>
<td>DFHBSTZS adds an advanced program-to-program communication (APPC) session to the terminal control program.</td>
</tr>
<tr>
<td>DFHBSTZV</td>
<td>DFHTBSxx</td>
<td>DFHBSTZV adds the parts of a terminal or session TCT table entry that are special to VTAM and IRC.</td>
</tr>
<tr>
<td>DFHBSTZZ</td>
<td>DFHTBSxx</td>
<td>DFHBSTZZ adds a non-APPC session to the TCT. (APPC is advanced program-to-program communication.)</td>
</tr>
<tr>
<td>DFHBSTZ1</td>
<td>DFHTBSxx</td>
<td>DFHBSTZ1 adds support for a remote terminal to a CICS system.</td>
</tr>
<tr>
<td>DFHBSTZ2</td>
<td>DFHTBSxx</td>
<td>DFHBSTZ2 adds support for a remote advanced program-to-program communication (APPC) connection.</td>
</tr>
<tr>
<td>DFHBSTZ3</td>
<td>DFHTBSxx</td>
<td>DFHBSTZ3 adds a 3270 to the TCT.</td>
</tr>
<tr>
<td>DFHBSXGS</td>
<td>DFHBSMIR, DFHZTSP</td>
<td>DFHBSXGS generates a unique session name for an LU6.2 TCTTE.</td>
</tr>
<tr>
<td>DFHBSZZ</td>
<td>DFHTBSxx</td>
<td>DFHBSZZ adds a terminal or session to the TCT.</td>
</tr>
<tr>
<td>DFHBSZZS</td>
<td>DFHTBSxx</td>
<td>DFHBSZZS adds a new session to LU6.2 support.</td>
</tr>
</tbody>
</table>
**DFHBSZZV**

Entry points: DFHBSZZV  
Called by: DFHTBSxx  
Description: DFHBSZZV adds a VTAM terminal or session to the TCT.

---

**DFHCAPB**

Entry points: DFHCAPNA  
Called by: DFHTCRP  
Description: DFHCAPB processes command analysis for VTAM terminal definitions contained in a load module table DFHRDTxx for TCT migration.

---

**DFHCCNV**

Entry points: DFHCCNV  
Called by: DFHCHS, DFHMIRS  
Description: DFHCCNV provides conversion of user data from ASCII to EBCDIC and from EBCDIC to ASCII for function-shipped requests from CICS OS/2 users. It is called from either the LU2 remote server program DFHCHS or the mirror program DFHMIRS, for EXEC CICS requests and replies originating from the identified server or mirror. For any function-shipped request it is invoked twice, once on the inbound side and once on the outbound path. DFHCCNV is passed the EXEC CICS parameter list by its caller. On the request side, this occurs after DFHCHS or DFHMIRS has called transformer 2 but before DFHEIP is invoked. On the response side, this occurs after DFHEIP returns to DFHCHS or DFHMIRS but before transformer 3 is invoked. External reference is made to a pregenerated CICS OS/2 conversion table, DFHCNV.

---

**DFHCMP**

Entry points: DFHCMPNA  
Called by: DFHETR  
Description: The CICS monitoring compatibility module is invoked by the old event monitoring point of EXEC CICS ENTER TRACEID to interlace to the monitoring domain.

---

**DFHCPY**

Entry points: DFHCPYNA  
Called by: DFHPRK  
Description: The 3270 copy program (transaction CSCY) causes data to be copied from screen to printer in a (VTAM) 3270 system. DFHCPY is invoked by DFHPRK (only if the 3270 has the copy feature) and issues a DFHTC TYPE=COPY macro to the printer. DFHCPY then initiates DFHRKB.

---

**DFHCRC**

Entry points: DFHCRCNA  
Called by: MVS  
Description: The interregion abnormal exit module is a CICS module that contains an ESTAE exit to terminate interregion communication in abnormal conditions. DFHCRC issues a CLEAR request to the interregion SVC.

---

**DFHCRNP**

Entry points: DFHCRNNA  
Called by: DFHCRSP, dispatcher  
Description: DFHCRNP, the connection manager (transaction CSNC), controls IRC connections. It establishes and breaks these connections and processes inbound requests to attach tasks (for example, mirror) to communicate with connected systems.

---

**DFHCRQ**

Entry points: DFHCRQNA  
Called by: transaction CRSQ  
Description: The remote schedule page program is invoked periodically to delete requests to attach a transaction on a remotely owned terminal if those requests have been outstanding for more than the ATI purge delay interval.

---

**DFHCRR**

Entry points: DFHCRRNA  
Called by: DFHCRNP  
Description: The interregion session recovery program performs session recovery on behalf of primary or secondary IRC sessions.

---

**DFHCRS**

Entry points: DFHCRSNA  
Called by: transaction CRSR  
Description: The remote scheduler program builds and ships AIDs for automatic transaction initiation when the terminal is in a remote address space. It receives requests to schedule an AID shipped to it from a remote address space.
DFHCRSP
Entry points: DFHCRSNA
Called by: DFHEIP, DFHSJJ1
Description: The interregion communication startup module can be invoked, either at system initialization or by a CEMT request, in order to make the CICS address space available for communication by other address spaces. DFHCRSP issues a logon request to the interregion communication SVC routine and attaches transaction CSNC (DFHCRNP).

DFHCRT
Entry points: DFHCRTNA
Called by: transaction CXRT
Description: DFHCRT is the relay program used when a transaction attempts to allocate a conversation to a remote advanced program-to-program (APPC) terminal.

DFHCSA
Entry points: DFHCSANA
Called by: Not applicable
Description: The DFHCSA module contains the common system area (CSA) and CSA optional features list, the queue control area (QCA) and, for HPO systems, the SRB interface control area.

DFHCSDUP
Entry points: DFHCUCNA
Called by: MVS
Description: The CSD utility program is an offline program that provides services for the CSD. The utility command processor (DFHCUCP) validates commands and invokes the appropriate routine to execute the requested function. DFHCSDUP calls DFHDMP to access the CSD.

DFHCSSC
Entry points: DFHCSSNA
Called by: DFHSJJ1, DFHSNSN, DFHSUSN, DFHTCRP, DFHZCUT
Description: DFHCSSC, the signon time-out program, is invoked as a system task by DFHSJJ1 and DFHTCRP to perform XRF takeover sign-off time-out processing. It is invoked elsewhere as the CSSC transaction for time-out processing of the following:
• Terminals signed on with the TIMEOUT option
• Entries in the internally managed signon table (SNT)
• Entries in the local userid tables (LUITs).

The CSSC transaction is scheduled when task termination determines that a time-out is necessary. When DFHCSSC is executed, it examines all signed-on terminals, all entries in the SNT managed by DFHTMP, and all entries in the LUITs. It signs off or deletes expired entries as appropriate, and then reschedules itself to perform later time-outs if required.

DFHCSVC
Entry points: DFHCSVC
Called by: MVS
Description: This module is a type 3 SVC that passes control to the various required routines, dependent on the parameter passed to it. On a first request for a particular function, it loads the required module and puts its address in the AFCB and then branches to that code. Further calls result in the address in the AFCB being branched to.
Returns to: Type 3 SVC

DFHCUCAB
Entry points: DFHCUCB
Called by: DFHCAPB
Description: The resource definition online command analyzer interprets a VTAM resource definition in command form and produces a parameter list.

DFHCUCB
Entry points: DFHCUCB
Called by: DFHCUCP
Description: The resource definition online command builder receives commands and transforms them to a format for use by the command processors.

DFHCUCCB
Entry points: DFHCUCB
Called by: DFHCAPB
Description: This program extracts a single entry from a loaded RDT table containing VTAM resource definitions for TCT migration.

DFHCUCDB
Entry points: DFHCUCD
Called by: DFHCAPB
Description: The resource definition online command default values program modifies the parameter list produced by DFHCUCAB by inserting the default values.
DFHCWTO
Entry points: DFHCWTNA
Called by: CWTO transaction
Description: The console write-to-operator module is a CICS-supplied transaction that allows an operator to send a message to the console operator. DFHCWTO issues SVC 35 (WTO) to pass the message to the operator’s console.

DFHDBAT
Entry points: AENTRY
Called by: DFHERM, IMS database resource adapter (DRA).
Description: This program provides a mapping between the external architectures of CICS (the resource manager interface (RMI) and of DBCTL (the database resource adapter (DRA)). Both are independently defined and different. DFHDBAT is part of the support for the CICS-DBCTL interface and runs in an application program environment. DFHDBAT is invoked by a DFHRMCA request through the CICS RMI. The RMI supplies DFHDBAT with a parameter list from which DFHDBAT constructs the DRA INIT, DRA TERM, and DRA THREAD parameter lists. It must also transform the DRA parameter list back, after a DL/I call to the format expected by CICS. Thus, DFHDBAT is also referred to as the CICS-DBCTL adapter-transformer.

DFHDBCON
Entry points: DFHDBCON
Called by: DFHDBME
Description: This program issues a CICS-DBCTL interface connection request to the CICS-DBCTL adapter-transformer, DFHDBAT. DFHDBCON is part of the support for the CICS-DBCTL interface and runs in an application program environment.

DFHDBCR
Entry points: DFHDBCR
Called by: DFHDBCT
Description: DFHDBCR is the CICS/DBCTL XRF tracking program. DFHDBCR runs in an alternate CICS system during the tracking phase. DFHDBCR receives messages from the active CICS system regarding the state of the connection to DBCTL, and drives the XXDFB and XXDTO exits and takes appropriate action.

DFHDBCT
Entry points: DFHDBCT
Called by: DFHDBCTX, DFHDBAT
Description: This program processes any elements placed on the CICS-DBCTL control work element (CWE) chain. DFHDBCT is part of the support for the CICS-DBCTL interface and runs in an application program environment. It is invoked when the CICS-DBCTL connection program, DFHDBCON, attempts to connect to DBCTL. The program then issues a wait. The DFHDBCT program is posted whenever an element is placed on the CWE chain.

DFHDBCTX
Entry points: DFHDBCTX
Called by: DFHDBAT
Description: This program notifies the CICS-DBCTL control transaction of changes in the state of the CICS-DBCTL interface. DFHDBCTX is part of the support for the CICS-DBCTL interface. It does not run in a CICS environment and thus does not use any CICS services. This exit is invoked by the DBCTL adapter on behalf of the DBCTL DRA.

DFHDBDI
Entry points: DFHDBDI
Called by: DFHDBCT
Description: This program disables the CICS-DBCTL adapter program and cleans up the storage used by the CICS-DBCTL interface programs. DFHDBDI is part of the support for the CICS-DBCTL interface and runs in an application program environment. DFHDBDI is invoked by the CICS/VS DBCTL control program, DFHDBCT, just before it terminates.

DFHDBDSC
Entry points: DFHDBDSC
Called by: DFHDBCT, DFHDBME
Description: This program issues a CICS-DBCTL interface disconnection request to the CICS-DBCTL adapter-transformer. DFHDBDSC is part of the support for the CICS-DBCTL interface and runs in an application program environment.

DFHDBIQ
Entry points: DFHDBIQ
Called by: CDBI transaction
Description: This program is the CDBI CICS-supplied transaction. Its function is to inquire on the current status of the CICS-DBCTL interface. DFHDBIQ is part
of the support for the CICS-DBCTL interface.

**DFHDBME**

**Entry points:** DFHDBME

**Called by:** CDBC transaction

**Description:** This program is the CDBC CICS-supplied transaction. Its function is to provide a front end for making certain changes to the status of the CICS-DBCTL interface. DFHDBME is part of the support for the CICS-DBCTL interface.

**DFHDBMOX**

**Entry points:** DFHDBMOX

**Called by:** DFHDBAT

**Description:** This program outputs monitoring information supplied by DBCTL to the monitoring domain, using monitoring domain services. The information is supplied by DBCTL when it has processed a PSB schedule request and a thread termination request. This exit forms part of the support for the CICS-DBCTL interface. It runs in a CICS application environment. This exit is invoked by the CICS-DBCTL adapter.

**DFHDBP**

**Entry points:** DFHDBPNA

**Called by:** DFHAPRC

**Description:** This program invokes DWE processors when a UOW backs out.

**DFHDBREX**

**Entry points:** DFHDBREX

**Called by:** DFHDBAT

**Description:** This program is the CICS-DBCTL resume exit. The resume exit is driven whenever the adapter or the DRA requires to resume a task which they have suspended. This exit forms part of the support for the CICS-DBCTL interface. It does not run in a CICS environment and thus cannot use CICS services.

**DFHDBSPX**

**Entry points:** DFHDBSPX

**Called by:** DFHDBAT

**Description:** This program is the CICS-DBCTL suspend exit. The suspend exit is driven whenever the adapter or the DRA requires to suspend a task. DFHDBSPX forms part of the support for the CICS-DBCTL interface. It runs in a CICS application environment.

**DFHDBSSX**

**Entry points:** DFHDBSSX

**Called by:** DFHDBAT

**Description:** DFHDBSSX is the CICS/DBCTL status exit. In the event of a DRA thread failure, DFHDBSSX is called to transfer ownership of PCB storage to CICS. When the task ends, DFHDBSSX is called to release this storage.

**DFHDBSTX**

**Entry points:** DFHDBSTX

**Called by:** DFHDBAT

**Description:** This program is the CICS-DBCTL statistics exit. The exit outputs CICS-DBCTL session termination statistics to the statistics domain. DFHDBSTX forms part of the support for the CICS-DBCTL interface. It runs in a CICS application environment, but it can also be invoked during CICS orderly termination. This exit is invoked by the CICS-DBCTL adapter.

**DFHDBTOX**

**Entry points:** DFHDBTOX

**Called by:** DFHDBAT

**Description:** This program is the CICS-DBCTL token exit. The function of this exit is to provide the CICS-DBCTL adapter with task tokens for tasks that have not been through the DBCTL call processor, DFHDLIDP, or the DBCTL connection program, DFHDBCON, or the DBCTL disconnection program, DFHDBDSC, where task tokens are usually generated. DFHDBTOX forms part of the support for the CICS-DBCTL interface. It runs in a CICS application environment. This exit is invoked by the CICS-DBCTL adapter.

**DFHDBUEX**

**Entry points:** DFHDBUEX

**Called by:** DFHDBCT, DFHDBDSC

**Description:** DFHDBUEX is the user-replaceable CICS-DBCTL exit program. It is invoked whenever CICS successfully connects to DBCTL and whenever CICS disconnects from DBCTL. DFHDBUEX forms part of the support for the CICS-DBCTL interface. It runs in a CICS application environment.

**DFHDCP**

**Entry points:** DFHDCPNA

**Called by:** DFHDC macro, DFHEDC

**Description:** DFHDCP translates DFHDC macro
requests for a transaction dump to DU domain TRANSACTION_DUMP calls.

**DFHDES**

**Entry points:** DFHDESNA

**Called by:** DFHZEV1, DFHZEV2, DFHZOPN

**Description:** DFHDES performs data encryption and bind-time security.

**DFHDIP**

**Entry points:** DFHDIPNA

**Called by:** DFHACP, DFHDI macro, DFHEDL, DFHKCP, DFHMCP, DFHTOM, DFHZEMW, DFHZRSP, DFHZSUP

**Description:** The data interchange program acts as a function manager when transactions want to communicate with batch devices using SNA support. DFHDIP builds and receives FMHs, which control the data set selection and function currently being performed by the batch device.

The main subroutines of DFHDIP are:

- DESTCHEK - Destination change
- DIABORTE - Abort
- D1CONRTE - Continue
- D1ENDRTE - End
- D1INARTE - Transaction attach
- D1INPRTE - Input
- D1NOTRTE - Note
- D1QUERTE - Query.

**DFHDLI**

**Entry points:** DFHDLINA

**Called by:** User application, DFHMIRS, DFHSPP

**Description:** DFHDLI is the DL/I call router program. It decides which call processor is to be used for the request: DBCTL or REMOTE. It then invokes the appropriate processor: DFHDLIDP or DFHDLIRP.

**DFHDLIDP**

**Entry points:** DFHDLIDP

**Called by:** DFHDLI

**Description:** DFHDLIDP is the DBCTL call processor. It services DL/I calls for PSBs that are owned by a DBCTL subsystem, and invokes the DL/I task-related user exit (adapter) to interface with DBCTL.

**DFHDLIRP**

**Entry points:** DFHDLIRP

**Called by:** DFHDLI

**Description:** DFHDLIRP is the remote call processor. It services DL/I calls that are function-shipped to another CICS system.

**DFHDMP**

**Entry points:** DFHDMPNA

**Called by:** DFHAMP, DFHCS Dup

**Description:** The definition file management program handles physical changes to the CSD. The main processes in DFHDMP are:

- BUILDKWA (DM16) - Build key work area
- CONNECT (DM01) - CONNECT
- CREATSET (DM11) - Create SET
- DELETE (DM05) - DELETE
- DISCONN (DM02) - DISCONNECT
- ENDBRO (DM10) - End BROWSE
- ERASESET (DM12) - Delete SET
- GETNEXT (DM09) - Get next record
- LOCK (DM06) - LOCK
- QUERYSET (DM13) - QUERYSET
- READ (DM04) - Read CSD control records
- RELSEKWA (DM17) - Free key work area
- SETBRO (DM08) - Set browse
- UNLOCK (DM06) - UNLOCK
- WRITE (DM03) - WRITE.

**DFHDRPG**

**Entry points:** DFHDRPNA

**Called by:** DFHEIP

**Description:** DFHDRPG is the EXEC interface processor for EXEC DLI commands for database sharing. It receives the parameters of the command and from them builds a list that is appropriate to call DFHDRPE, the program request handler. On return from DFHDRPE, the status code in the PCB is examined. For some codes, an MVS abend is executed; the other codes are passed back to the application program.
**DFHDDBA$, DFHSDB1$**

**Entry points:** DFHDSBNA

**Called by:** DFHPBP

**Description:** The data stream build program produces the final device-dependent data stream for each page of BMS output. It is invoked only for processing data streams that are not in 3270 format. DFHDSB removes blanks from the ends of lines, converts logical new-line characters into the device-dependent equivalents (adding idle characters where necessary), and inserts horizontal and vertical tab characters if supported.

---

**DFHDU530**

**Entry points:** DFHDPUPNA

**Called by:** MVS

**Description:** The dump utility program formats and prints transaction dumps from a CICS transaction dump data set (DFHDMPA or DFHDMPB). The transaction dumps are written to the data set by the dump domain. They contain information about the state of a particular transaction at the time of a transaction abend or user-requested dump.

---

**DFHDXACH**

**Entry points:** DFHDXACH

**Called by:** DFHDBCR, DFHDBCT

**Description:** DFHDXACH is a stub that is also MVS-attached, and which branches to an input address.

---

**DFHDXSTM**

**Entry points:** DFHDXSTM

**Called by:** DFHDBCT, DFHDBCR

**Description:** DFHDXSTM is used to attach, detach, and inquire on MVS subtasks attached by DFHDBCR and DFHDBCT.

---

**DFHDYP**

**Entry points:** DFHDYP

**Called by:** DFHAPRT

**Description:** This is the system-provided (default) dynamic routing program invoked from the CICS relay program (DFHAPRT) when a remote transaction is defined as being dynamic.

---

**DFHEAI**

**Entry points:** DFHEI1

**Called by:** User application

**Description:** This is a stub that is link-edited with an assembler-language application program to provide communication with DFHEIP. The command-language translator turns each EXEC CICS command into a call statement. The external entry point invoked by the call is resolved to an entry point in this stub. The address of the entry point in DFHEIP (DFHEIPCN) is found through a chain of system and CICS control blocks.

---

**DFHEAI0**

**Entry points:** DFHEAI0

**Called by:** User application

**Description:** This is a stub that is link-edited with an assembler-language application program to provide communication with DFHEIPA, part of the EXEC interface layer, for the prolog and epilog calls generated by the command-language translator in the application program. The external entry point invoked by the calls is resolved to an entry point in this stub. The address of the entry point in DFHEIPA (DFHEIPAN) is found using a chain of system and CICS control blocks.

---

**DFHEAP1$**

**Entry points:** PREPROC

**Description:** The assembler-language translator module performs the following functions:

- Runs offline.
- Takes on an input file.
- Produces an output or listing file.
- Gives a return code according to the highest severity of the message produced:
  
  - 0 - no message
  - 4 - warning
  - 8 - error
  - 12 - severe error
  - 16 - translator failure.

- Replaces CICS commands by invocations of the DFHECALL macro, and inserts invocations of DFHEIEN, DFHEIRET, DFHEISTG, and DFHEIEND macros at appropriate places.
- Inserts diagnostics resulting from errors in commands, as comments in the output program that are not listed on the listing file.

---

**DFHEBF**

**Entry points:** DFHEBFNA

**Called by:** DFHEIP

**Description:** DFHEBF is the EXEC interface processor for the field edit built-in function, DEEDIT.

---

**DFHEBU**

**Entry points:** DFHEBUNA

**Called by:** DFHETL, DFHETC

**Description:** The EXEC function management header
(FMH) construction module is called by DFHETC when a SEND or CONVERSE command is being processed, and ATTACH function management headers have to be built and concatenated ahead of user data.

---

**DFHECI**

**Entry points:** DFHEI1

**Called by:** User application

**Description:** This is a link-edit stub similar to DFHEAI, except that it is used for COBOL application programs.

---

**DFHECID**

**Entry points:** DFHEIN01

**Called by:** DFHECIP

**Description:** The command interpreter module analyzes CECI commands, and manages its displays. It uses the EXEC interface to invoke other CICS functions.

---

**DFHECIP**

**Entry points:** DFHEIN00

**Called by:** CECI transaction

**Description:** The command interpreter program performs preliminary validation and initialization for the CECI transaction, and links to DFHECID.

---

**DFHECP1$**

**Entry points:** PREPROC

**Description:** The COBOL translator module performs the following functions:
- Runs offline.
- Takes on an input file.
- Produces an output or listing file.
- Gives a return code according to the highest severity of the message produced:
  - 0 - no message
  - 4 - warning
  - 8 - error
  - 12 - severe error
  - 16 - translator failure.
- Inserts DFHEIBLK and COMMAREA declarations in the LINKAGE section.
- Inserts the EIB definition in the LINKAGE section.
- Inserts the DIB definition (for DL/I HLPI) in the WORKING_STORAGE section.
- In the PROCEDURE DIVISION, the translator inserts a USING clause in the DIVISION statement, and replaces all CICS and DL/I commands by COBOL CALL statements.
- Inserts diagnostics resulting from any errors in commands, as messages in the translator listing file.

---

**DFHEDAD**

**Entry points:** DFHESP01

**Called by:** DFHEDAP

**Description:** The resource definition online (RDO) transactions module analyzes the commands, and manages the displays for CEDA, CEDB, and CEDC. It uses the EXEC interface.

---

**DFHEDAP**

**Entry points:** DFHESP00

**Called by:** CEDA, CEDB, CEDC transaction

**Description:** The resource definition online (RDO) transactions program performs preliminary validation and initialization for CEDA, and links to DFHEDAD.

**Returns to:** DFHEIP

---

**DFHEDC**

**Entry points:** DFHEDCNA

**Called by:** DFHEIP

**Description:** DFHEDC is the EXEC interface processor for dump commands.

---

**DFHEDFBR**

**Entry points:** DFHEDFBR

**Called by:** CEBR transaction, DFHEDFD

**Description:** The temporary-storage browse transaction browses, copies, or deletes entries in a temporary-storage queue. It interprets commands and PF key actions.

---

**DFHEDFD**

**Entry points:** DFHEDFD

**Called by:** DFHEDFP

**Description:** The EDF display program is invoked from DFHEDFP to analyze and display the current status of the user program. DFHEDFD stores control information about a temporary-storage message queue and uses BMS to format the display screen. DFHEDFD interfaces with other CICS control programs using the EXEC interface.

---

**DFHEDFM**

**Description:** The EDF map set contains BMS maps used by DFHEDFD to format the EDF display.
**DFHEDFP**

**Entry points:** DFHEDFNA

**Called by:** transaction CEDF

**Description:** The EDF main program is the control program for EDF. DFHEDFP can be invoked in one of two ways:
1. Directly from the EDF display terminal by entering the CEDF transaction identification
2. By pressing the user-defined PF key.

DFHEDFP is also attached by DFHEDFX as the main program of the EDF task.

---

**DFHEDFR**

**Entry points:** DFHEDFNA

**Called by:** Not applicable

**Description:** The EDF response table contains a description of the exception responses for each EXEC command and the abend codes associated with error responses. DFHEDFR is used by DFHEDFD to interpret the responses obtained from an EXEC command.

---

**DFHEDFX**

**Entry points:** DFHEDFNA

**Called by:** DFHACP, DFHEIP, program manager

**Description:** The EDF task switch program is invoked from DFHACP, DFHEIP, or program manager when a program is running in debug mode. DFHEDFX suspends the user task and attaches the debugging task, passing it information about the user task in the TWA of the debugging task.

---

**DFHEDI**

**Entry points:** DFHEDINA

**Called by:** DFHEIP

**Description:** DFHEDI is the EXEC interface processor for data interchange commands.

---

**DFHEDP**

**Entry points:** DFHEDPNA

**Called by:** DFHERM

**Description:** DFHEDP converts command-level DL/I statements into a call parameter list acceptable to DL/I. In addition, it provides 31-bit application support by moving segment I/O areas above and below the 16MB line as required.

---

**DFHEDP1$**

**Entry points:** PREPROC

**Description:** The C/370 translator module performs the following functions:
- Runs offline.
- Takes on an input file.
- Produces an output or listing file.
- Gives a return code according to the highest severity of the message produced:
  - 0 - no message
  - 4 - warning
  - 8 - error
  - 12 - severe error
  - 16 - translator failure.
- Inserts the EIB definition at the head of the translated output.
- If the DLI translator option is specified, inserts the DIB definition
- Replaces all CICS and DL/I commands in the input program by function calls (dfhexec) in the output program.
- Inserts diagnostics from any errors in commands, as messages on the translator listing file.

---

**DFHEEI**

**Entry points:** DFHEEINA

**Called by:** DFHEIP

**Description:** DFHEEI is the EXEC interface processor for DFHEIP ADDRESS, ASSIGN, PUSH, POP, and HANDLE commands.

---

**DFHEEX**

**Entry points:** DFHEEXNA

**Called by:** DFHETC

**Description:** The EXEC function management header (FMH) extraction module is called by DFHETC when a RECEIVE or CONVERSE command is being processed, and when data has to be extracted from ATTACH function management headers.

---

**DFHEFRM**

**Entry points:** DFHEFRM

**Called by:** DFHDBP, DFHSPP

**Description:** DFHEFRM is the EXEC interface file control syncpoint processor. At syncpoint commit or rollback time, DFHEFRM deletes the FFLE entries that were created by DFHFCEI for the task.
DFHEGL
Entry points: DFHEGLNA
Called by: DFHEIP
Description: DFHEGL is the EXEC interface processor for unmapped LU6.2 commands.

DFHEIIC
Entry points: DFHEICNA
Called by: DFHEIP
Description: DFHEIIC is the EXEC interface processor for interval control commands.

DFHEIDTI
Entry points: DFHEIDTI
Called by: DFHEIP
Description: DFHEIDTI is the EXEC interface processor for ASKTIME and FORMATTIME. DFHEIDTI updates the time and date fields in the EIB and certain time fields in the CSA, and returns the current time, or date, to the application.

DFHEIP
Entry points: DFHEIPNA
Called by: application programs
Description: DFHEIP is the main EXEC interface module. See "Chapter 35. EXEC interface" on page 511 for further information.

DFHEIPA
Entry points: DFHEIPAN
Called by: DFHEAI0
Description: DFHEIPA is part of the EXEC interface layer. It acquires and partially initializes the DFHEISTG dynamic storage when called from the DFHEIENT macro in an assembler-language application program. It frees this storage when called from the DFHEIRET macro.

DFHEIFIC
Entry points: DFHEIFIC
Called by: DFHEIP
Description: DFHEIFIC is the file control EXEC interface module, providing an interface between DFHEIP and file control. It locates the AFCTE, and performs the security check. For a remote file, DFHEIFIC passes the request to a transformer, which then ships the request to the other system. For a local file, DFHEIFIC converts the EXEC argument list to an FCFR parameter list (as defined by the DFHFCFRA DSECT) and calls DFHFCFR, the file control file request handler. After the request completes, DFHEIFIC builds return code information in the EIB.

DFHEISR
Entry points: DFHEISR
Called by: DFHEDI, DFHEGL, DFHEIQMS, DFHEMS, DFHEOP, DFHETC, DFHETL, DFHTDB, DFHXFFC, DFHXFX
Description: DFHEISR obtains buffers and copies data for the calling EXEC interface modules, at the location and in the storage key required by the application.

DFHEJC
Entry points: DFHEJCNA
Called by: DFHEIP
Description: DFHEJC is the EXEC interface processor for journaling commands.

DFHEKC
Entry points: DFHEKCNA
Called by: DFHEIP
Description: DFHEKC is the EXEC interface processor for task control commands.

DFHELII
Entry points: DFHEI1
Called by: User application
Description: This is a link-edit stub similar to DFHEAI, except that it is used for C/370 application programs.

DFHEMS
Entry points: DFHEMSNA
Called by: DFHEIP
Description: DFHEMS is the EXEC interface processor for BMS commands.

DFHEMTA
Entry points: DFHEMT00
Called by: User application
Description: The master terminal programmed interface program is a special version of DFHEMTP that a user application can link to for master terminal services.
**DFHEMTD**

Entry points: DFHEMT01

Called by: DFHEMTA, DFHEMTP, DFHEOTP, DFHESTP

Description: The master terminal module analyzes the commands, and manages displays for CEMT, CEOT, and CEST transactions. It uses the EXEC interface.

**DFHEMTP**

Entry points: DFHEMT00

Called by: CEMT transaction

Description: The master terminal program performs preliminary validation and initialization for the CEMT transaction, and links to DFHEMTD.

**DFHEOTP**

Entry points: DFHEMT00

Called by: CEOT transaction

Description: The master terminal program performs preliminary validation and initialization for the CEOT transaction, and links to DFHEMTD.

**DFHEPC**

Entry points: DFHEPCNA

Called by: DFHEIP

Description: DFHEPC is the EXEC interface processor for program control commands.

**DFHEPI**

Entry points: DFHEI1

Called by: User application

Description: This is a link-edit stub similar to DFHEAI, except that it is used for PL/I application programs.

**DFHEPPS**

Entry points: PREPROC

Description: The PL/I translator module performs the following functions:
- Runs offline.
- Takes on an input file.
- Produces an output or listing file.
- Gives a return code according to the highest severity of the message produced:
  - 0 - no message
  - 4 - warning
  - 8 - error
  - 12 - severe error
  - 16 - translator failure.

- If the input program is a MAIN procedure, inserts DFHEIPTR as the first parameter on the PROCEDURE statement to address the EIB. The translator also inserts declarations of the EIB and certain temporary variables.
- Replaces all CICS and DL/I commands in the input program by CALL statements in the output program.
- Inserts diagnostics from any errors in commands, as messages on the translator listing file.

**DFHEPS**

Entry points: DFHEPSNA

Called by: DFHEIP

Description: DFHEPS is the link between DFHEIP and the JES interface program, DFHPSP.

**DFHERM**

Entry points: DFHERMNA

Called by: DFHEIP

Description: DFHERM is called by DFHEIP on behalf of the other components of CICS to manage the connection between CICS and non-CICS products.

**DFHESC**

Entry points: DFHESCNA

Called by: DFHEIP

Description: DFHESC is the EXEC interface processor for storage control commands.

**DFHEISP**

Entry points: DFHESPNA

Called by: DFHEIP

Description: DFHEISP is the EXEC interface processor for syncpoint commands.

**DFHESTP**

Entry points: DFHEMT00

Called by: CEST transaction

Description: The master terminal program performs preliminary validation and initialization for the CEST transaction, and links to DFHEMTD.

**DFHETC**

Entry points: DFHETCNA

Called by: DFHEIP

Description: DFHETC is the EXEC interface processor for terminal control commands.
DFHETD
Entry points: DFHETDNA
Called by: DFHEIP
Description: DFHETD is the EXEC interface processor for transient data commands. The EXEC requests are routed from DFHETD to DFHTDP using the corresponding DFHTD CTYPE requests.

DFHETL
Entry points: DFHETLNA
Called by: DFHETC
Description: DFHETL is the EXEC interface processor for mapped LU6.2 commands.

DFHETR
Entry points: DFHETRNA
Called by: DFHEIP
Description: DFHETR is the EXEC interface processor for trace commands.

DFHETS
Entry points: DFHETSNA
Called by: DFHEIP
Description: DFHETS is the EXEC interface processor for temporary-storage commands.

DFHEXI
Entry points: DFHEXINA
Called by: DFHZARQ
Description: The exceptional input program is invoked from DFHZCP when unexpected input is received from a VTAM 3270 terminal that has a task attached. DFHEXI checks whether the input is the result of a 3270 print function key being pressed; if so, DFHEXI issues a DFHTC TYPE=PRINT macro, and then unlocks the keyboard; in any case, DFHEXI then passes control back to DFHZCP.

DFHFCBD
Entry points: DFHFCBD
Called by: DFHFCFR
Description: DFHFCBD handles BDAM file control requests except for OPEN and CLOSE.

DFHFCDN
Entry points: DFHFCDN
Called by: DFHAMFC, DFHAMPFI, DFHEIQDN, DFHEIQDS, DFHFCLF, DFHFCMT, DFHFCN, DFHFCRC, DFHFCRO, DFHFCRD, DFHFCRP
Description: DFHFCDN builds data set name blocks at cold start or in response to CEDA requests. It also examines or modifies data set name blocks in response to EXEC CICS INQUIRE or EXEC CICS SET commands.

DFHFCDTST
Entry points: DFHFCDTST
Called by: DFHFCFR
Description: DFHFCDTST processes file control requests to access data table records for READ-ONLY requests against CICS-maintained tables, and for all record requests against user-maintained tables. It calls data table services to retrieve or modify table records, calls DFHFCVS to retrieve data from the VSAM source data set if it is not in the table, and calls DFHFCDTX to function ship requests that cannot be satisfied by sharing.

DFHFCFR
Entry points: DFHFCFR
Called by: DFHAPLI, DFHAPSM, DFHDTLX, DFHDMPCA, DFHEIFC, DFHERM, DFHFCDTST, DFHFCFR, DFHFCPS, DFHFCRC, DFHFCRP, DFHUEH
Description: DFHFCFR is the central module in the file control component. It handles file control requests issued by DFHFCEI (requests from application programs), or by other CICS modules (internal file control requests). DFHFCFR ensures that the file is both opened and enabled, acquires an FRTE as necessary, performs request validity checking, and then routes the request to the appropriate access-method dependent module (DFHFCBD for BDAM, DFHFCVS for non-RLS VSAM and also for update or browse requests against a CICS-maintained data table, DFHFCRS for RLS VSAM, and DFHFCDTST for all other data table requests).
DFHFCFS
Entry points: DFHFCFS
Called by: DFHAMFC, DFHDMPCA, DFHDMRM, DFHDTLX, DFHIEQDS, DFHFCDTS, DFHFCSR, DFHFCLP, DFHFCQU, DFHFCRC, DFHFCRD, DFHFCRU, DFHFCSD, DFHFCU, DFHFCSV
Description: DFHFCFS changes the state of a file. It invokes DFHFCN to open, or close, files.

DFHFCL
Entry points: DFHFCLNA
Called by: DFHFCN
Description: DFHFCL is a file control program that is link-edited into DFHFCFS. DFHFCL builds and deletes VSAM LSR pools. It is called by DFHFCN with a parameter list that specifies the pool number (1 through 8) and the action to be taken (build or delete).

DFHFCM
Entry points: DFHFCMNA
Called by: DFHFCFS
Description: DFHFCM is a file control program that is link-edited into DFHFCFS. When records are added via a VSAM path, DFHFCM is called to open the base associated with the path.

DFHFCMT
Entry points: DFHFCMT
Called by: DFHAFMT, DFHAMFC, DFHAMPFI, DFHDMPCA, DFHEDFX, DFHIEQDS
Description: DFHFCMT builds file control table entries in response to CEDA commands. It also examines or modifies FCT entries in response to EXEC CICS INQUIRE or EXEC CICS SET commands.

DFHFCN
Entry points: DFHFCNNA
Called by: DFHFCFS
Description: DFHFCN is a file control program that is link-edited into DFHFCFS. DFHFCN opens and closes files. If a file has not been allocated, DFHFCN allocates it, and frees it on closure.

DFHFCRL
Entry points: DFHFCRL
Called by: DFHAMFC
Description: DFHFCRL modifies SHRCTL blocks (describing VSAM LSR pools) in response to CEDA requests.

DFHFCRP
Entry points: DFHFCRP
Called by: DFHCIN2
Description: The file control restart program builds the file control environment and initializes file control.

DFHFCSD
Entry points: DFHFCSD
Called by: DFHSTP
Description: DFHFCSD is called during CICS controlled shutdown processing to close all open files managed by CICS file control.

DFHFCST
Entry points: DFHFCST
Called by: DFHSTFC, DFHSTLS
Description: DFHFCST is called to collect or reset file or LSRPOOL statistics on request from DFHSTFC or DFHSTLS.

DFHFCU
Entry points: DFHFCUNA
Called by: CSFU transaction
Description: DFHFCU issues an OPEN for files specified in the file control table (FCT). This program examines the FCT, and calls DFHFCFS to open all specified files.

DFHFCVR
Entry points: DFHFCVR, UPADEXIT
Called by: DFHFCBD, DFHFCFR, DFHFCVR, DFHFCSV, VSAM
Description: DFHFCVR is a file control program that is link-edited into DFHFCFS. It handles requests to VSAM, and also contains the VSAM UPAD exit.

DFHFCSV
Entry points: DFHFCSV
Called by: DFHFCDTS, DFHFCSR
Description: DFHFCSV handles requests for file control services made against VSAM files. These services include:
  • Communication with files defined in the file control table
Logging of changes to these files by DFHFCJL and the log manager.

Syncpoint services.

**DFHFDP**

**Entry points:** DFHFDPNA

**Called by:** DFHFD macro

**Description:** DFHFDP translates DFHFD macro requests for a system dump to DU domain SYSTEM_DUMP calls.

**DFHFEP**

**Entry points:** DFHFEPNA

**Called by:** CSFE transaction

**Description:** The FE terminal test program can be used to send a complete character set to a terminal or to echo input or to turn tracing on or off. This program is an application program and does not exit to any other CICS modules. However it does use CICS facilities.

**DFHGMM**

**Entry points:** DFHGMMNA

**Called by:** DFHKCP

**Description:** The “good morning” program is invoked by the CSGM system transaction to write a “good morning” message to VTAM logical units when a satisfactory OPNDST has occurred (and if the message has been requested in the TCT TYPE=TERMINAL entry).

**DFHHPsvc**

**Entry points:** IGCnnn

**Called by:** DFHZHPSR (via an SVC call)

**Description:** This is a type 6 SVC module used only on MVS. Its sole purpose is to cause MVS to dispatch an SRB. DFHHPsvc provides part of the CICS high performance option (HPO) code, and is invoked only if HPO is in use. In the entry point name, nnn is the number of the SVC.

**Returns to:** MVS

**DFHICP**

**Entry points:** DFHICPNA

**Called by:** DFHEIC, DFHIC macro

**Description:** The interval control program is used for time management and has two main functions:
1. Services DFHIC macros under the control of a requesting task’s TCA
2. Detects the expiration of time-dependent events, as defined in ICEs.

The main subroutines of DFHICP are:

- ICCANCLN - Cancel a time-ordered request
- ICCXPNL - Time expiration analysis
- ICGTIMEN - Current time of day
- ICGTTTDM - Data retrieval
- ICICECRN - Build basic ICE
- ICPCTSN - Task initiation
- ICPOSTN - Signal expiration of a specified time
- ICRESETN - Time of day clock reset support
- ICSCEDN - ICE schedule
- IOWAITN - Delay processing of a task.

**DFHIIPAS, DFHIIP1$**

**Entry points:** DFHIIPNA

**Called by:** DFHMCP

**Description:** The non-3270 input mapping program performs all BMS input mapping functions for all devices except the 3270. On exit from the module, the input data has been mapped into a newly acquired TIOA that is returned to the application program and is then addressable using BMS DSECTs in the application.

The main subsections of DFHIIP are:

- IIMID - GETMAINs TIOA to return to user, and maps page buffer into it using specified map.
- IIREAD - Reads input data, issuing DFHTC or DFHDI requests to get data from the terminal.
- IISCAN - Scans data stream for device-dependent control characters and creates page buffer.

**DFHIRP**

**Entry points:** DFHIRPNA

**Called by:** DFHCRc, DFHCRNP, DFHCRSP,
DFHDRFD, DFHDRPE, DFHDRPF, DFHsrf, DFHstp, DFHZC

**Description:** The interregion communication program is used to pass data from one region to another in the same CEC. The programs being run in the regions are usually CICS programs, but DFHIRP does not assume this.

**DFHIRW10**

**Entry points:** As defined in interest ladder

**Called by:** DFHIRP, DFHXMP

**Description:** The interregion work exit delivers work to the IRC control task (CSNC). DFHIRW10 is called whenever DFHIRP or DFHXMP has work to deliver to a system that logged on with DFHIRW10 as its

9. **Interest ladder** ladder within DFHIRW10 that expresses interest in all types of MRO work.
interregion work exit. This module checks whether the work being delivered to the target system requires that work be enqueued on CSNC; if so, it enqueues the work and posts CSNC. DFHIRW10 is invoked in access register (AR) mode and user key.

**DFHISP**

**Entry points:** DFHISPNA  
**Called by:** DFHDLI, DFHEIP, DFHEIFC

**Description:** The intersystem communication program is invoked when a request to access resource has to be shipped to a remote system (through ISC or MRO). The requests passed to DFHISP are:
- File control
- Interval control
- Temporary storage
- Transient data
- DL/I.

DFHISP controls the acquisition, use, and freeing of a session to the remote system, and invokes DFHXFP or DFHXFX to process requests and replies. Two user exits are provided in DFHISP: XISCONA can be used to control the queuing of requests from DFHISP to allocate intersystems sessions, and XISLCLQ can be used to override the LOCALQ option of the transaction attributes. XISCONA is invoked for any function-shipping requests that cannot be processed immediately. XISLCLQ is provided to support the local queuing of function-shipped START NOCHECK requests when the link to the remote system is out of service. If a START NOCHECK request is queued, DFHISP starts the CMPX transaction when the link is brought in to service.

**DFHJCP**

**Entry points:** DFHJCPNA  
**Called by:** DFHEJC, DFHJC macro

**Description:** The journal control program (DFHJCP) either processes a request to get a JCA control block, or has been called to write to a journal. In the latter case it examines the information in the JCA that is passed with the request and decides whether to call the recovery manager or the log manager based on whether it finds journalname DFHLOG in the JCA or not. There are three separate calls to the DFHLGGL gate of the log manager: one for a write, a put or a wait request. The same is true for the recovery manager calls, which use the DFHRMRE gate. In addition there is a call to this gate for requests which have keypoint record data with them.

When control returns from either of these domains, the domain’s outcome is mapped onto a valid return code which is put into the JCA before control returns back to the calling program.
**DFHKCSP**

**Entry points:** DFHKCSPA, DFHKCSPI, DFHKCSPD, DFHKCSPF, DFHKCSPP

**Description:** The task SRB control program is part of the high performance option (HPO) code available on CICS on MVS. It runs in SRB mode and resides in protected storage.

---

**DFHLIP**

**Entry points:** DFHLINA

**Called by:** DFHEDFX, DFHEIP, DFHPCPS, DFHSIJI, DFHSTP

**Description:** The language interface program acts as a single point of contact between CICS and AD/Cycle Language Environment/370, and also between CICS and the language environments for VS COBOL II and C/370. To invoke a Run-Time Language Interface (RTL) or Extended Run-Time Language Interface (ERTL) function, the requesting module calls DFHLIP by issuing a DFHCEE FUNCTION= macro. DFHLIP performs all the interface work with the language, including the handling of any errors.

The interface functions driven by DFHLIP and the modules that call DFHLIP for those functions are as follows. An asterisk (*) after a function name shows that the function call is handled entirely within DFHLIP itself, and control remains in DFHLIP upon successful completion of the thread initialization function.

Unless otherwise indicated, each function is used for all three environments. Where alternative function names are given, the name applicable to Language Environment/370 is used in the requesting module’s DFHCEE macro call regardless of the language environment.

- **DFHEDFX** - Determine working storage
  - (Language Environment/370)
  - OR Working storage locate
    - (VS COBOL II and C/370)
- **DFHEIP** - Perform GOTO
  - (Language Environment/370 only)
- **DFHPCPS** - Establish ownership type
  - (Language Environment/370)
    - OR Determine program type
      - (VS COBOL II and C/370)
    - Thread initialization
    - Run-unit initialization *
    - Run-unit begin invocation *
    - (Language Environment/370 only)
    - Run-unit end invocation
    - (Language Environment/370 only)
    - Run-unit termination
    - Thread termination
- **DFHSIJI** - Partition initialization
- **DFHSTP** - Partition termination.

---

**DFHLUP**

**Entry points:** DFHLUPNA

**Description:** DFHLUP is the LU6.2 services manager. It initializes and shuts down a network, and resynchronizes flows.

---

**DFHMCPA, DFHMCPES, DFHMCPIS**

**Entry points:** DFHMCPNA

**Called by:** DFHBMS macro, DFHEMS

**Description:** The mapping control program processes DFHBMS macro requests and completes the processing of a logical message when a task terminates without issuing a DFHBMS TYPE=PAGEOUT. DFHMCP’s main function is to analyze DFHBMS requests and to pass control to the appropriate modules. Other functions include the loading of maps and partition sets, and scheduling of output messages transmitted by temporary storage.

The main subsections of DFHMCP are:

- **MPCPCO** - Completes logical message build message control record for temporary storage
- **MCPDNE** - DWE processing, invoked by DFHKCP to complete BMS processing at application termination
- **MCPINPT** - Handles all input requests
- **MCPIN** - TYPE=IN (EXEC CICS RECEIVE MAP)
- **MCPMPL** - Loads map set and locates map
- **MCPEMB** - TYPE=PAGEOUT (EXEC SEND TEXT)
- **MCPPUG** - TYPE=PAGEOUT (EXEC CICS SEND PAGE)
- **MCPURGE** - TYPE=PAGEOUT (EXEC CICS PURGE MESSAGE)
- **MCPROUTE** - TYPE=ROUTE (EXEC CICS ROUTE).

---

**DFHMCS**

**Entry points:** DFHMCSNA

**Called by:** DFHMCP

**Description:** DFHMCS is the BMS fast path module for standard and full-function BMS, and the program for minimum BMS support. It is called by DFHMCP if the request satisfies one of the following conditions:

- It is a noncumulative direct terminal send map or receive map issued by a command-level program.
- It is for a 3270 display or an LU3 printer which does not support outboard formatting. If the terminal supports partitions, it is in the base state.
- The CSPQ transaction has been started.
- The message disposition has not changed.

---

**DFHMGP**

**Entry points:** DFHMGPNA

**Called by:** DFHACP, DFHCRQ, DFHCRT, DFHEOP, DFHEP, DFHRTE, DFHZEMW, DFHZERH, DFHZISI, DFHZTSP, DFHZXRL

**Description:** The message generation program
provides an interface for sending CICS messages to the terminal end user.

**DFHMGT**

**Entry points:** DFHMGTNA

**Called by:** DFHMGP

**Description:** The message prototype control table, or message generation table, consists of a series of copybooks, DFHMGTnn, each of which contains up to 100 messages that are issued by DFHMGP.

**DFHMIRS**

**Entry points:** DFHMIRNA

**Called by:** Task initiation

**Description:** The mirror program is invoked when a request to access a resource is received from a remote ISC system or from a remote MRO system. DFHMIRS may be thought of as returning the answer to the requesting actions of DFHISP. It is DFHMIRS that controls the receipt of requests and transmission of replies.

DFHMIRS processes requests from:
- MRO-connected systems
- LU6.1 connected systems
- LU6.2 sync level 1 connected systems
- LU6.2 sync level 2 connected systems.

The input to DFHMIRS consists of a TCTTE representing the session between CICS and its session partner, and a TIOA containing the function shipping request.

The TIOA is passed to DFHXFP (transformer 2) for conversion of the request from transmission format to the parameter list format required for DFHEIP or DFHDLI. If the data requires conversion (transaction CPMI), an EXEC CICS LINK is used to link to the data conversion program DFHCCNV, passing a COMMAREA that contains the EXEC CICS parameter list for the request where applicable. DFHMIRS then passes the request to DFHEIP or DFHDLI for execution.

On return from DFHEIP or DFHDLI the data conversion program is called to convert the reply (if applicable), and then the transformer program DFHXFP (transformer 3) is called to convert the reply parameter list to transmission format. DFHMIRS then determines the DFC to send with the reply and transmits the reply to the requesting system. If the mirror task has modified protected resources, it continues receiving requests and transmitting replies until a syncpoint request is received from the remote system.

A mirror task on an IRC link suspends itself on completion of a request and it is then available for use by any other MRO function-shipped request. The dispatcher terminates the mirror task if it is not reused within ten seconds.

**DFHML1**

**Entry points:** DFHML1NA

**Called by:** DFHMCP, DFHPBP

**Description:** The SCSPRT logical unit type 1 output mapping routine is called by DFHPBP to build a page of data stream from a chain of map and application data structure copies. The data contains only features that the TTP says are supported by the target terminal. This routine is called when NLEOM is specified for 3270 printers or LU3 printers.

The main subsections of DFHML1 are:
- MLISPACE - Calculate space for chaining and mapping
- MLIPMCA - Format the chains that describe the maps
- MLIPF - Process map fields.

**DFHMROQP**

**Entry points:** DFHMRONA

**Called by:** DFHCRNP, DFHCRSP

**Description:** The MRO work queue enable/disable program is invoked by the DFHMROQM macro for ENABLE and DISABLE requests (other requests are processed by an inline expansion). DFHMROQP is called by DFHCRSP to enable the MRO work queues when starting interregion communication, and by either DHCRSP or DFHCNRP to disable the work queues when stopping interregion communication. MRO work queues are used to deliver work to the IRC control task (CSNC).

**DFHMSP**

**Entry points:** DFHMSPNA

**Called by:** CMSG transaction

**Description:** The message switching program routes a message entered at the terminal to one or more operator-defined terminals or to other operators. DFHMSP can be used in conversational mode to process operands entered from separate input operations. In this case the operands already processed are preserved in temporary storage.

The main sections and subroutines of DFHMSP are:
- MSBMSRT - Check for complete operands
- MSCNVRS - Issue conversational response
- MSCONTIN - Process conversational response
- MSMSG4 - MSG operand
- MSNTRY - Process operands
- MSROUTE - Route operand.
**DFHMXP**

**Entry points:** DFHMXPNA

**Called by:** Automatic transaction initiation

**Description:** The local queuing shipper provides the means of transferring to a remote system a START request that has been temporarily deferred by use of the local queuing option.

**DFHM32A$, DFHM32I$**

**Entry points:** DFHM32NA

**Called by:** DFHMCP, DFHPBP

**Description:** For a BMS output request, the 3270 mapping program generates the appropriate data stream for a 3270 device, and returns control to DFHPBP which invokes the DFHTPP module to send the data to the appropriate destination, which is either to the direct terminal, or to temporary storage, or back to the caller. For a BMS input request, the data stream from a 3270 device is examined and mapped into a user application TIOA format.

The main subsections of DFHM32 are:

- BMFMHTST - Create beginning of 3270 data stream (FMH cursor positioning)
- BMMID - Input mapping
- BMMMS - Merge maps (output mapping)
- M32PF - Process field.

**DFHPBPAS, DFHPBP1$**

**Entry points:** DFHPBPNA

**Called by:** DFHMCP

**Description:** The page and text build program positions maps or text, including header or trailer maps or text, within a page of output. For non-3270 devices, the module creates a page buffer containing the user’s data which is then passed to DFHDSB to produce a device-dependent data stream. When mapping, this includes merging the data supplied by the application with the constant data included in the map. For 3270 devices, copies of the maps and application-supplied data for a page are chained together, to be processed by module DFHM32, to produce a 3270 data stream. The page and text build program creates dummy maps, and chains them in the same way for 3270 text building. For LU1 printers with extended attributes, copies of the maps and application-supplied data for a page are chained together, to be processed by module DFHML1 to produce an SCS data stream. The page and text build program creates dummy maps, and chains them in the same way for text building. After the maps have been processed by DFHDSB, DFHM32, or DFHML1, DFHPBP calls DFHTPP to write them out.

The main subroutines of DFHPBP are:

- PBDOUTPT - Mapping/text build complete, decide whether to call data stream generator and which one (DFHDSB or DFHM32). Return to caller (DFHMCP).
- PBD00005 - Main control logic, request analysis.
- PBD01000 - Map placement logic (3270 and non-3270 mapping).
- PBD01130 - Non-3270 mapping.
- PBD10000 - Pageout routine.
- PBD11000 - Modify field positions within map (used by 3270 and non-3270 mapping).
- PBD20000 - Text processing (3270 and non-3270).
- PBD30000 - 3270 mapping.
- PBFMHLBD - Build FMH if FMHPARM specified (non-3270 text and map processing).

**DFHPD530**

**Entry points:** DFHPD530

**Called by:** MVS IPCS program

**Description:** DFHPD530 uns as an exit from the MVS IPCS program. It formats an MVS system dump (SDUMP) using the IPCS service routines to extract data and print output, including interpreted trace.

**DFHPEP**

**Entry points:** DFHPEPNA

**Called by:** DFHACP

**Description:** The program error program is CICS-supplied and establishes a base register, establishes addressability to the COMMAREA passed from DFHACP using a DFHPC CTYPE=LINK_URM macro call, and returns control to DFHACP. DFHPEP can be modified by the user to perform further recovery operations.

**DFHPHP**

**Entry points:** DFHPHPNA

**Called by:** DFHMCP, DFHTOM

**Description:** The partition handling program has one entry point, and starts with a branch table that passes control to the required routine according to the request.

The main routines of DFHPHP are:

- PHPPSI - Loads a partition set
- PHPPSC - Destroys any existing partitions and creates new partitions
- PHPPIN - Extracts the AID, cursor position, and partition ID
- PHPXE - Activates the appropriate partition if data is received from an unexpected partition.

**DFHPL1O1**

**Description:** The PL/I interface module contains the following routines:
DFHPL1N - Initial entry point for PL/I programs under CICS
DFHPL1I - CICS macro service interface
DFHPL1C - Set the CSA address
IBMBOCLA/B/C - Startup routines for open/close functions

DFHPRK
Entry points: DFHPRKNA
Called by: DFHZATT
Description: The 3270 print key program (transaction CSPK) is invoked when, under VTAM, the 3270 program access key designated as the print key is pressed and no task is attached to the terminal. If the 3270 hardware copy feature is present, DFHPRK attaches task CSCY to the printer designated in the TCTTE, and DFHCPY is executed. If the copy feature is not present, DFHPRK executes a DFHTC TYPE=PRINT macro.

DFHPSP
Entry points: DFHPSPNA
Called by: DFHEPS
Description: DFHPSP is the system spooling interface control module.

DFHPSPDW
Entry points: DFHPSPDW
Called by: DFHSPP
Description: DFHPSPDW is the system spooling interface DWE.

DFHPSPSS
Entry points: DFHPSPSS
Called by: DFHPSP
Description: The system spooling JES interface subtask module attaches a subtask to check whether a writer name and a token have been supplied. It opens and closes JES data sets, reads a record, and writes a record.

DFHPSPST
Entry points: DFHPSPST
Called by: DFHPSPSS
Description: DFHPSPST is the system spooling JES interface control module.

DFHPSSVC
Entry points: DFHPSSNA
Called by: DFHPSPSS, DFHPSPST
Description: DFHPSSVC is the system spooling interface module that retrieves a data set name for a given external writer name, dynamically allocates it, and returns its DDNAME.

DFHPUP
Entry points: DFHPUPNA
Called by: DFHAMP, DFHCSDUP
Description: The parameter utility program transforms the definition data of the CSD. In the CSD, the data is held in a compacted form and each field is self-identifying. Elsewhere in the processing, these fields are handled in parameterized form, using an argument address list. It also serves to transform the resource definition to the original high-level command.

DFHP3270
Entry points: DFHP32NA
Called by: CSPP transaction, DFHTCP, DFHZCP
Description: The 3270 print program prints 3270 data received from a screen on a 3270 printer. The data is compressed where possible and then transmitted to the printer.

DFHQRY
Entry points: DFHQRY
Called by: DFHALP, DFHTCTI, DFHZATT
Description: The query transaction (DFHQRY) sends a READ PARTITION QUERY structured field to a 3270, analyzes the response, and completes information in the corresponding TCTTE. DFHQRY can be attached by DFHALP, DFHTCTI, or DFHZATT.

DFHRCEX
Entry points: DFHRCEX
Called by: DFHFCBP, DFHTCBP, DFHUSBP
Description: DFHRCEX enables the global user exits for emergency restart processing.

DFHRKB
Entry points: DFHRKBA
Called by: DFHCOPY
Description: The release 3270 keyboard program is initiated by DFHCOPY to release a 3270 keyboard. It does this by issuing a DFHTC TYPE=WRITE macro.
that sends a 3270 write control character.

DFHREST

Entry points: DFHREST

Called by: DFHXMTA

Description: The transaction restart program, DFHREST, is a user-replaceable module that helps you determine whether or not a transaction is restarted. The default DFHREST module requests a transaction restart under certain conditions; for example, for a program isolation deadlock, one of the tasks is backed out and automatically restarted, and the other is allowed to complete its update.

DFHRLRA$, DFHRLR1$

Entry points: DFHLRRLNA

Called by: DFHMCP

Description: The route list resolution program builds a terminal type parameter (TTP) control block for each type of terminal for which a message is to be built. A TTP is acquired for each terminal type in the user route list and the direct terminal if there is one.

The main subsections of DFHRLR are:

RLRALL - Routing with ROUTE=ALL specified in application
RLRLIST - Routing with route list specified in application
RLROPCL - Routing with OPCODE= specified in application
RLRTEBY - Nonrouting, non-LDC device (that is direct terminal)
RLR3601 - Nonrouting LDC device.

DFHRMSY

Entry points: DFHRMSNA

Called by: DFHERMSP, DFHERMRS

Description: The purpose of task-related user exit resynchronization is to resolve any in-doubt LUWs. Task-related user exit resynchronization is called by DFHERMRS during execution of the RESYNC command to restore the CICS end of the thread that was interrupted by the failure of the connection with the resource manager.

It is also called by DFHERMSP when a wait is unshunted and requires RMI resynchronization with a resource manager.

DFHRTC

Entry points: DFHRTCNA

Called by: CSSF transaction

Description: The CSSF transaction is invoked on the remote system when a CRTE routing session is to be canceled. CSSF runs the CRTE cancel command processor, DFHRTC, to sign off the user and terminate the extended routing session. DFHRTC calls DFHSUSN to sign off the surrogate.

DFHRTE

Entry points: DFHRTENA

Called by: transaction CRTE, DFHSNTU

Description: The transaction routing program establishes a transaction routing session with a remote region specified by the user. Subsequent input is analyzed by DFHRTE, the transaction code extracted, and a request issued to DFHZTSP to route the transaction to the required system.

DFHSFP

Entry points: DFHSFP

Called by: CESF trans.

Description: The sign-off program signs off the user who invoked the CESF transaction.

DFHSIA1

Entry points: DFHSIANA

Called by: DFHAPSIP

Description: The DFHSIA1 system initialization program loads and initializes the CSA.

DFHSIB1

Entry points: DFHSIBNA

Called by: DFHAPSIP

Description: The DFHSIB1 system initialization program loads the CICS nucleus.

DFHSIC1

Entry points: DFHSICNA

Called by: DFHAPSIP

Description: The DFHSIC1 system initialization program initializes the transaction manager and the storage manager domain's macro compatibility interface, acquires a TCA for LIFO functions during initialization, initializes user exits, and processes the START parameter.

DFHSID1

Entry points: DFHSIDNA

Called by: DFHAPSIP

Description: The DFHSID1 system initialization program performs the following functions:
• Adds storage subpools for transient data use
• Allocates storage for transient data control blocks:
  – TDST
  – MBCA, MBCBs, and MQCBs, I/O buffers if required
  – MRCA, ACBs, MRCBs, and RPLs
• Creates the DCTE and SDSCI for CXRF.

DFHSIF1
Entry points: DFHSIFNA
Called by: DFHAPSIP
Description: The DFHSIF1 system initialization program initializes terminal control. DFHSIF1:
• Opens the VTAM ACB
• Builds hash-table entries for non-VTAM terminals
• Constructs a DFHZCP module list in the TCT prefix
• Initializes the attach tables.

DFHSIG1
Entry points: DFHSIGNA
Called by: DFHAPSIP
Description: The DFHSIG1 system initialization program opens the dump data set.

DFHSIH1
Entry points: DFHSIHNA
Called by: DFHAPSIP
Description: The DFHSIH1 system initialization program:
• Loads the DBCTL call processor (DFHDLIDP)
• Loads the remote DBCTL call processor (DFHDLIRP) if necessary
• Attaches the TCP task.

DFHSII1
Entry points: DFHSIIANA
Called by: DFHAPSIP
Description: The DFHSII1 system initialization program establishes AP domain recovery routines in DFHSRP and calls DFHICRC to initialise Interval Control services. It attaches the CPLT transaction to run the first stage PLTPI programs, the CSTP transaction (the TCP task) and a system transaction to run the rest of AP initialization (the III task). The rest of DFHSII1, running as the III task:
• Starts XRF control transactions if required
• Attaches the CICS restart tasks to run in parallel:
  – Security interface
  – Transient data
  – Terminal control
  – Program control
  – Task control
  – File control
  – Common programming interface (CPI)
  – Partner resource manager
  – Object recovery
  – Autoinstall terminal model manager
• Waits for the restart tasks to complete
• Processes the GRPLIST parameter

DFHSIJ1
Entry points: DFHSIJNA
Called by: DFHAPSIP
Description: DFHSIJ1 is the last to be executed in the process of system initialization. It issues the message ‘CONTROL IS BEING GIVEN TO CICS’ and passes control back to DFHAPSIP. DFHSIJ1:
• Links to DFHCRSP, if IRCSTRT=YES is specified as a system initialization parameter, to start up the interregion communication session
• Links to DFHPSIP to enable the system spooling interface
• Enables the DL/I high-level programming interface by acquiring an exit program block and addressing DFHEDP
• Enables AUTOINSTALL
• Links to the second-stage PLT programs listed in DFHPLT, then deletes this table
• Issues a DFHLDLDM SET_OPTIONS call to instruct the loader domain to write all outstanding program definitions to the catalogs.

DFHSIP
Entry points: DFHKESIP
Called by: MVS
Description: DFHSIP initializes CICS and also contains code for the following domains:
• Kernel (KE)
• Domain manager (DM)
• Dispatcher (DS)
• Dump (DU)
• Global catalog (GC)
• Local catalog (LC)
• Loader (LD)
• Lock manager (LM)
• Message (ME)
• Parameter manager (PA)
• Storage manager (SM)
• Trace (TR).

DFHSKP
Entry points: DFHSKMNA, DFHSKC, DFHSKE
Called by: MVS, DFHFCL, DFHFCM, DFHFCN, DFHPSSS, DFHSTP, DFHXSMM
Description: DFHSKP consists of these modules, which are link-edited together:
DFHSKM - subtask manager
DFHSKC - subtask control program
DFHSKE - subtask execution program.

DFHSKM calls and, if necessary, attaches DFHSKC to process the created work queue element (WQE). DFHSKM also causes termination of the subtask when requested, and handles DWE processing and task cancel requests. DFHSKC starts an operating system subtask, DFHSKE, and waits for its completion. DFHSKE processes WQEs, looking at in-progress and waiting queues on a first-in, first-out basis. DFHSKE intercepts program checks and operating system abends.

DFHSMSSCP
Entry points: DFHSMSSCP
Called by: DFHSC macro
Description: The storage control program is called as a result of DFHSC GETMAIN and FREEMAIN macro requests issued from CICS modules.

DFHSNAT
Entry points: DFHSNAT
Called by: DFHCRNP, DFHZISP, DFHZSUP (via DFHSUSN)
Description: The attach-time signon/sign off interface program provides support for the signon and sign off of LU6.2 sessions.

DFHSNNFY
Entry points: DFHSNNFY
Called by: IRRDPR10
Description: The CICS segment notify exit is called by RACF whenever a change is made to a user’s CICS segment in the RACF database.

DFHSNMIIG
Entry points: DFHSNMIIG
Called by: MVS
Description: The signon table migration utility program produces a CLIST file containing ADDUSER and ALTUSER commands that provide RACF with all the user attributes for each user entry in the signon table (SNT). This CLIST file is run by a TSO user to migrate the user information to RACF.

DFHSNP
Entry points: DFHSNP
Called by: CESN transaction
Description: The signon program is called in response to a CESN signon request. DFHSNP interprets the signon parameters, prompts the operator for more parameters if needed, and passes the values to the security manager for verification.

DFHSNSN
Entry points: DFHSNSN
Called by: DFHCSSC, DFHSNAT (via DFHSUSN)
Description: The optimized signon/sign off interface program provides a mechanism for optimizing calls to the security manager. It achieves this optimization using the signon table (SNT).

DFHSNVCL
Entry points: DFHSNVCL
Called by: IRRDPR02
Description: The OPCLASS validation exit is called by RACF to validate the operands of the OPCLASS subparameter of the CICS parameter in the ADDUSER or ALTUSER TSO commands. DFHSNVCL checks whether the operands are in the range 1 through 24.

DFHSNVID
Entry points: DFHSNVID
Called by: IRRDPR02
Description: The OPIDENT validation exit is called by RACF to validate the operand of the OPIDENT subparameter of the CICS parameter in the ADDUSER or ALTUSER TSO commands.

DFHSNVPR
Entry points: DFHSNVPR
Called by: IRRDPR02
Description: The OPPRTY validation exit is called by RACF to validate the operand of the OPPRTY subparameter of the CICS parameter in the ADDUSER or ALTUSER TSO commands. DFHSNVPR checks whether the operand is in the range 0 through 255.

DFHSNVTO
Entry points: DFHSNVTO
Called by: IRRDPR02
Description: The TIMEOUT validation exit is called by RACF to validate the operand of the TIMEOUT subparameter of the CICS parameter in the ADDUSER or ALTUSER TSO commands.
subparameter of the CICS parameter in the ADDUSER or ALTUSER TSO commands. DFHSNVTO checks whether the operand is in the range 1 through 60.

---

**DFHSPP**

**Entry points:** DFHSPPNA

**Called by:** DFHESP, DFHSP macro

**Description:** The syncpoint program is invoked during a user-specified syncpoint (by a DFHSP macro) or at task termination. For a rollback request only, DFHSPP calls DFHDBP to restore recoverable resources. It scans the DWE chain invoking the appropriate DWE processors, and performs the necessary syncpoint logging. It dequeues all resources enqueued by the transaction. DFHSPP processes any DWEs connected with the resource manager, and processes the RESYNC command.

The main subroutines of DFHSPP are:

- SPP00005 - Write DWE log data
- SPP02020 - Build a DWE chain that can be logged
- SPP03000 - End.

---

**DFHSRLE**

**Entry points:** DFHSSRLI

**Called by:** DFHSRP

**Description:** DFHSRLE is called during recovery processing after a system abend has occurred, to build the SRP_ERROR_DATA block and pass control to the XSRAB global user exit.

---

**DFHSRP**

**Entry points:** DFHSSRPNA

**Called by:** AP domain recovery routines

**Description:** The system recovery program deals with program check interrupts, system abends, and runaway tasks in the AP domain. For a program check, DFHSRP abends the task with abend code ASRA. For a system abend, DFHSRP searches the SRT for the abend code that has arisen and, if a match is found, calls DFHSRLE to invoke the XSRAB global user exit (if active). Afterwards, DFHSRP can either abend CICS or attempt to keep it running with only the faulty task abended (ASRB). For a runaway task, DFHSRP abends the task with abend code AICA.

---

**DFHSSEN**

**Entry points:** DFHSSSEN

**Called by:** MVS subsystem interface

**Description:** The subsystem end-of-memory routine is invoked by the MVS subsystem interface at all end-of-task (EOT) and end-of-memory (EOM) events when the CICS subsystem has been initialized by module DFHSSIN. It cleans up any subsystem control blocks owned by the terminating CICS region.

---

**DFHSSENC**

**Entry points:** DFHSSGC

**Called by:** DFHCSVC, DFHSSSEN (through the subsystem interface)

**Description:** The subsystem generic connect routine records the existence of active CICS address spaces. When the first CICS address space becomes active in an MVS image, DFHSSENC enables the subsystem broadcast facility of MVS console management. When the last CICS address space becomes inactive in an MVS image, it disables the broadcast facility.

---

**DFHSSWRT**

**Entry points:** DFHSSWTA

**Called by:** MVS console support

**Description:** The subsystem interface WTO router is invoked for all MVS console messages when the console message broadcast facility has been enabled by DFHSSGC. DFHSSWRT routes DFH messages to 1552 CICS Transaction Server: CICS Diagnosis Reference
DFHSSWTO, and routes MODIFY command text to DFHSSWTF.

**DFHSSWTO**

Entry points: DFHSSWTO
Called by: DFHSSWT
Description: This module inserts the CICS region’s applid into all DFH messages issued under a CICS TCB whose applid can be determined.

**DFHSSWTF**

Entry points: DFHSSWTF
Called by: DFHSSWT
Description: This module suppresses signon passwords that are supplied on CESN transactions entered through MODIFY commands on an MVS console. Any passwords are replaced by eight asterisks.

**DFHSTDT**

Entry points: DFHSTDT
Called by: DFHAPST
Description: This module is called by DFHAPST to collect or reset dynamic transaction backout statistics. Statistics are written to the SMF data set or made available on the API according to the type of request.

**DFHSTFC**

Entry points: DFHSTFC
Called by: DFHAPST
Description: This module is called by DFHAPST to collect or reset file control statistics. Statistics are written to the SMF data set or made available on the API according to the type of request.

**DFHSTIB**

Entry points: DFHSTIB
Called by: DFHAPST
Description: This module and called by DFHAPST to collect or reset IRC batch system connected statistics. Statistics are written to the SMF data set or made available on the API according to the type of request.

**DFHSTJC**

Entry points: DFHSTJC
Called by: DFHAPST
Description: This module is called by DFHAPST to collect or reset journal control statistics. Statistics are written to the SMF data set or made available on the API according to the type of request.

**DFHSTLK**

Entry points: DFHSTLK
Called by: DFHAPST
Description: This module is called by DFHAPST to collect or reset ISC/IRC statistics. Statistics are written to the SMF data set or made available on the API according to the type of request.

**DFHSTLS**

Entry points: DFHSTLS
Called by: DFHAPST
Description: This module is called by DFHAPST to collect or reset LSRPOOL statistics. Statistics are written to the SMF data set or made available on the API according to the type of request.

**DFHSTP**

Entry points: DFHSTPNA
Called by: DFHEMTP
Description: The main function of the system termination program is to shut down CICS. In sequence, DFHSTP performs the following functions (according to options specified):

1. Collects statistics now if immediate shutdown
2. Shuts down the resource managers
3. Terminates subsystem interface
4. Resumes suspended tasks
5. Executes the programs defined in the first part of DFHPLT
6. Rebuilds AIDs for paging sessions
7. Breaks the ICE and AID chains
8. Quiences IRC
9. Executes the programs defined in the second part of DFHPLT
10. Closes all open files managed by CICS file control
11. Synchronize with Recovery Manager shutdown keypoint
12. Call WKP to catalog terminals and profiles
13. Terminate extra partition TD
14. Signs off from the CAVM
15. Terminates general-purpose subtasking facility
16. Calls the kernel to terminate the system.

Returns to: MVS

**DFHSTSZ**

Entry points: DFHSTSZ
Called by: DFHAPST
Description: DFHSTSZ is called by DFHAPST to collect or reset FEPI statistics. Statistics are written to the SMF data set or made available on the API.
DFHSTTD

Entry points: DFHSTTD

Called by: DFHAPST

Description: DFHSTTD is called by DFHAPST to collect or reset transient data statistics. Statistics are written to the SMF data set or made available on the API according to the type of request.

DFHSTTM

Entry points: DFHSTTM

Called by: DFHAPST

Description: DFHSTTM is called by DFHAPST to collect or reset table manager statistics. Statistics are written to the SMF data set or made available on the API according to the type of request.

DFHSTTR

Entry points: DFHSTTR

Called by: DFHAPST

Description: DFHSTTR is called by DFHAPST to collect or reset terminal statistics. Statistics are written to the SMF data set or made available on the API according to the type of request.

DFHSTTS

Entry points: DFHSTTS

Called by: DFHAPST

Description: DFHSTTS is called by DFHAPST to collect or reset temporary-storage statistics. Statistics are written to the SMF data set or made available on the API according to the type of request.

DFHSUSN

Entry points: DFHSUSN

Called by: DFHCAP, DFHBSTS, DFHCRNF, DFHCSSC, DFHEEI, DFHEIQST, DFHERM, DFHESN, DFHMGPME, DFHMGPR0, DFHRCT, DFHSUSX, DFHTCTI, DFHTFQ, DFHTPR, DFHXSMN, DFHZCUT, DFHZEV1, DFHZEV2, DFHZISP, DFHZIS2, DFHZNAC, DFHZOPN, DFHZSUP, DFHZTSP, DFHZXCU

Description: DFHSUSN is used to create, destroy, and query the contents of a signon table element (SNTTE). It calls DFHSUSX to notify the XRF alternate system of the creation of destruction of SNTTEs. It also provides an interface for the creation and validation of encrypted passwords used in LU6.2 bind password processing.
**DFHTAJP**

**Entry points:** DFHTAJNA

**Description:** The time adjustment program calls DFHICP to reset the CSA's time fields according to the host-supplied time-of-day. DFHTAJP then scans the ICE chain and adjusts the expiry time of interval-controlled ICEs. Time-controlled ICEs are not adjusted but the ICE chain is reordered so that it is left in order by expiry time. Times held in the TCT and CSATCNNDT are decreased, and negative times are made zero. Lastly, DFHTAJP writes a message.

**DFHTBSB**

**Entry points:** DFHTBSB

**Called by:** DFHZCQIS

**Description:** DFHTBSB adds a node to the control-block structure. It is called during the dynamic installation of TCT resources, and calls routines in the control block builder.

**DFHTBSBP**

**Entry points:** DFHTBSBP

**Called by:** DFHTBSB, DFHTBSBP

**Description:** DFHTBSBP is the recursive part of DFHTBSB.

**DFHTBSD**

**Entry points:** DFHTBSD

**Called by:** DFHZCQDL

**Description:** DFHTBSD deletes a node in a CICS terminal network.

**DFHTBSDP**

**Entry points:** DFHTBSDP

**Called by:** DFHTBSD, DFHTBSDP

**Description:** DFHTBSDP is the recursive part of DFHTBSD.

**DFHTBSL**

**Entry points:** DFHTBSL

**Called by:** DFHTBSR, DFHZCQCH

**Description:** DFHTBSL creates the recovery record for a node during the dynamic installation of a TCT table entry using the CEDA INSTALL command, for example, and calls routines in the control-block builder.

**DFHTBSLP**

**Entry points:** DFHTBSLP

**Called by:** DFHTBSL, DFHTBSLP, DFHTBSSP

**Description:** DFHTBSLP is the recursive part of DFHTBSL.

**DFHTBSQ**

**Entry points:** DFHTBSQ

**Called by:** DFHZCQIQ

**Description:** DFHTBSQ is called to retrieve the parameters that were supplied to a TCT table entry at build time.

**DFHTBSQP**

**Entry points:** DFHTBSQP

**Called by:** DFHTBSQ

**Description:** DFHTBSQP is called by DFHTBSQ to retrieve parameters that were supplied to a TCT table entry at build time.

**DFHTBSR**

**Entry points:** DFHTBSR

**Called by:** DFHZCQRS

**Description:** DFHTBSR takes a table-builder recovery record and re-creates the corresponding table entry. It is called during warm or emergency restart.

**DFHTBSRP**

**Entry points:** DFHTBSRP

**Called by:** DFHTBSR

**Description:** DFHTBSRP is called by DFHTBSR.

**DFHTBSSP**

**Entry points:** DFHTBSSP

**Description:** DFHTBSSP performs a commit or rollback action for a previous table-builder change according to the outcome of a logical unit of work. Each action is dequeued from a DWE.

**DFHTBS00**

**Entry points:** DFHTBS

**Description:** DFHTBS00 is the main routine for DFHTBS and holds the addresses of the modules used to build control blocks for the dynamic installation of TCT resources.
**DFHTCBP**

**Entry points:** DFHTCBNA

**Description:** The terminal control backout program restores TCTTEs and other ISC state data during emergency restart.

**DFHTCP**

**Entry points:** DFHTCPNA

**Description:** DFHTCP is the terminal control program. The terminal control task is attached during system initialization and remains until termination. DFHTCP manages all non-VTAM terminals, which involves:

- Ensuring that I/O operations are started when possible on the lines
- Analyzing completion information
- Attaching transactions when data is received from a terminal and no task is attached to that terminal
- Servicing terminal control requests from user transactions.

The modules and subsections of DFHTCP are:

- **DFHTCAM** - Terminal control TCAM device dependent
- **DFHTCCLC** - Terminal control line control scan routine
- **DFHTCCOM** - Terminal control common logic
- **DFHTCCSS** - Terminal control start-stop common logic
- **DFHTCDEF** - Terminal control symbol definition
- **DFHTCORS** - Terminal control storage handling
- **DFHTCSAM** - Terminal control sequential terminal device dependent
- **DFHTCTI** - Terminal control task initiation
- **DFHTCTRN** - Terminal control translate tables.

**DFHTCRP**

**Entry points:** DFHTCRP

**Description:** DFHTCRP initializes and recovers terminal control definitions and protected messages. It is run as a task during CICS initialization.

**DFHTCRPC**

**Called by:** DFHZXQO

**Description:** DFHTCRPC is the XRF tracking interface for ZCP sessions. It is one of a set of routines called by DFHZXQO from the same CALL statement, the entry point address having been passed to DFHZXQO. This routine calls DFHZXST (through DFHZXS) to make changes to the session state.

**DFHTCRPU**

**Called by:** DFHZXQO

**Description:** DFHTCRPU is the XRF tracking interface for signon table elements (SNTTEs). It is one of a set of routines called by DFHZXQO from the same CALL statement, the entry point address having been passed to DFHZXQO. This routine calls DFHSUSX to add or delete tracked SNTTEs, and to make changes to the signed-on state.

**DFHTDA**

**Entry points:** DFHTDANA

**Called by:** DFHAKP, DFHAMCSD, DFHAPTD, DFHCNRP, DFHCRQ, DFHDBP, DFHEIQMS, DFHEIQSQ, DFHESE, DFHETD, DFHJCP, DFHMCP, DFHMGP00, DFHRCRP, DFHRUP, DFHSIII, DFHESTP, DFHSTD, DFHTCAP, DFHTDRP, DFHTEPM, DFHTPQ, DFHTRP, DFHTSRP, DFHWKP, DFHZNAC

**Description:** DFHTDA, which is link-edited with RMODE(24), handles DFHTD macro requests. In particular:

- DFHTD TYPE=GET|PUT|PURGE requests are converted to the corresponding DFHTD CTYPE=GET|PUT|PURGE requests.
- DFHTD CTYPE=GET|PUT|PURGE requests for intrapartition queues are routed to DFHTDQ for further processing.
- All of the processing for DFHTD CTYPE=GET|PUT for extrapartition queues is done under the QR TCB.
- Much of the processing for DFHTD CTYPE=OPEN|CLOSE for extrapartition queues is done under the RO TCB.

**DFHTDA**

**Entry points:** DFHTDANA

**Called by:** DFHAKP, DFHAMCSD, DFHAPTD, DFHCNRP, DFHCRQ, DFHDBP, DFHEIQMS, DFHEIQSQ, DFHESE, DFHETD, DFHJCP, DFHMCP, DFHMGP00, DFHRCRP, DFHRUP, DFHSIII, DFHESTP, DFHSTD, DFHTCAP, DFHTDRP, DFHTEPM, DFHTPQ, DFHTRP, DFHTSRP, DFHWKP, DFHZNAC

**Description:** DFHTDA, which is link-edited with RMODE(24), handles DFHTD macro requests. In particular:

- DFHTD TYPE=GET|PUT|PURGE requests are converted to the corresponding DFHTD CTYPE=GET|PUT|PURGE requests.
- DFHTD CTYPE=GET|PUT|PURGE requests for intrapartition queues are routed to DFHTDQ for further processing.
- All of the processing for DFHTD CTYPE=GET|PUT for extrapartition queues is done under the QR TCB.
- Much of the processing for DFHTD CTYPE=OPEN|CLOSE for extrapartition queues is done under the RO TCB.

CICS® Transaction Server for z/OS®, Version 2 Release 1 uses QSAM GL|PL mode processing, unlike previous CICS releases which used QSAM GL|PM mode processing.
DFHTDB
Entry points: DFHTDBNA
Called by: DFHTDA
Description: DFHTDB, which is link-edited with RMODE(ANY), handles DFHTD macro requests for intrapartition queues. In particular, DFHTDB:
- Manages the input and output cursors for each queue
- Manages space on the intrapartition data set
- Initiates transactions when trigger levels are reached
- Manages the buffers; processing is done under the QR TCB
- Manages the strings; processing is done under the CO TCB.

DFHTDEXL
Entry points: EX11RTNE
Called by: QSAM
Description: DFHTDEXL contains the DCB abend exit routine used for extrapartition processing.

DFHTDP
Entry points: DFHTDANA
Called by: DFHAKP, DFHAMCSD, DFHAPT, DFHCRNP, DFHCRC, DFHDBP, DFHIEQMS, DFHIEQSO, DFHSE, DFHTD, DFHMC, DFHMGP00, DFHRCRP, DFHRUP, DFHSIII, DFHSTP, DFHSTTD, DFHTACP, DFHTDRP, DFHTEPM, DFHTQ, DFHTR, DFHTSRP, DFHWKP, DFHZNAC
Description: DFHTDP is a load module link-edited from object modules for DFHTDA, DFHTDEXL, and DFHTDX.

DFHTDQ
Entry points: DFHTDBNA
Called by: DFHTDA
Description: DFHTDQ is a load module link-edited from object modules for DFHTDB.

DFHTDRM
Entry points: DFHTDRM
Called by: DFHDBP
Description: DFHTDRM is the transient data recovery manager processor. If transient data has any outstanding resources, DFHTDRM is called at phase 1 syncpoint (or backout). For phase 1 syncpoint (or backout) requests, DFHTDRM issues a request to mainline transient data (DFHTDA) to reset any resources that have not yet been released.

DFHTDRP
Entry points: DFHTDRNA
Called by: DFHTDX
Description: DFHTDRP handles transient data recovery during CICS initialization. In particular, DFHTDRP:
- Adds the entries found in the DCT load module by calling the DFHTDTM gate.
- Restores input and output cursors for intrapartition queues on warm start; the cursors are recovered by DFHRUP on emergency restart
- Restores the CI state map on warm start
- Opens extrapartition queues
- Opens the intrapartition data set
- Recovers the CI state map on emergency restart.

DFHTDTM
Entry points: DFHTDTM
Called by: DFHALP, DFHIEQMS, DFHIEQSO, DFHSE, DFHSZRPM, DFHTDRP
Description: DFHTDTM manages the entries in the destination control table. It is used to add, update and delete entries in this table and records images of each entry on the global catalog for use during a warm start or emergency restart. It allows table entries to be inquired upon.

DFHTDX
Entry points: DFHTDXNA
Called by: Task initiation
Description: DFHTDX is the initial program invoked by the transient data recovery task. It links to program DFHTDRP.

DFHTEP
Entry points: DFHTEPNA
Called by: DFHTACP
Description: The terminal error program is invoked by DFHTACP using a DFHPC CTYP=LINK_URM macro. The sample DFHTEP (invoked only if there is no customer-supplied version) puts a terminal out of service if the number of terminal errors detected by DFHTACP exceeds default values contained in DFHTEP tables.

DFHTMP
Entry points: DFHTMPNA
Called by: DFHTMAC
Description: The table management program performs locates, adds, deletes, locks, and unlocks to entries in
certain CICS tables. DFHTMP uses a hash table for these operations.

The main subroutines of DFHTMP are:
- CHKTTC - Check table type code
- COMMIT - Commit table changes
- CRTCLE - Create a change list element
- CRTDWE - Create deferred work element
- DELDWE - Cancel deferred work element
- DEQALDE - Dequeue on directory element
- DEQUE - Dequeue on table modification
- DYNHASH - Dynamic re-hash
- ENQDEQDE - Enqueue/dequeue on directory element
- ENQUEUE - Enqueue on table modification
- GET_STORAGE - Get storage from the CICS shared subpool
- GET_TASK_STORAGE - Get task lifetime 31-bit storage
- GET_TASK_STORAGE_COND - Get task lifetime 31-bit storage (conditionally)
- GET_STORAGE_FAILURE - Get storage failure routine
- FREE_STORAGE - Release storage from the CICS shared subpool
- FREE_TASK_STORAGE - Release task lifetime 31-bit storage
- LOCATE_PREVIOUS_DE - Locate previous directory element in collating series
- LOCATETE - Locate a table/directory entry
- LOCDFIRE - Locate a free directory element
- NOTERL - Note Read Lock
- SETABORD - Set up alphabetic ordering pointer for a given table type
- TMFINDLOCK - Find a read lock
- TMPDWEPP - Deferred work element processor
- TMSLOCK - Set a read lock
- TMUNLOCK - Release a read lock
- UNQUIES - Unquiesce a directory element.

DFHTON
Entry points: DFHTONNA
Called by: DFHDBP, DFHSPP
Description: The terminal object resolution module is called by DFHDBP or DFHSPP during DWE processing for DFHTOR. It calls DFHTOR with end-LUW-cancel or end-LUW-commit code to perform cancel or commit of changes to TERMINAL, TYPETERM, CONNECTION, or SESSIONS definitions.

DFHTOR
Entry points: DFHTORNA
Called by: DFHAMP, DFHTON
Description: DFHTOR is the terminal object resolution program. DFHAMP calls DFHTOR for a TERMINAL, TYPETERM, CONNECTION, or SESSIONS object in a CICS system definition (CSD) file that is being installed, or when DFHAMP encounters an end-of-group. DFHTOR processes the objects and passes them to the terminal control builder program (DFHZCQ). The DFHTON entry is used for DWE processing.

DFHTORP
Entry points: DFHTORNA
Called by: DFHSII1
Description: DFHTORP is the terminal object recovery program. It is called during CICS initialization to purge TYPETERM and model terminal definitions from the catalog on a cold start, and to recover these definitions on an emergency restart.

DFHTPPA$, DFHTPP1$
Entry points: DFHTPPNA
Called by: DFHDSB, DFHM32
Description: The terminal page processor program handles DFHBMS TYPE=OUT, STORE, and RETURN requests. If OUT, DFHTPP sends the complete page using DFHTC macro requests; if STORE, the page is sent to temporary storage; and if RETURN, no output operation takes place but the page is returned to the application program.

The main subroutines of DFHTPP are:
- TPNODDS - TYPE=STORE (PAGING) requests
- TP OUT - TYPE=OUT (TERMINAL) requests (the macro DFHTOM is used by both DFHTPP and DFHTPR to handle output to terminals)
- TRPRETPG - TYPE=RETURN (SET) requests.

Returns to: DFHPBP

DFHTPQ
Entry points: DFHTPQNA
Called by: DFHICP, DFHMCP, DFHTCP
Description: The undelivered messages cleanup program is initiated periodically in order to cancel the delivery of BMS messages that have been placed in temporary storage, but have remained undelivered for an interval exceeding the purge delay time interval specified by the PRGDLAY system initialization parameter, if this has a nonzero value.

DFHTPR
Entry points: DFHTPRNA
Called by: DFHMCP, DFHTCP
Description: The terminal page retrieval program (transaction CSPG) is invoked:
- By automatic transaction initiation as a result of a SCHEDULE issued by DFHTPS
- By a DFHPGLK LINK from DFHMCP, when CTRL=RETAIN or RELEASE on DFHBMS TYPE=PAGEOUT (RETAIN or RELEASE on SEND PAGE at command level)
- When CSPG or an operator paging command is entered at a terminal.

If the message is autopaged, DFHTPR retrieves the pages of the message in order, transmits them to the terminal, and then purges the message. Otherwise DFHTPR runs pseudo-conversationally. All further input is passed to DFHTPR, until the message is purged explicitly or implicitly. If the input is a valid paging command (page retrieval, page copy, page
purge, or page chaining), it is processed. It is rejected if explicit purge is required, or passed back to normal task initiation if automatic purge is allowed.

The main subsections of DFHTPR are:
- DFHMSPUT - Send error message to terminal
- TPENCCHN - Encode and execute page chain
- TPENCCOP - Encode and execute page copy
- TPENCPUR - Execute page purge
- TPENCRET - Encode page retrieval
- TPERETA - Reset to autopaging
- TPERETQ - Page query
- TPEEXIT - Exit from program
- TPEXPUR - Execute page purge
- TPEXRET - Execute page retrieval
- TPTSGET - Get MCR or page from temporary storage

DFHTPS

Entry points: DFHTPSNA
Called by: DFHICP, DHMCP

Description: The terminal page scheduling program (transaction CSPS) is invoked for each terminal type to which a BMS logical message built with TYPE=STORE is to be sent. For each terminal designated by the originating application program, DFHTPR is scheduled to display the first page of the logical message if the terminal is in paging status, or the complete message if it is in autopage status. DFHTPS contains the following major subsections, each dealing with a separate function:

- DFHTPSNA—used when DFHTPS is invoked by automatic initiation on expiry of ICE, and as a result of an IC PUT request issued by DFHMC (there is no associated terminal). This invocation schedules CSPG for terminals on this system, and schedules CSPS on the link to each remote system which owns terminals contained in the route list for the message (that is the function of TPS02000).
- TPS01000—used when DFHTPS is linked to from DFHMC for direct paging requests to a terminal on a remote system. The task has a surrogate TCTTE as its primary facility, and owns a relay link connected to the terminal owning system. This section ships the pages of the message to the terminal-owning region, where it is re-created by the relay program (DFHAPRT) which issues BMS, STORE, TEXT, NOEDIT, and PAGEOUT requests.
- TPS02000—used when DFHTPS is scheduled by TPS01000 to run against the link to a remote system. This routine ships the logical message to the remote system and deletes the terminals on the remote system from the terminal list in the original message control record. (TPS03000 receives the information at the remote system.)
- TPS03000—used when DFHTPS is invoked by an ATTACH request from a remote system (that is, originated by TPS01000 or TPS02000). This routine receives the shipped logical message and issues BMS

ROUTE, TEXTBLD, NOEDIT, and PAGEOUT requests to re-create the logical message on the terminal-owning region.

DFHTPS contains the following subroutine:
- TPSHIPM—ships a complete logical message.

DFHTRAP

Entry points: DFHTRANA
Called by: DFHTRPT

Description: The FE global trap/trace exit is provided for diagnostic use only under the guidance of service personnel.

DFHTR530/AMDUSREF

Entry points: DFHTRPRG
Called by: IPCS

Description: The CICS GTF trace formatting routine is invoked by IPCS processing of the GTFTRACE keyword when a CICS entry (USR F6C, format ID X'EF') is encountered. For each entry, it writes a line containing the job name and then formats the entry in the same form as DFHTU530 does for an auxiliary trace print. AMDUSREF is defined as an alias for DFHTR530 because IPCS looks for a program called AMDUSRxx to format entries with format ID xx.

DFHTRP

Entry points: DFHTRPNA
Called by: Many AP domain modules

Description: The trace control program translates DFHTR, DFHTRACE, and DFHLFM macro requests to write trace entries into TR domain TRACE_PUT requests. DFHTRP collects the data required in the trace for the specified trace ID into a standard layout and issues the TRACE_PUT call. For requests to change the various trace flags that control tracing, DFHTRP issues KEDD format calls to the kernel domain.

DFHTRZCP

Entry points: DFHTRZCP
Called by: CEDA transaction, DFHTCRP, DFHTOR

Description: DFHTRZCP builds a terminal builder parameter set.

DFHTRZIP

Entry points: DFHTRZIP
Called by: CEDA transaction, DFHTCRP, DFHTOR

Description: DFHTRZIP builds a chain of builder parameter sets for sessions.
**DFHTRZPP**

Entry points: DFHTRZPP

Called by: CEDA transaction, DFHTCRP, DFHTOR

Description: DFHTRZPP builds a pool builder parameter set.

**DFHTRZXP**

Entry points: DFHTRZXP

Called by: CEDA transaction, DFHTCRP, DFHTOR

Description: DFHTRZXP builds a connection builder parameter set.

**DFHTRZYP**

Entry points: DFHTRZYP

Called by: CEDA transaction, DFHTCRP, DFHTOR

Description: DFHTRZYP builds a TYPETERM builder parameter set.

**DFHTRZZP**

Entry points: DFHTRZZP

Called by: CEDA transaction, DFHTCRP, DFHTOR

Description: DFHTRZZP merges a TYPETERM builder parameter set into a terminal builder parameter set.

**DFHTSP**

Entry points: DFHTSPNA

Called by: DFHACP, DFHAKP, DFHALP, DFHCRQ, DFHDIP, DFHDIP, DFHESE, DFHETS, DFHICP, DFHICP, DFHMS, DFHRTE, DFHSII, DFHTSP, DFHTCRP, DFHTIP, DFHTIP, DFHTSP, DFHTSP, DFHSP, DFHZISP, DFHZRAQ, DFHZRAP, DFHZRP

Description: The temporary-storage program services DFHTS requests. It maintains the tables, directories, and maps necessary to keep track of every temporary-storage record and of available space on the VSAM auxiliary storage or in main storage. The main subroutine of DFHTSP is DFHTSPAM, which manages auxiliary storage (including multiple buffers and strings).

**DFHTU530**

Entry points: DFHTRPRA

Called by: MVS

Description: The trace utility program formats and prints trace records stored on the auxiliary trace data set. This utility program is run as a separate job, and extracts selected trace entries as specified on parameter statements supplied as part of the input to the program.

**DFHUCNV**

Entry points: DFHUCNV

Called by: DFHCCNV

Description: DFHUCNV is a sample program for CICS OS/2 user data conversion. Users can write their own version of DFHUCNV to apply any conversion. If specified, a user-supplied conversion is applied before the standard conversion. DFHUCNV is invoked for each EXEC CICS request and reply that has resulted from a CICS OS/2 function shipping request and may require conversion of user data from ASCII to EBCDIC (inbound from CICS OS/2) or from EBCDIC to ASCII (outbound). DFHCCNV issues an EXEC CICS LINK to DFHUCNV before attempting any standard conversions. This allows a user program to convert data of type USERDATA, as defined in the CICS OS/2 conversion macros (DFHCNV).

The sample program obtains addressability to the COMMAREA passed to it, and checks that the request is a temporary-storage (TS) request. Then it checks that DFHCCNV managed to locate a conversion template for the resource (a TS queue) with this name, and scans and checks the template using the supplied template pointer and length. If the check is successful, the program translates the user data field as appropriate.

**DFHUEH**

Entry points: DFHUEHNA

Called by: CICS management modules containing exit points

Description: The user exit handler is the link between an exit point in a CICS management module in the AP domain, and the user code. DFHUEH invokes in turn each started exit program for that exit point, passing a parameter list defined in the CICS management module.

**DFHUEM**

Entry points: DFHUENMA

Called by: DFHEIP

Description: The EXEC interface processor for the ENABLE, DISABLE, and EXTRACT user exit commands.

**DFHUSBP**

Entry points: DFHUSBNA

Called by: DFHRCRP

Description: The user backout program sends records,
journaled by the user to the system log, to a user exit during emergency restart. The records are extracted by DFHRU from the restart data set. They may exist for any logical unit of work, whether in flight or not, depending on the JCRSTRID value specified when the record was written.

**DFHWCCS**

**Entry points:** DFHWCCS  
**Called by:** Many CAVM modules  
**Description:** DFHWCCS provides common services for the CAVM:  
- MVS FREEMAIN  
- MVS GETMAIN  
- MVS POST  
- Message or MVS ABEND  
- Create CAVM process block.  
**Returns to:** MVS abend, caller

**DFHWCGNT**

**Entry points:** DFHWCGNA  
**Description:** DFHWCGNT is the entry point list for CAVM modules above the 16MB line.

**DFHWATT**

**Entry points:** DFHWATT  
**Called by:** DFHWDNA, DFHWMG1, DFHWMP1, DFHWXPI  
**Description:** DFHWDTT creates the CAVM process.

**DFHWDNA**

**Entry points:** DFHWDNA  
**Called by:** DFHWSRTR  
**Description:** DFHWDNA attaches the initial CAVM process. It sets up lock tables, the dispatcher control area, the LIFO control area, and the dispatcher ESPIE and ESTAE exits.  
**Returns to:** DFHWDISP

**DFHWDISP**

**Entry points:** DFHWDISP, DFHWDNA  
**Called by:** DFHWAT, DFHWDNA  
**Description:** DFHWDISP is the CAVM process dispatcher. It dispatches the next ready CAVM process, or waits for an external event. It dispatches the initial CAVM process.  
**Returns to:** Dispatched process, caller of DFHWDNA

**DFHWDSRP**

**Entry points:** DFHWDSRP  
**Called by:** DFHWDNA, CAVM program check/abend  
**Description:** DFHWDSRP establishes the ESPIE/ESTAE CAVM process. It performs CAVM process error handling for processes with ESPIE or ESTAE routines.

**DFHWGWAT**

**Entry points:** DFHWGWAT  
**Called by:** Many CAVM modules  
**Description:** DFHWGWAT causes the current CAVM process to wait for specific events.  
**Returns to:** DFHWDISP

**DFHWKP**

**Entry points:** DFHWKPNA  
**Called by:** DFHSRT  
**Description:** DFHWKP takes a warm keypoint at the normal termination of CICS. This program is part of the restart component.

**DFHWLGET**

**Entry points:** DFHWLGET  
**Called by:** Many CAVM modules  
**Description:** DFHWLGET gets the LIFO stack entry for CAVM modules running above the 16MB line.

**DFHWXPI**

**Entry points:** DFHWXPI  
**Called by:** Many CAVM modules  
**Description:** DFHWXPI gets the LIFO stack entry for CAVM modules running above the 16MB line.

**DFHWXPI**

**Entry points:** DFHWXPI  
**Called by:** Many CAVM modules  
**Description:** DFHWXPI gets the LIFO stack entry for CAVM modules running above the 16MB line.
DFHWMI
Entry points:  DFHWMI
Called by:  DFHWSXPI
Description:  DFHWMI allocates the CAVM message-manager communication area. It calls each of the main message-manager modules, which then initialize themselves.

DFHWMMT
Entry points:  DFHWMMT
Called by:  DFHWMRD, DFHWWMR
Description:  DFHWMMT provides VSAM GET and PUT services for the CAVM message data set.

DFHWMPG
Entry points:  DFHWMPG
Called by:  DFHWMP1, DFHWWMR
Description:  DFHWMPG copies message data into the buffer provided by the user of PUTMSG, PUTREQ, PUTRSP, and CAVM message-manager services. It provides an ESPIE routine to handle program checks occurring during the copying.

DFHWMP1
Entry points:  DFHWMP1
Called by:  DFHWMI, DFHWDISP, DFHWDSRP
Description:  DFHWMP1 is the main module of the CAVM message-manager PUT MESSAGE service. It is called by DFHWMI to initialize service, and attach itself as a message-writer CAVM process; by DFHWDISP to run as a message-writer CAVM process that writes messages to the CAVM message data set; and by DFHWDSRP to handle ESPIE and ESTAE exits for the message writer.

DFHWMQG
Entry points:  DFHWMQG
Called by:  DFHWMS20
Description:  DFHWMQG runs under the CICS TCB above the 16MB line. It processes GETMSG CAVM message-manager requests. It waits for a message to arrive, then copies from the main-memory message queue created by the CAVM message-reader process.

DFHWMS
Entry points:  DFHWMSNA
Called by:  Users of CAVM message services
Description:  The CAVM message-manager service interface routine runs under the CICS TCB above the 16MB line. It builds a dummy CAVM process block, so that subsequent modules can run in an XRF LIFO environment, and calls DFHWMS20 to process a request passed by the caller.

DFHWMS20
Entry points:  DFHWMS20
Description:  The CAVM message-manager service interface selects the request type and passes requests to DFHWMQP (PUTMSG, PUTREQ, PUTRSP) or DFHWMQG (GETMSG).
DFHWMWR
Entry points: DFHWMWR
Called by: DFHWMP1
Description: The CAVM message-manager message write routine takes data from PUTMSG requests and copies them into CI buffers to be written to the CAVM message data sets.

DFHWOS
Entry points: DFHWOSNA
Description: The overseer startup module loads DFHWOSA and passes control to it.

DFHWOSA
Entry points: DFHWOSNA
Called by: DFHWOS
Description: The overseer services initialization module processes control parameters, loads DFHWOSB, and sets up entry points for overseer services.

DFHWOSB
Entry points: DFHWOSNA
Called by: Overseer program
Description: The overseer service module processes requests from the overseer program which are issued by the DFHWOSM macro.

DFHWSRTR
Entry points: DFHWSMNA
Called by: DFHXRA, MVS after attach of new TCB
Description: The CAVM state-management request router and subtask entry point is the initial entry point for a CAVM task attached by DFHWSSN1 to process the CAVM SIGNON command. It calls DFHWSSN2 to continue the processing of the SIGNON request and, if it is accepted, calls DFHWSDINA to attach the tick generator module DFHWSTI as the first and highest-priority CAVM process. It is called under the CICS TCB to queue the CAVM TAKEOVER command for processing by the CAVM task, and to initiate processing of the CAVM SIGNOFF command by detaching the CAVM task. DFHWSRTR is the initial entry point for MVS subtasks attached by the CAVM task to perform various functions, such as issuing requests for CSVC services, or formatting new CAVM data sets when they are used for the first time.

DFHWSSN1
Entry points: DFHWSSNA
Called by: DFHXRA
Description: DFHWSSN1 is the CAVM state management SIGNON initial entry point. The CICS task issues an MVS LINK, specifying load module DFHWSSON to perform a CAVM SIGNON request. DFHWSSN1 attaches the CAVM task to execute the request, waits to see if it is successful, detaches the task and, if it is not successful, reports the result to CICS.

DFHWSSN2
Entry points: DFHWSSN2
Called by: DFHWSRTR
Description: The CAVM state management SIGNOFF request handler is entered under the CAVM TCB to process a CAVM SIGNON request. It allocates storage for, and initializes, key CAVM control blocks, sets up DFHWSSOF as an ESTAE exit, calls DFHWSSN3 to OPEN the CAVM data sets, reads the state management record from the control data set, uses the JES inquire-job-status CSVC service provided by DHWTI, and looks for surveillance signals from other CAVM users to check whether the environment is such that the requested SIGNON can be accepted. It prompts the operator for job status information if necessary. If SIGNON is accepted, it updates the state management record and status CIs to record that this job has signed on to the CAVM. When possible, it also cleans up out-of-date information in the CAVM data sets left behind by jobs that were unable to sign off properly before terminating.

DFHWSSN3
Entry points: DFHWSSN3
Called by: DFHWSSN2
Description: The CAVM state management data set initialization routine builds ACBs, and opens and validates the CAVM control and message data sets for CAVM SIGNON. It builds the reserve parameter list for serializing accesses to the control data set. If new CAVM data sets are being used for the first time, it attaches an MVS subtask to record relevant information in each data set’s control interval, and to format the CIs needed by state management.

DFHWSSOF
Entry points: DFHWSSOF
Called by: MVS recovery/termination manager
Description: DFHWSSOF is the CAVM state management SIGNOFF request handler. During SIGNON processing, this module is established as an
ESTAE exit for the CAVM task. It purges outstanding I/O requests, reads the state management record from the control data set, and searches it to see if this job has signed on to the CAVM. If so, it updates the status CI and state management record to indicate that the job has signed off. It makes the TAKEOVER message available to DFHWSRTR when an active system signs off after takeover has started.

DFHWSSR
Entry points: DFHWSSR
Called by: DFHWDISP
Description: The CAVM surveillance status reader runs as a process controlled by the XRF dispatcher, DFHWDISP. It reads the status CI of the partner system from the control data set or the message data set, generates internal CAVM events, and drives the NOTIFY exit when the partner’s status changes, or its surveillance signals cease. For an alternate system, it monitors and records the time-of-day clock difference when the active system is running in a different CEC.

DFHWSSW
Entry points: DFHWSSW
Called by: DFHWDISP
Description: The CAVM surveillance status writer runs as a CAVM process controlled by the CAVM dispatcher, DFHWDISP. It writes a system’s current status to its status CI in the control data set, or the message data set, to make it available to its partner and to provide a surveillance signal; generates an internal CAVM event when a status write completes; and puts the current time-of-day clock reading in the status CI to permit DFHWSSR to deduce the time-of-day clock difference when the active system and the alternate system are running in different CECs.

DFHWSTI
Entry points: DFHWSTI
Called by: DFHWDISP
Description: The CAVM surveillance tick generator and CICS status monitor runs as a CAVM process controlled by the CAVM dispatcher, DFHWDISP. It issues an MVS STIMER for the surveillance interval and, when this expires, generates an internal CAVM clock-tick event, calls the inquire-CICS-status exit, and schedules the surveillance status writer processes, to cause a surveillance signal reporting this system’s current status to be written to the control data set or the message data set.

DFHWSTKV
Entry points: DFHWSTKV
Called by: DFHWSSR
Description: The CAVM state management TAKEOVER request handler runs as a CAVM process controlled by the CAVM dispatcher DFHWDISP. When a new active SIGNON has been detected, it reads the state management record from the control data set and attaches an MVS subtask to invoke DFHWTI’s validate-CLT CSVC service. When a TAKEOVER command has been issued, it reads the state management record, validates the TAKEOVER request, and attaches an MVS subtask to use DFHWTI’s JES inquire-job-status service to determine the current state of the active system.

If the active system is still signed on to CAVM, it updates the state management record to indicate that a takeover is in progress, places the TAKEOVER message for the active system in the alternate system’s status, and attaches an MVS subtask to invoke DFHWTI’s TAKEOVER-initiate service.

After the active system has signed off (or terminated), it requests DFHWSSR to read the active system’s final status, quiessces surveillance processing, and updates the state management record and status CIs to indicate the stage reached by takeover. It then arranges for surveillance processing to be resumed in active mode. It attaches an MVS subtask to invoke DFHWTI’s process-CLT CSVC service if necessary.

When the active system has finally terminated, it updates the state management record to take its place as the new active system, generates internal CAVM events, and calls the NOTIFY exit to report the progress of the TAKEOVER request, including acceptability of the time-of-day clock reading. It terminates by returning to DFHWDISP.

DFHWSXPI
Entry points: DFHWSXPI
Called by: DFHWSTI
Description: The CAVM state management CAVM process initialization runs under the tick generator CAVM process towards the end of SIGNON. It attaches the TAKEOVER CAVM process (alternate systems only), two status writer CAVM processes, and two status reader CAVM processes, and then calls the CAVM message management initialization module.

DFHWTI
Entry points: DFHWTINA
Called by: DFHCSCVC from: DFHWSSN2, DFHWSTKV, DFHZXSTS
Description: Takeover initiation is the primary
function of this module, and is requested by CAVM state management at takeover to terminate the CICS active system issue commands in the CLT, and wait until the CICS active system terminates. Other XRF services provided by this module are to determine whether a job is running, to issue the operator commands for the overseer program, to issue MODIFY USERVAR to VTAM, to validate the CLT, and to process the CLT.

---

**DFHWTRP**

Entry points: DFHWTRP  
Called by: Many CAVM modules  
Description: DFHWTRP makes a trace entry in the CAVM main-memory trace table.

---

**DFHXCP**

Entry points: DFHXCPNA  
Called by: DFHKCP  
Description: DFHXCP processes DFHKC CANCEL, CHAP, RESUME, SUSPEND, and WAIT macro calls to the transaction manager.

---

**DFHXCPC**

Entry points: DFHXCPC  
Called by: DFHKCP  
Description: DFHXCPC processes DFHKC ATTACH, CHANGE, DEQ, DEQALL, ENQ, and SRB macro calls to the transaction manager. It receives DFHKC INITIALIZE, REPLACE, and WAITINIT macro calls to the transaction manager and passes them on to DFHKCQ.

---

**DFHXCP1**

Entry points: DFHXCP1  
Called by: DFHXCPC  
Description: DFHXCP1 finds a new range of free transaction numbers when the current range has been used up.

---

**DFHXFP**

Entry points: DFHXFPNA  
Called by: DFHISP, DFHMIRS  
Description: The online data transformation program takes data addressed from a parameter list (command-level or DL/I), and constructs an FMH suitable for transmission to a remote ISC or MRO system; DFHXFP also performs the reverse transformation.

---

**DFHXFQ**

Entry points: DFHXFQNA  
Called by: DFHXEPRH  
Description: The batch data transformation program executes in an EXCI region. DFHXFQ takes data addressed from a DPL parameter list and constructs an FMH suitable for passing to the online region; DFHXFQ also performs the reverse transformation.

---

**DFHXFX**

Entry points: DFHXFXNA  
Called by: DFHISP, DFHMIRS  
Description: DFHXFX performs the same logical transformations of function shipping requests as DFHXFP but in a manner that is optimized for the MRO environment. It is not used for the transformation of DL/I requests; these are processed by DFHXFP.

---

**DFHXMP**

Entry points: DFHXMPNA  
Called by: DFHDRPE, DFHZCX  
Description: The cross-memory program is invoked by a program call (PC) instruction and uses MVS cross-memory services to pass data from one subsystem to another within the same processing unit. The communicating subsystems are usually in CICS address spaces, but DFHXMP does not assume this.

---

**DFHXRA**

Entry points: DFHXRA  
Called by: DFHAPDM, DFHCSSC, DFHCXCU, DFHDBCX, DFHDBCX, DFHDCS, DFHSCS, DFHSTP, DFHTCPC, DFHTDTP, DFHXRCP, DFHXRSP, DFHZNAC, DFHZOPN, DFHZSLS  
Description: DFHXRA is the program that executes the DFHXR macro. It runs under the CICS TCB in AMODE(24). In general, it uses CICS macros to invoke other services. Exceptions are MVS LINK to DFHWSSON to sign on to the CAVM, and MVS LOAD and DELETE for DFHWSSMS to sign off from the CAVM, and to initiate takeover. It invokes global user exit XXRSTAT, which can lead to the abend 208.

---

**DFHXRB**

Entry points: DFHXRBNA  
Called by: DFHWDSRP, DFHWMQH, DFHWMDR, DFHWSSR, DFHWSTKV  
Description: DFHXRB is the XRF notify exit program. Its address is passed to the CAVM when CICS signs on to the CAVM. It runs under the CAVM TCB in
AMODE(31); reacts to events detected by various CAVM modules; and creates a queue of work elements (chained from XRWECHN) to be processed by DFHXRSP.

**DFHXRC**

**Entry points:** DFHXRCNA

**Called by:** DFHWSSN2, DFHWSTI

**Description:** DFHXRC is the CICS-status exit program. Its address is passed to the CAVM when CICS signs on to the CAVM. It runs under the CAVM TCB in AMODE(31), and returns the latest CICS-status data to be written to the state management data set.

**DFHXRCP**

**Entry points:** DFHXRCNA

**Called by:** DFHPCP

**Description:** The XRF console communication task runs under the CICS TCB in AMODE(24). It processes MODIFY commands received by CICS during initialization of the alternate system. It initiates takeover, shuts down the active system, and manages trace and dump as required.

**DFHXRE**

**Entry points:** DFHXRENA

**Called by:** DFHPCP

**Description:** The XRF startup program is the entry point for the system task attached by DFHXRA. It links to DFHXRE, whichever module was indicated by DFHXRA.

**DFHXRP**

**Entry points:** DFHXRANA

**Called by:** Not applicable

**Description:** DFHXRP consists of six object modules link-edited together:

- **DFHXRA** - XRF request processor
- **DFHXRB** - XRF NOTIFY exit program
- **DFHXRC** - XRF inquire status exit program
- **DFHXRE** - XRF startup program
- **DFHXRF** - XRF CAVM sign-off interface
- **DFHWMS** - CAVM message manager service interface.

It is loaded by DFHSIB1.

**DFHXRSP**

**Entry points:** DFHXRSNA

**Called by:** DFHXRA

**Description:** DFHXRSP is the XRF surveillance program, which runs as a program under a CICS transaction. It runs under the CICS TCB in AMODE(31); processes the queue of work elements created by DFHXRB; attaches the catch-up transaction CXCU, initiates takeover, and shuts down CICS as required; and can issue abends 206 and 207.

**DFHXSMN**

**Entry points:** DFHXSMNA

**Called by:** DFHBSTS, DFHCRNP, DFHDLIDP, DFHDLIRP, DFHEDFP, DFHEIPSE, DFHSII1, DFHSUSN, DFHSUXS, DFHTACCP, DFHZSUP

**Description:** The security manager is invoked by the DFHSEC macro, and provides an interface to the external security manager (ESM). DFHXSMN validates the parameters passed, then calls DFHXSMX as a general-purpose subroutine to invoke the ESM.

**DFHXSMX**

**Entry points:** DFHXSMNA

**Called by:** DFHXSMN

**Description:** DFHXSMX is the subroutine used by the security manager to invoke the external security manager (ESM). For resource checking, this routine first issues the MVS RACROUTE REQUEST=FASTAUTH macro, which calls the ESM in problem state. All other security functions require the caller to be in supervisor state. For these functions, and for a failed FASTAUTH call that requires logging, the CICS SVC is issued under a general purpose subtask, entered by the DFHSK macro, to shield the main CICS task from any imbedded waits that may occur in the ESM.

**DFHXSS**

**Entry points:** DFHXSSNA

**Called by:** DFHCSVC

**Description:** DFHXSS invokes the external security manager (ESM) for all functions that need to be invoked while authorized, except for the EXTRACT functions for which it passes control to DFHXSSB.

**DFHXSSB**

**Entry points:** DFHXSSB

**Called by:** DFHXSS

**Description:** This module extracts data from the ESM’s database. DFHXSSB extracts userid-related data at signon time, and session key information at LU6.2 session bind time. It uses the MVS RACROUTE REQUEST=EXTRACT macro.
DFHXSWM

**Entry points:** DFHXSWM

**Called by:** DFHXSMN

**Description:** DFHXSWM passes and retrieves messages to and from the XRF alternate system to see if security initialization is required in the XRF environment.

DFHXTCI

**Entry points:** DFHXTCI

**Description:** DFHXTCI is the transaction invoked when the alternate system begins a takeover. It examines the TCT to locate the terminals with XRF backup sessions, and queues these TCTTEs to DFHZSES for the SESSIONC CONTROL=SWITCH command.

DFHXTP

**Entry points:** DFHXTPNA

**Called by:** DFHTPS, DFHZTSP, DFHZXRL, DFHZXRT

**Description:** The terminal sharing transformation program comprises four logical modules (known as transformers 1 through 4). DFHXTP transforms routing requests into the LU type 6 format for shipping to a remote CICS address space.

DFHZABD

**Entry points:** DFHZABD1

**Called by:** TC CTYPE= requests

**Description:** If a TC CTYPE request is issued when ZCP has been generated without VTAM support, DFHZABD is invoked to abend the transaction.

DFHZACT

**Entry points:** DFHZACT1

**Called by:** DFHZDSP

**Description:** The activate scan routine scans the four TCTTE activity queues: activate, log, wait, and NACP. DFHZACT scans the activate queue for request bits that may be set in the TCTTEs; for each request, DFHZACT calls the appropriate module. If no requests are outstanding, the TCTTE is removed from the queue. If the NACP queue is not empty, DFHZACT attaches DFHZNAC (if not already attached). Similarly, if the log queue is not empty, DFHZACT attaches DFHZRLG. DFHZACT scans the wait queue. If automatic resource definition is in the system, DFHZACT looks for any corresponding work elements. For each work element, DFHZATA is attached.

DFHZAIT

**Entry points:** DFHZAIT1

**Called by:** DFHSIF1

**Description:** The attach initialization tables routine initializes local tables used by the mainline task-attach routine, DFHZATT. DFHZAIT generates the page command table from information supplied by the system initialization table, modifying it for use by DFHZATT. DFHZAIT also initializes the transaction code delimiter table.

DFHZAND

**Entry points:** DFHZAND1

**Called by:** DFHZARQ

**Description:** The abend control block builder is used to assist in building the transaction abend block when an abend has occurred in an interconnected system. Its function is to extract the error sense bytes, and the diagnostic message sent by the other system, and to copy these into the block. As an initial step in its processing, DFHZAND acquires storage for the block itself.

DFHZARER

**Entry points:** DFHZARER

**Called by:** DFHZARL, DFHZARR, DFHZARRA

**Description:** DFHZARER tidies up after an LU6.2 protocol error or session failure has been detected. For some errors, it calls DFHZNAC.

DFHZARL

**Entry points:** DFHZARL1

**Called by:** DFHACP, DFHCPCBA, DFHCPCCLC, DFHCRS, DFHEGL, DFHETL, DFHLUP, DFHXFP, DFHXTP, DFHZARL, DFHZARM, DFHZERH, DFHZISP, DFHZLUS, DFHZSUP, DFHZTSP, DFHZXRL, DFHZXRT

**Description:** DFHZARL is called via the DFHLUC macro, which passes the LU6.2 request in a parameter list mapped by the DFHLUCDS DSECT. If the request is for a remote APPC device, DFHZARL passes the parameter list to DFHZXRL for processing. (APPC is advanced program-to-program communication.) Otherwise, it examines the parameter list to determine the function required. Most functions are processed by DFHZARL. However, it calls the following modules as indicated:

- DFHZARER - Protocol errors and exceptions
- DFHZARR - RECEIVE requests
- DFHZARRA - FREE-STORE requests
- DFHZERH - Handling FMH7s and negative responses
DFHZISP - ALLOCATE and FREE requests
DFHZRVL - Receiving SNA indicators from VTAM
DFHZSDL - Sending data to VTAM.

It also manages the logical receive buffer pointers TCTERBLA and TCTERBLL in a consistent manner with the physical receive buffer pointers TCTERBA and TCTERBDL, as (address, length) pairs.

DFHZARM
Entry points: DFHZARM1
Called by: DFHZARQ, DFHETL, DFHZISP
Description: DFHZARM handles DFHTC macros for LU6.2 sessions.

DFHZARQ
Entry points: DFHZARQ1
Called by: DFHETC, DFHTC macro
Description: The application request interface module analyzes the terminal control request from the application. For a VTAM terminal, it sets the appropriate flags and calls the required module or adds the TCTTE to the activate chain.

DFHZARR
Entry points: DFHZARR
Called by: DFHZARL
Description: DFHZARR controls the receive function for LU6.2 application requests. It calls DFHZARRC to decide what to process next, or whether it is necessary to call its inline subroutine DFHZARR1 to receive more data. Then it processes the returned item, and decides whether receive is complete. If the receive is not complete, DFHZARR loops, calling DFHZARRC and processing the returned item, until enough data has been received. DFHZARR uses the inline subroutine DFHZARR0 and the DFHZARRA module to control various receive buffers. It also uses DFHZARRF to receive FMH7s and negative responses, DFHZUSR to control the conversation state, and the inline subroutine DFHZARR1 to handle the type of receive and how much data is to be received.

DFHZARR0 is responsible for updating the logical buffer pointers TCTERBLA and TCTERBLL, shifting up data in the LU6.2 receive buffer, and resetting associated indicators, for example, TCTECCDR in the TCTTE LUC extension.

DFHZARR1 is responsible for setting fields TCTEMINL and TCTEMAXL in the TCTTE LUC extension to inform DFHZRVL how much data to receive and whether the request is a receive immediate or a receive and wait. DFHZARR1 calls DFHZARR0 to shift up data in the LU6.2 receive buffer, and then calls DFHZRVL to receive RUs from VTAM by placing requests on the active chain.

DFHZARRA
Entry points: DFHZARRA
Called by: DFHZARL, DFHZARR
Description: DFHZARRA controls all functions concerned with the LU6.2 application receive buffer. These include GETMAIN and FREEMAIN of buffers, copying data into a buffer, and updating the pointer to the next free slot.

DFHZARRC
Entry points: DFHZARRC
Called by: DFHZARR
Description: DFHZARRC is responsible for examining what has been received from VTAM on a particular session (for example, data, PS headers, FMH7s, and indicators), and for deciding what should be processed next on behalf of the application. The result is returned to DFHZARR.

DFHZARRF
Entry points: DFHZARRF
Called by: DFHZARR
Description: DFHZARRF receives LU6.2 FMH7s and negative responses. It calls the DFHZARR0 subroutine to shift up data in the LU6.2 receive buffer, and then calls DFHZERH.

DFHZASX
Entry points: DFHZASX1
Called by: VTAM
Description: The asynchronous command exit module is called by VTAM if an asynchronous command is received. The only such commands are request shutdown, quiesce at end of chain, release quiesce, and signal. DFHZASX sets up the TCTTE appropriately and returns control to VTAM.

DFHZATA
Entry points: DFHZATA
Called by: DFHZACT
Description: The autoinstall program runs as the CATA transaction and performs operations necessary to INSTALL autoinstallable terminals. It requests information from a user program where appropriate.
DFHZATD

Entry points: DFHZATD
Called by: DFHZACT, DFHZNAC
Description: The autoinstall delete program runs as the CATD transaction and performs operations necessary to DELETE autoinstalled terminals. It requests information from a user program where appropriate.

DFHZATDX

Entry points: DFHZATDX
Called by: DFHZATA, DFHZATD
Description: DFHZATDX is the user program for autoinstall. It is called when:
- An autoinstall INSTALL is in progress
- An autoinstall DELETE has just completed
- An autoinstall INSTALL has failed.
For INSTALL, DFHZATDX selects a model name and the corresponding TRMIDNT to be used by the terminal control builder program (DFHTBSxx). This program can be used as a model for a user program.

DFHZATI

Entry points: DFHZATI1
Called by: DFHZACT
Description: The automatic task initiation module checks for stress conditions, calls DFHZZSIM if the node is not in session, acquires an RPL if necessary, and issues a conditional DFHKC TYPE=AVAIL macro. DFHZATI initiates bid protocols to decide whether the LU is available.

DFHZATMD

Entry points: DFHZATMD
Called by: DFHZATMF
Description: This program deletes all remote terminal definitions that are flagged (by DFHZATMF) for deletion.

DFHZATMF

Entry points: DFHZATMF
Called by: DFHZATMD
Description: This program flags remote terminals for Mass-deletion (by DFHZATMD). It is a part of the transaction routing component, and is started to flag all skeletons that have been unused for more than the terminal latency period for deletion.

DFHZATR

Entry points: DFHZATR
Called by: DFHZATR, DFHZXRE0
Description: The autoinstall restart program runs as the CATR transaction at CICS startup after the time period specified in the AIRDELAY parameter. DFHZATR scans all autoinstalled terminals, and causes the CATD transaction to be called to delete any autoinstalled terminals that have not been used during the AIRDELAY interval.

DFHZATS

Entry points: DFHZATS
Called by: DFHZTSP, DFHCRS
Description: The remote autoinstall program runs as the following four transactions:
CITS The remote autoinstall function that is attached by DFHZTSP.
CDTS The remote delete function that is attached by DFHZTSP or DFHCRS.
CFTS The remote reset function that flags terminals for mass deletion after a CICS restart and is attached by DFHZTSP or DFHCRS.
CMTS The mass delete function of remote terminals that is attached by DFHZATS transaction CFTS if it finds any terminals for deletion.

DFHZATT

Entry points: DFHZATT1
Called by: DFHZACT
Description: The task attach module checks for stress conditions, allocates an RPL if necessary, and determines the task to be attached either from the data, or from the TCTTE (if the previous transaction specified TRANID), or from the AID (for a 3270). DFHZATT also checks for paging commands (having been modified by DFHZAIT). Finally a conditional ATTACH is issued. The module is applicable for VTAM, SRL, and MVS console support.

DFHZBAN

Entry points: DFHZBAN
Called by: DFHZOPN
Description: The terminal control bind analysis program checks that a bind is valid and supportable and, if requested, sets the TCTTE information that supports the session parameters.
**DFHZBKCT**

**Entry points:** DFHZBKTI

**Called by:** DFHZSDL, DFHZSLX, DFHZRLX, DFHZLUS

**Description:** DFHZBKCT maintains the bracket state for LU6.2.

---

**DFHZBLX**

**Entry points:** DFHZBLX

**Called by:** DFHZSCX

**Description:** DFHZBLX is the part of SCIP exit which processes LU6.2 binds. It matches a TCTTE to the BIND and schedules DFHZOPN to complete the BIND process. This module returns to VTAM.

---

**DFHZCA**

**Entry points:** DFHZCANA

**Called by:** See component submodules

**Description:** DFHZCA is the name of the load module created when the following modules are link-edited together:

- DFHZACT - Activate scan
- DFHZFRE - FREEMAIN request
- DFHZGET - GETMAIN request
- DFHZQUE - Chaining
- DFHZRST - RESETR.

---

**DFHZCB**

**Entry points:** DFHZCBNA

**Called by:** See component submodules

**Description:** DFHZCB is the name of the load module created when the following modules are link-edited together:

- DFHZATI - Automatic task initiation
- DFHZDET - Task detach
- DFHZHPSR - HPO send/receive
- DFHZLRP - Logical record presentation
- DFHZRAC - Receive-any completion
- DFHZRAS - Receive-any slowdown processing
- DFHZRVS - Receive specific
- DFHZRVX - Receive specific exit
- DFHZSDR - Send response
- DFHZSDS - Send DFSYN
- DFHZSDX - Send DFSYN data exit
- DFHZSSX - Send DFSYN exit
- DFHZUIX - User input exit.

---

**DFHZCC**

**Entry points:** DFHZCCNA

**Called by:** See component submodules

**Description:** DFHZCC is the name of the load module created when the following modules are link-edited together:

- DFHZZERER - LU6.2 protocol error and exception handler
- DFHZZRL - LU6.2 application request logic
- DFHZZARM - LU6.2 migration logic
- DFHZZARR - LU6.2 application receive request logic
- DFHZZARRA - LU6.2 application receive buffer support
- DFHZZARRC - LU6.2 classify what next to receive
- DFHZZARRF - LU6.2 receive PHF7 and ERI
- DFHZZBKT - LU6.2 bracket state machine
- DFHZZCHS - LU6.2 chain state machine
- DFHZZCNT - LU6.2 contention state machine
- DFHZZCRT - LU6.2 RPL_B state machine
- DFHZZRLP - LU6.2 post-VTAM receive logic
- DFHZZRLX - LU6.2 receive exit program
- DFHZZRVL - LU6.2 pre-VTAM receive logic
- DFHZZSDL - LU6.2 send logic
- DFHZZSLX - LU6.2 send exit program
- DFHZZSTAP - MRO or LU6.2 conversation state determination
- DFHZZUSR - LU6.2 conversation state machine.

---

**DFHZCHS**

**Entry points:** DFHZCHS1

**Called by:** DFHZRLX, DFHZSDL, DFHZSLX

**Description:** DFHZCHS maintains the chain state for LU6.2.

---

**DFHZCLS**

**Entry points:** DFHZCLS1

**Called by:** DFHZACT

**Description:** The close destination module obtains an RPL if necessary, issues CLSDST to VTAM, and checks if it was accepted. The CLSDST exit handles the completion of the request. DFHZCLS performs a normal closedown procedure according to the LU type (for example, LU6 sends SBI and BIS). In the case of an abnormal closedown, DFHZCLS performs immediate termination, using CLSDST or TERMSESS commands. If the terminal was automatically defined, it is put out of service.

---

**DFHZCLX**

**Entry points:** DFHZCLX1

**Called by:** VTAM

**Description:** The close destination exit module receives control from VTAM when a CLSDST or TERMSESS request completes. If the CLSDST or TERMSESS was successful, DFHZCLX cleans up TCTTE and returns to VTAM; otherwise it enqueues the TCTTE to DFHZNAC and then returns to VTAM.
**DFHZCNA**

**Entry points:** DFHZCNA1

**Called by:** DFHZDSP

**Description:** The system console activity control program is responsible for CICS system requests. It performs the following functions:
- **Shutdown**—when all other access method terminals have been quiesced, quiesces console support, allowing CICS to terminate.
- **Resume**—resumes tasks waiting on read request when they are completed.
- **Detach**—releases all TIOAs associated with a completed task.
- **Attach**—passes the data associated with a MODIFY command (in a TIOA attached to a console TCTTE) to DFHZATT to create a task.
- **ATI**—determines whether a console TCTTE is available for automatic task initiation.

**DFHZCNR**

**Entry points:** DFHZCNR1

**Called by:** DFHZARQ

**Description:** The system console application request program performs READ, WRITE, and CONVERSE operations to an MVS system console that is used as a terminal.

**DFHZCNT**

**Entry points:** DFHZCNT1

**Called by:** DFHZLUS, DFHZRLX

**Description:** DFHZCNT maintains the contention state for LU6.2.

**DFHZCP**

**Entry points:** DFHZCPNA

**Called by:** See component submodules

**Description:** DFHZCP is the name of the load module created when the following modules are link-edited together:
- DFHZARQ - Application request handler
- DFHZATT - Attach routine
- DFHZCNA - System console activity control
- DFHZDSP - Dispatcher
- DFHZISP - Allocate/free-point routine
- DFHZSUP - Startup task
- DFHZUCT - 3270 uppercase translation.

**DFHZCQ**

**Entry points:** DFHZCQ

**Called by:** DFHAMTP, DFHCRS, DFHQRY, DFHTCRP, DFHWKP, DFHZATA, DFHZATD, DFHZTSP, DFHZXCU

**Description:** DFHZCQ is the control program for all requests for the dynamic add and delete of terminal control table entries. It is called by resource definition online (RDO) to:
- Cold start group lists
- Cold or warm start nonmigrated VTAM resources
- Dynamically install using the CEDA transaction.

The main subroutines of DFHZCQ are:
- DFHZCQCH - Catalog a TCT element
- DFHZCQDL - Delete
- DFHZCQIN - Initialize DFHZCQ
- DFHZCQIQ - Inquire about TCTTE
- DFHZCQIS - Install TCTTE
- DFHZCQIT - Add macro-generated TCTTE
- DFHZCQRS - Restore ZC resource.

**DFHZCQDL**

**Entry points:** DFHZCQDL

**Called by:** DFHZCQ00, DFHZNAC, RDO

**Description:** DFHZCQDL dynamically deletes a TCT entry when the entry is quiesced. This module is part of DFHZCQ.

**DFHZCQIN**

**Entry points:** DFHZCQIN

**Called by:** DFHTCRP

**Description:** DFHZCQIN initializes DFHZCQ for all its operations. This module is part of DFHZCQ.

**DFHZCQIQ**

**Entry points:** DFHZCQIQ

**Called by:** DFHZTSP

**Description:** DFHZCQIQ obtains the parameters for a TCT resource and is called by DFHZTSP in the terminal-owning node as part of the process of shipping a TCT definition to a remote system. This module is part of DFHZCQ.

**DFHZCQIS**

**Entry points:** DFHZCQIS

**Description:** DFHZCQIS installs a TCTTE. If the resource already exists, the old resource is deleted.
DFHZCQIT
Entry points: DFHZCQIT
Description: DFHZCQIT adds a macro-generated TCTTE to a CICS system.

DFHZCQRS
Entry points: DFHZCQRS
Description: During emergency restart or warm start, DFHTCRP restores terminal control resources to the state they were in before the last shutdown of CICS, using the restart data set.

DFHZCRQ
Entry points: DFHZCRQ1
Called by: TC CTYPE requests
Description: The CTYPE request module analyzes DFHTC CTYPE commands, and calls or links to the appropriate send module.

DFHZCRT
Entry points: DFHZCRT1
Called by: DFHZACT, DFHZARL, DFHZFRE, DFHZINAC, DFHZRAC, DFHZRLP, DFHZRVL, DFHZSDL, DFHZSHU, DFHZSTU, DFHZTPX
Description: DFHZCRT maintains the RPL_B state for LU6.2.

DFHZCUT
Entry points: DFHZCUT
Called by: DFHCSSC, DFHLUP, DFHSNAT, DFHTCPLR
Description: DFHZCUT manages the persistent verification signed-on-from list, also known as the local userid table (LUIT). There is one LUIT per connection supporting persistent verification.

DFHZCW
Entry points: DFHZCWNNA
Called by: See component submodules
Description: DFHZCW is the name of the load module created when the following modules are link-edited together:
DFHZERH - LU6.2 error program
DFHZEV1 - LU6.2 BIND security
DFHZEV2 - LU6.2 BIND security
DFHZLUS - LU6.2 session management program.

DFHZCX
Entry points: DFHZCWNAX
Called by: See component submodules
Description: DFHZCX is the name of the load module created when the following modules are link-edited together:
DFHZABD - Abend routine for incorrect requests
DFHZAND - Build TACB before issuing PC abends
DFHZCNR - System console application request
DFHZIS1 - ISC or IRC syncpoint
DFHZIS2 - IRC internal requests
DFHZLOC - Locate TCTTE and ATI requests
DFHZSTU - Terminal control status change.

DFHZCXR
Entry points: DFHZCXR
Called by: See component submodules
Description: DFHZCXR is the generic name allocated to a composite module that is not called by any other code. It includes the following transaction-routing related modules:
DFHZTSP - Terminal-sharing program
DFHZXRL - Routes LU6.2 commands to TOR
DFHZXRT - Receives LU6.2 commands from AOR.

DFHZCY
Entry points: DFHZCYN
Called by: See component submodules
Description: DFHZCY is the name of the load module created when the following modules are link-edited together:
DFHZASX - DFASY exit
DFHZDST - SNA-ASCII translation
DFHZLEX - LERAD exit
DFHZLGX - LOGON exit
DFHZLTX - LOSTERM exit
DFHZNSP - Network services exit
DFHZOPA - Open VTAM ACB
DFHZRRX - Release request exit
DFHZRSY - Resynchronization
DFHZSAX - Send synchronous command exit
DFHZSCX - SESSION control input exit
DFHZSDA - Send synchronous command
DFHZSES - SESSIONC
DFHZSEX - SESSIONC exit
DFHZSHU - Shutdown VTAM
DFHZSIM - SIMLOGON
DFHZSIX - SIMLOGON exit
DFHZSKR - Send response to command
DFHZSLS - Set logon start
DFHZSYN - Handle CTYPE=SYNC or CTYPE=RECOVER request
DFHZSYX - SYNAD exit
DFHZTPX - TPEND exit
DFHZTRA - Create ZCP or V10 trace requests
DFHZXRC - XRF session state data analysis.
DFHZCZ

**Entry points:** DFHZCZNA

**Called by:** See component submodules

**Description:** DFHZCZ is the name of the load module created when the following modules are link-edited together:

- DFHZCLS - CLSDST
- DFHZCLX - CLSDST exit
- DFHZCRO - Command request
- DFHZEMW - Error message writer
- DFHZOPN - OPNDST
- DFHZOPX - OPNDST exit
- DFHZRAQ - Read-ahead queuing
- DFHZRAR - Read-ahead retrieval
- DFHZTAX - Turnaround exit.

DFHZDET

**Entry points:** DFHZDET1

**Called by:** DFHZACT, DFHZISP

**Description:** The task detach module receives control when a detach request is issued by DFHZISP. If a WRITE is pending (deferred write or any write), the SEND routine is called. If the SEND cannot complete, the DETACH request is left on the activate queue. If requests are queued then DFHZACT drives DFHZDET when the operation is complete. If the node is in between bracket state, an end bracket is sent.

DFHZDSP

**Entry points:** DFHZDSP1

**Called by:** DFHSII1

**Description:** The dispatcher module handles the dispatching of modules for execution, and gives control to VTAM modules of ZCP using DFHZACT.

DFHZDST

**Entry points:** DFHZDST1

**Called by:** DFHZRVX, DFHZSDS

**Description:** The data stream translator module translates data between EBCDIC and ASCII code while that data is being sent and received on VTAM sessions.

DFHZEMW

**Entry points:** DFHZEMW1

**Called by:** DFHACP, DFHZDET, DFHZNAC, DFHZRAC

**Description:** The error message writer module handles all requests for error messages on VTAM supported terminals/LUs. According to the request flags, it:
- Sends a negative response
- Purges unprocessed inbound data until EOC or CANCEL is received
- Sends an error message.

DFHZERH

**Entry points:** DFHZERH1

**Called by:** DFHZARL, DFHZARRF

**Description:** DFHZERH handles the sending and receiving of LU6.2 FMH7s and negative responses. It also manages the logical receive buffer pointers TCTERBLA and TCTERBLL in a consistent manner with the physical receive buffer pointers TCTERBA and TCTERBDL, as (address, length) pairs.

DFHZEV1

**Entry points:** DFHZEV11

**Description:** DFHZEV1 is the LU6.2 bind-time security encryption validation program, part 1.

DFHZEV2

**Entry points:** DFHZEV21

**Description:** DFHZEV2 is the LU6.2 bind-time security encryption validation program, part 2.

DFHZFRE

**Entry points:** DFHZFRE1

**Called by:** DFHZACT, DFHZEMW, DFHZCLS, DFHZCLX

**Description:** The FREEMAIN module is used to free storage (RPLs, NIBs, bind areas, TIOAs, buffer lists, LUC send/receive buffers, and extract logon data) acquired by ZC modules. Some storage is also freed by other ZC modules.

DFHZGET

**Entry points:** DFHZGET1

**Called by:** DFHZACT, DFHZARL, DFHZATI, DFHZATT, DFHZCLS, DFHZISP, DFHZOPN, DFHZRAC, DFHZRST, DFHZRSY, DFHZRVL, DFHZRSV, DFHZSDA, DFHZSDL, DFHZSDR, DFHZSDS, DFHZSEs, DFHZSKR

**Description:** The GETMAIN module is used to acquire an RPL, NIB, bind area, TIOA, buffer list, or LUC send/receive buffer. DFHZGET also sets up the dynamic NIB using the information in the NIB descriptor block. Normally, when a ZC module requires some of the above storage, it invokes DFHZGET to obtain the storage; if this is unsuccessful, it may queue the request, and then DFHZACT calls DFHZGET on behalf of the caller.
**DFHZHPRX**

**Entry points:** DFHZHPNA

**Called by:** DFHKCSP (via DFHZHPSR and DFHKCP)

**Description:** In authorized path SRB mode, DFHZHPRX issues VTAM EXECRPL.

**DFHZHPSR**

**Entry points:** DFHZHPS1

**Called by:** DFHZRVS, DFHZSDS

**Description:** DFHZHPSR is the SEND and RECEIVE module for the HPO environment.

**DFHZISP**

**Entry points:** DFHZISP1

**Called by:** DFHISP, DFHKCP

**Description:** The intersystem program services ISC requests to free, or point to, a particular TCTTE within a specified system, or to allocate a TCTTE within a specified system. DFHZISP also handles ATI requests, and checks for a terminal time-out.

**DFHZIS1**

**Entry points:** DFHZIS11

**Description:** DFHZIS1 handles the transmissions control CTYPE requests of Prepare, Syncpoint Request (SPR), Commit, and Abort. Each request is translated into the appropriate ISC/IRC action and is transmitted to the connected system.

**DFHZIS2**

**Entry points:** DFHZIS21

**Called by:** DFHZARQ, DFHZIS1

**Description:** The intersystem program provides services for CICS system code that wants to use intersystem or interregion (IRC) function requests:

**RECEIVE**

- Is invoked when DFHCRNP gets input data as a result of a ‘switch first’ SVC request.

**IOR**

- The IRC input/output routine. This interfaces with the IRC SVC in order to send data to the other end of the connection, or await data from there.

**GETDATA**

- Is used to fetch input data into a TIOA.

**DISCONNECT**

- Disconnects a given IRC link.

**STOP**

- Quiesces interregion activity, either for connections to a given system, or for the whole of IRC.

**LOGOFF**

- Issues a logoff request to the IRC SVC. This completes IRC activity for this CICS system.

**OPERATIVE**

- Allows connections to be made to a given system.

**RECAIBRT**

- Processes input abend FMHs (FMH07).

**DFHZLEX**

**Entry points:** DFHZLEX1

**Called by:** VTAM

**Description:** The logical error address (LERAD) exit module receives control from VTAM when a logical error is detected. Logical errors are usually the result of an incorrectly defined terminal table.

**DFHZLGX**

**Entry points:** DFHZLGX1

**Called by:** VTAM

**Description:** The logon exit module receives control from VTAM when a terminal logs on to the network. DFHZLGX scans the CICS NIBs and, if a match is found, sets an OPNDST request in the corresponding TCTTE and places it on the activate queue. If no match is found, DFHZLGX defines a terminal automatically, if possible, by allocating an autodefine work element which holds the CINIT_RU. The work element is then queued for activate scan processing. Otherwise, a dummy TCTTE is placed on the NACP queue to write an error message.

**DFHZLOC**

**Entry points:** DFHZLOC1

**Called by:** DFHTC CTYPE=LOCATE

**Description:** The locate module provides two functions:

- Locates specific TCTTEs, TCTSEs, and SESSIONs in the TCT
- Locates LDC information.

**DFHZLRP**

**Entry points:** DFHZLRP1

**Called by:** DFHZARQ, DFHZSUP

**Description:** The logical record presentation module handles deblocking of input data. The delimiters that are recognized are new line (NL), interchange record separator (IRS), and transparent (TRN). One logical record is returned for each DFHTC TYPE=READ request.
**DFHZLTX**

**Entry points:** DFHZLTX1

**Called by:** VTAM

**Description:** The lost terminal (LOTERM) exit module receives control when VTAM detects a loss of contact with a node. There are three possible return codes set by VTAM on entry to this routine:

- **node lost, recovery in progress**
  The terminal is placed out of service with no further action taken.

- **node lost, recovery successful**
  The TCTTE is queued to the NACP queue with a ‘successful’ error code set; NACP issues a CLSDST; schedules a SIMLOGON, and issues an information message.

- **node lost, no recovery or unsuccessful recovery**
  The TCTTE is queued to the NACP queue with an ‘unsuccessful’ error code set; NACP issues a CLSDST and also the appropriate message.

**DFHZLUS**

**Entry points:** DFHZLUS1

**Description:** DFHZLUS handles session management for LU6.2 sessions.

**DFHZNAC**

**Entry points:** DFHZNANA

**Called by:** DFHZACT

**Description:** The node abnormal condition program is attached by DFHZACT when an error in communication with a logical unit occurs. DFHZNAC performs the following functions:

- Analyzes abnormal conditions
- Sends appropriate messages to the CSNE transient data destination
- Invokes the user-supplied (or sample) node error program
- Takes the appropriate actions resulting from the defaults which may have been modified by the node error program.

DFHZNAC consists of the following copybooks:

- DFHZNCA - Primary error action table and exits
- DFHZNCE - Take action routine
- DFHZNCS - Sense decode routine
- DFHZNCSV - VTAM return code routine.

**DFHZNEP**

**Entry points:** DFHZNENA

**Called by:** DFHZNAC

**Description:** The user-replaceable node error program provides:

- A general environment within which it is easy for users to add their own error processors
- Fundamental error recovery actions for a VTAM 3270 network
- The default NEP where the user selects a NEP at system initialization.

**DFHZNSP**

**Entry points:** DFHZNSP1

**Called by:** VTAM

**Description:** The network service program is invoked when VTAM detects a network service error; for example, when attempting to connect two nodes together, or when the link between two nodes is broken unexpectedly. This module receives control from the VTAM NSEXIT.

**DFHZOPA**

**Entry points:** DFHZOPA1

**Called by:** DFHEIQVT

**Description:** The open VTAM ACB module is invoked by DFHEIQVT when the master terminal command VTAM OPEN is issued. The ACB is opened and DFHZSLS is called to accept logon requests.

**DFHZOPN**

**Entry points:** DFHZOPN1

**Called by:** DFHZACT

**Description:** The open destination module acquires storage for an RPL and NIB and BIND areas if the TCTTE does not have these resources already, and sets up the BIND image if required. DFHZOPN then issues a VTAM OPNDST macro (or OPNSEC macro if secondary, to respond to an incoming BIND) to establish a session between CICS and the remote LU.

**DFHZOPX**

**Entry points:** DFHZOPX1

**Called by:** VTAM

**Description:** The open destination exit module receives control from VTAM on completion of the OPNDST macro in DFHZOPN. If the OPNDST was successful, it indicates in the TCTTE that SDT (start data transfer) is to be sent and checks whether a “good morning” message should be triggered. It then returns to VTAM.
DFHZQUE
Entry points: DFHZQUE1
Called by: All ZCP exits called by VTAM, DFHTCQUE macro
Description: The queue manipulation module processes all requests to add or remove a TCTTE to or from a ZCP activate queue. Additions to the activate queue made by mainline modules use compare-and-swap (CS), because an exit routine may also be adding to the queue asynchronously.

DFHZRAC
Entry points: DFHZRAC1
Called by: DFHZDSP
Description: The receive-any completion module processes the completion of receive-any requests, sets up the TIOA to be passed to attach, and reissues the RECEIVE_ANY macro.

DFHZRAQ
Entry points: DFHZRAQ1
Called by: DFHZARQ, DFHZSYN
Description: The read-ahead queuing module is used to save the inbound data stream in temporary storage when an interlock is caused by both the host and the terminal wanting to send data at the same time.

DFHZRAR
Entry points: DFHZRAR1
Called by: DFHZARQ
Description: The read-ahead retrieval module is called to retrieve data previously saved in temporary storage by DFHZRAQ.

DFHZRAS
Entry points: DFHZRAS1
Called by: DFHZRAC
Description: The receive-any slowdown processing module issues RECEIVE SPEC NQs on LU6.2 sessions for connections and modegroups for which there are ALLOCATE requests queued. This is only done on sessions considered most likely to lead to freeing a "flooding" situation that occurred when LU6.2 connections were reestablished after a failure.

DFHZRLG
Entry points: DFHZRLNA
Called by: DFHZACT
Description: The response logger program logs responses received for protected data sent to an APB. DFHZRLG processes TCTTEs on the log queue when attached by DFHZACT.

DFHZRLP
Entry points: DFHZRLP1
Called by: DFHZDSP
Description: DFHZRLP handles the completion of LU6.2 RECEIVE requests, using the receive RPL addressed by field TCTERPLB in the TCTTE LUC extension. It also manages the logical receive buffer pointers TCTERBLA and TCTERBLL in a consistent manner with the physical receive buffer pointers TCTERBA and TCTERBDL, as (address, length) pairs.

DFHZRLX
Entry points: DFHZRLX1
Called by: VTAM
Description: DFHZRLX is a VTAM exit routine that queues the completed RPL for (post-VTAM) processing by DFHZRLP.

DFHZRRX
Entry points: DFHZRRX1
Called by: VTAM
Description: The release request exit module receives control from VTAM when another application program has requested connection to a terminal currently connected to CICS. If the terminal is not busy, a CLSDST request is queued to the activate chain. Otherwise the release request indicator is set and the request is processed later by module DFHZDET.

DFHZRSP
Entry points: DFHZRSNA
Called by: DFHZACT, DFHZATI, DFHZCRQ, DFHZDET, DFHZEMW, DFHZERH, DFHZNAC, DFHZRAC, DFHZRSY, DFHZSTU
Description: The resynchronization send program performs 3614-dependent actions and is also used to retransmit committed output messages. The message is retrieved from temporary storage if necessary.

DFHZRST
Entry points: DFHZRST1
Called by: DFHZACT, DFHZATI, DFHZCRQ, DFHZDET, DFHZEMW, DFHZERH, DFHZNAC, DFHZRAC, DFHZRSY, DFHZSTU
Description: The RESETSR module changes the mode of a session with a terminal and cancels unsatisfied RECEIVE requests. The mode that is set can be Continue Any (CA) or Continue Specific (CS) and RTYPE=DFSYN, DFASY, or RESP.

---

**DFHZRSY**

Entry points: DFHZRSY1

Called by: DFHZACT

Description: The resynchronize module resynchronizes CICS and other nodes of the network. DFHZRSY checks whether inbound and outbound sequence numbers are valid.

---

**DFHZRVL**

Entry points: DFHZRVL1

Called by: DFHZARL, DFHZARRL

Description: DFHZRVL processes RECEIVE commands for LU6.2 sessions, using the receive RPL (RPL_B) addressed by field TCTERPLB in the TCTTE LUC extension. The processing state of the receive RPL is held in the RPL_B state machine field TCTERPBS, also in the TCTTE LUC extension.

---

**DFHZRVS**

Entry points: DFHZRVS1

Called by: DFHZACT

Description: The receive specific module initiates a DFSYN receive specific to obtain the next logical record from a node when a user application issues a RECEIVE command.

---

**DFHZRVX**

Entry points: DFHZRVX1

Called by: VTAM

Description: The receive specific exit module receives control from VTAM when a receive specific is completed. If the data received is too long for the TIOA provided, the overlength data flag is turned on in the TCTTE and the TCTTE is put back on the activate chain. Otherwise, the response is checked and marked in the TCTTE. The data length is set in the TIOA and the FMH is removed.

---

**DFHZSAX**

Entry points: DFHZSAX1

Called by: VTAM

Description: The send DFASY exit module receives control from VTAM when an asynchronous command has completed. It places the TCTTE on the NACP queue if recovery is needed.

---

**DFHZSCX**

Entry points: DFHZSCX1

Called by: VTAM

Description: The SCIP exit module is entered whenever the following asynchronous commands are received:
- Non-LU6.2 BIND (as secondary)
- UNBIND (as secondary)
- STSN (as secondary)
- Clear (as secondary)
- SDT (as secondary)
- Request recovery (as primary).

The module correlates BINDs to a TCTTE and schedules DFHZOPN to complete the BIND process. For the other commands, it takes appropriate action and then schedules DFHZNAC using the NACP queue. This module calls DFHZBLX to process LU6.2 binds.

---

**DFHZSDA**

Entry points: DFHZSDA1

Called by: DFHZACT, DFHZSDS

Description: The send data flow asynchronous module handles asynchronous command requests. It ensures that an rpl is allocated, primes the rpl for the requested command, and issues the VTAM asynchronous send macro.

---

**DFHZSDL**

Entry points: DFHZSDL1

Called by: DFHZARL

Description: DFHZSDL processes SEND commands for LU6.2 sessions, using the RPL addressed by field TCTERPLA in the TCTTE.

---

**DFHZSDR**

Entry points: DFHZSDR1

Called by: DFHZACT, DFHZCRQ, DFHZDET, DFHZRVS, DFHZSDA, DFHZSDS

Description: The send response module sends responses to nodes when a synchronization request for a terminal is made and a response is outstanding from a previous operation. If errors occur during task initiation, this module is responsible for the negative response.
**DFHZSDS**

**Entry points:** DFHZSDS1  
**Called by:** DFHZACT, DFHZARQ, DFHZATI, DFHZATT, DFHZDET  
**Description:** The send data synchronous module sets up and issues the appropriate VTAM send macro for requests of “send data” or an SNA synchronous command.

**DFHZSDX**

**Entry points:** DFHZSDX1  
**Called by:** VTAM  
**Description:** The send data synchronous exit module receives control from VTAM when a SEND request is complete. It checks the RPL for successful completion of the message sent and takes appropriate action.

**DFHZSES**

**Entry points:** DFHZSES1  
**Called by:** DFHZACT, DFHZRSY  
**Description:** The session control module is entered whenever a session control command is requested by CICS. It sets up and issues the VTAM SESSIONC command.

**DFHZSEX**

**Entry points:** DFHZSEX1  
**Called by:** VTAM  
**Description:** The SESSIONC exit module receives control from VTAM when a SESSIONC command has completed. If the command was successful, it turns off the corresponding flags and enqueues the TCTTE on the activate chain. If the completion was not successful, the TCTTE is placed on the NACP queue for recovery processing.

**DFHZSHU**

**Entry points:** DFHZSHU1  
**Called by:** DFHZDSP  
**Description:** The close VTAM ACB module is invoked whenever CICS and VTAM are being uncoupled. This may be as a result of DFHZTPX being driven as the result of a VTAM halt command or the issue of the master terminal command SET VTAM,CLOSE | IMMCLOSE. The status of all sessions is checked and, when all are inactive, the ACB is closed.

**DFHZSIM**

**Entry points:** DFHZSIM1  
**Called by:** DFHZACT  
**Description:** The simulate logon module is entered to issue a VTAM SIMLOGON or REQSESS (if secondary) request to place a node in session without the operator having to logon. LU6.2 can be selected by mode name.

**DFHZSIX**

**Entry points:** DFHZSIX1  
**Called by:** VTAM  
**Description:** Whenever a SIMLOGON or REQSESS command has been completed, this exit routine is scheduled by VTAM. On successful completion, it turns off the SIMLOGON requested flag and enqueues the TCTTE or TCTME on the activate chain or, if NACP is required, for NACP processing.

**DFHZSKR**

**Entry points:** DFHZSKR1  
**Called by:** DFHZACT  
**Description:** The send command response module sends responses to VTAM commands including response to BIND, STSN, and SDT. A positive or negative response can be sent. The module is for secondary LU support only.

**DFHZSLS**

**Entry points:** DFHZSLS1  
**Called by:** DFHZDSP, DFHZOPA  
**Description:** The SETLOGON start module issues SETLOGON to cause VTAM to accept automatic logon requests, and issues the initial RECEIVE ANYs for RPLs in the receive-any pool. DFHZSLS also examines the SIT to determine whether autodefine is used. If it is, the appropriate system initialization parameters are copied to the TCT prefix.

**DFHZSLX**

**Entry points:** DFHZSLX1  
**Called by:** VTAM  
**Description:** DFHZSLX is a VTAM exit routine that handles the completion of LU6.2 SEND requests.

**DFHZSSX**

**Entry points:** DFHZSSX1  
**Called by:** VTAM  
**Description:** The send data flow synchronous exit
module receives control when the send of a DFSYN command has been completed.

**DFHZSTAP**

**Entry points:** DFHZSTA1  
**Called by:** DFHEGL, DFHETC, DFHETL  
**Description:** DFHZSTAP determines the state of an MRO or LU6.2 conversation from an application viewpoint.

**DFHZSTU**

**Entry points:** DFHZSTU1  
**Called by:** DFHTC CTYPE=STATUS, DFHEIQMT, DFHEIQSC, DFHEIQST  
**Description:** DFHZSTU changes the status of TCTTEs and TCTSEs. It can change the following statuses:  
- Inservice  
- Outservice  
- Intlog | No intlog  
- Page | Autopage  
- ATI | NATI.

**DFHZSUP**

**Entry points:** DFHZSUP1  
**Called by:** DFHKCP  
**Description:** The startup task module is the entry point for all terminal-related tasks. DFHZSUP performs the following functions:  
- Sets up the TCTTE status  
- Performs security checking  
- Performs logging of the TCTTE status and input TIOA  
- Performs PCT option checking  
- Passes control to transaction program, for example, user application, DFHACP, DFHAPRT.

**DFHZSYN**

**Entry points:** DFHZSYN1  
**Called by:** DFHDBP  
**Description:** DFHZSYN handles CTYPE=SYNC and RECOVER requests. For protected message support, DFHSPP issues CTYPE=SYNC to clear protected messages. For RECOVER requests, DFHZSYN ensures that no further I/O is issued to that session, and that UNBIND flows.

**DFHZSYX**

**Entry points:** DFHZSYX1  
**Called by:** VTAM  
**Description:** The SYNAD exit module receives control from VTAM when a catastrophic error is encountered. DFHZSYX determines the type of error and the appropriate action to be taken, and schedules NACP using the NACP queue to complete the recovery processing.

**DFHZTAX**

**Entry points:** DFHZTAX1  
**Called by:** VTAM  
**Description:** The turnaround exit module is called by VTAM on completion of the SEND operation initiated by DFHZRVS in order to perform a turnaround in flip-flop protocol.

**DFHZTPX**

**Entry points:** DFHZTPX1  
**Called by:** VTAM  
**Description:** The TPEND exit module receives control when VTAM is terminating. It schedules a CLSDST for each active session if quick shutdown is required, and sets bits in the TCT prefix so that DFHZSHU is invoked.

**DFHZTRA**

**Entry points:** DFHZTRA1  
**Called by:** DFHZACT, DFHZDET, DFHZRAC, DFHZRLP, DFHZRVS, DFHZSDL, DFHZSDR, DFHZSDS  
**Description:** DFHZTRA creates VIO trace entries.

**DFHZTSP**

**Entry points:** DFHZTSP1  
**Called by:** DFHAPRT, DFHISP, DFHRTE, DFHTPS, DFHZARQ, DFHZCQ, DFHZSUP  
**Description:** The terminal sharing program acquires a TCTTE for a link to a remote CICS address space, and transfers request data to that space. DFHZTSP also receives requests from the remote address space.

**DFHZUCT**

**Entry points:** DFHZUCT1  
**Called by:** DFHAPRT, DFHZARQ, DFHZCNA, DFHZRAC, DFHZRVS, DFHZSUP  
**Description:** The uppercase translate module converts a VTAM 3270 data stream into uppercase.
DFHZUIX
Entry points: DFHZUIX1
Called by: DFHZACT, DFHZRAC, DFHZRVX
Description: The user input exit module is called
directly (by DFHZRAC) or indirectly (by DFHZRVX via
DFHZACT) to link to the user’s XZCIN exit.

DFHZUSR
Entry points: DFHZUSR1
Called by: DFHACP, DFHETL, DFHZARER,
DFHZARL, DFHZARM, DFHZARR, DFHZARRE,
DFHZERH, DFHZOPX, DFHZSTAP, DFHZSUP,
DFHZUSR, DFHZXRL, DFHZXRT
Description: DFHZUSR maintains the conversation
state for LU6.2.

DFHZXCU
Entry points: DFHZXCU
Description: The VTAM XRF catch-up program is
used to send messages that allow a new alternate
system to catch up with the current state of the active
system for:
• TCT contents
• Bound/unbound state of sessions.

The program is invoked when a new alternate system
signs on.

DFHZXQO
Entry points: DFHZXQO
Called by: DFHTCRP, DFHZXST
Description: The XRF ZCP tracking queue organizer
allows pending XRF tracking activity to be stored in a
way that honors interdependencies, while allowing such
requests to be met as soon as all their
prerequisites are fulfilled. This component consists of a
data structure and accessing program that uses the
CICS catalog key structure to identify all the actions for
a single resource and the dependencies between them.
Actions are put into the structure on receipt in
DFHTCRP, and removed by DFHTCRP and at the end of
DFHZNAC processing for standby BIND and
CLSDST completion. The structure is freed at the end of
DFHTCRP tracking.

DFHZXRC
Entry points: DFHZXRC1
Called by: DFHZACT
Description: DFHZXRC analyzes the data received in
response to the SESSIONC CONTROL=SWITCH
command. It determines the state of the session at the
point when it was switched, and initiates the necessary
action to clean up and recover the session.

DFHZXRE0
Entry points: DFHZXRE0
Called by: System
Description: DFHZXRE0 runs the CXRE transaction to
perform autoconnect and XRF reconnect processing. It
also starts the acquire process for terminals with flag
TCTEXRE set.

DFHZXRL
Entry points: DFHZXRL1
Called by: DFHZARL, DFHZISP
Description: DFHZXRL is executed in an
application-owning region. It routes LU6.2 commands
to the terminal-owning region.

DFHZXRT
Entry points: DFHZXRT1
Called by: DFHZTSP
Description: DFHZXRT executes in a terminal-owning
region. It receives LU6.2 commands from the
application-owning region, and issues them to an APPC
device.

DFHZXST
Entry points: DFHZXST
Called by: DFHETC, DFHSIJ1, DFHTCRP,
DFHTCRPS, DFHZNAC, DFHZOPA, DFHZXCU
Description: XRF ZCP session-state tracking is called by:
• DFHZNAC for BIND/UNBIND completion in the
active system, and for standby-BIND and UNBIND
in the alternate system
• DFHETC for logon data freed in the active system
• DFHTCRPS to handle a tracking message
• DFHTCRP to terminate session tracking
• DFHZXCU for BIND/UNBIND catch-up in the
active system
• DFHSIJ1 and DFHZOPA to issue a SETLOGON
START command.
Part 4. Appendixes
### Bibliography

#### CICS Transaction Server for z/OS

- CICS Transaction Server for z/OS Migration Guide: GC34-5699
- CICS Transaction Server for z/OS Installation Guide: GC34-5697
- CICS Transaction Server for z/OS Program Directory: GI10-2525
- CICS Transaction Server for z/OS Licensed Program Specification: GC34-5698

The above titles are the only books available in hardcopy for CICS Transaction Server for z/OS Version 2 Release 1. All the remaining CICS and CICSPlex SM books are supplied in softcopy only in the CICS Information Center, which is distributed on CD-ROM.

#### CICS books for CICS Transaction Server for z/OS

<table>
<thead>
<tr>
<th>Category</th>
<th>Title</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>General</td>
<td>CICS User’s Handbook</td>
<td>SX33-6116</td>
</tr>
<tr>
<td></td>
<td>CICS Transaction Server for z/OS Glossary (softcopy only)</td>
<td>GC34-5696</td>
</tr>
<tr>
<td>Administration</td>
<td>CICS System Definition Guide</td>
<td>SC34-5725</td>
</tr>
<tr>
<td></td>
<td>CICS Customization Guide</td>
<td>SC34-5706</td>
</tr>
<tr>
<td></td>
<td>CICS Resource Definition Guide</td>
<td>SC34-5722</td>
</tr>
<tr>
<td></td>
<td>CICS Operations and Utilities Guide</td>
<td>SC34-5717</td>
</tr>
<tr>
<td></td>
<td>CICS Supplied Transactions</td>
<td>SC34-5724</td>
</tr>
<tr>
<td>Programming</td>
<td>CICS Application Programming Guide</td>
<td>SC34-5702</td>
</tr>
<tr>
<td></td>
<td>CICS Application Programming Reference</td>
<td>SC34-5703</td>
</tr>
<tr>
<td></td>
<td>CICS System Programming Reference</td>
<td>SC34-5726</td>
</tr>
<tr>
<td></td>
<td>CICS Front End Programming Interface User’s Guide</td>
<td>SC34-5710</td>
</tr>
<tr>
<td></td>
<td>CICS C++ OO Class Libraries</td>
<td>SC34-5705</td>
</tr>
<tr>
<td></td>
<td>CICS Distributed Transaction Programming Guide</td>
<td>SC34-5708</td>
</tr>
<tr>
<td></td>
<td>CICS Business Transaction Services</td>
<td>SC34-5704</td>
</tr>
<tr>
<td></td>
<td>Java in CICS</td>
<td>SC34-5881</td>
</tr>
<tr>
<td>Diagnosis</td>
<td>CICS Problem Determination Guide</td>
<td>GC33-5719</td>
</tr>
<tr>
<td></td>
<td>CICS Messages and Codes</td>
<td>GC33-5716</td>
</tr>
<tr>
<td></td>
<td>CICS Diagnosis Reference</td>
<td>LY33-6097</td>
</tr>
<tr>
<td></td>
<td>CICS Data Areas</td>
<td>LY33-6096</td>
</tr>
<tr>
<td></td>
<td>CICS Trace Entries</td>
<td>SC34-5727</td>
</tr>
<tr>
<td></td>
<td>CICS Supplementary Data Areas</td>
<td>LY33-6098</td>
</tr>
<tr>
<td>Communication</td>
<td>CICS Intercommunication Guide</td>
<td>SC34-5712</td>
</tr>
<tr>
<td></td>
<td>CICS Family: Interproduct Communication</td>
<td>SC34-0824</td>
</tr>
<tr>
<td></td>
<td>CICS Family: Communicating from CICS on System/390</td>
<td>SC34-1697</td>
</tr>
<tr>
<td></td>
<td>CICS External Interfaces Guide</td>
<td>SC34-5709</td>
</tr>
<tr>
<td></td>
<td>CICS Internet Guide</td>
<td>SC34-5713</td>
</tr>
<tr>
<td>Special topics</td>
<td>CICS Recovery and Restart Guide</td>
<td>SC34-5721</td>
</tr>
</tbody>
</table>

© Copyright IBM Corp. 1989, 2001
CICS Performance Guide
CICS IMS Database Control Guide
CICS RACF Security Guide
CICS Shared Data Tables Guide
CICS Transaction Affinities Utility Guide
CICS DB2 Guide

CICSPlex SM books for CICS Transaction Server for z/OS

General
CICSPlex SM Master Index
CICSPlex SM Concepts and Planning
CICSPlex SM User Interface Guide
CICSPlex SM View Commands Reference Summary

Administration and Management
CICSPlex SM Administration
CICSPlex SM Operations Views Reference
CICSPlex SM Monitor Views Reference
CICSPlex SM Managing Workloads
CICSPlex SM Managing Resource Usage
CICSPlex SM Managing Business Applications

Programming
CICSPlex SM Application Programming Guide
CICSPlex SM Application Programming Reference

Diagnosis
CICSPlex SM Resource Tables Reference
CICSPlex SM Messages and Codes
CICSPlex SM Problem Determination

Other CICS books
Designing and Programming CICS Applications
CICS Application Migration Aid Guide
CICS Family: API Structure
CICS Family: Client/Server Programming
CICS Transaction Gateway for OS/390 Administration
CICS Family: General Information
CICS 4.1 Sample Applications Guide
CICS/ESA 3.3 XRF Guide

Note: The CICS Transaction Server for OS/390: Planning for Installation book that was part of the library for CICS Transaction Server for OS/390, Version 1 Release 3, is now merged with the CICS Transaction Server for z/OS Installation Guide. If you have any questions about the CICS Transaction Server for z/OS library, see CICS Transaction Server for z/OS Installation Guide, which discusses both hardcopy and softcopy books and the ways that the books can be ordered.
Determining if a publication is current

IBM regularly updates its publications with new and changed information. When first published, both hardcopy and BookManager softcopy versions of a publication are usually in step. However, due to the time required to print and distribute hardcopy books, the BookManager version is more likely to have had last-minute changes made to it before publication.

Subsequent updates will probably be available in softcopy before they are available in hardcopy. This means that at any time from the availability of a release, softcopy versions should be regarded as the most up-to-date.

For CICS Transaction Server books, these softcopy updates appear regularly on the Transaction Processing and Data Collection Kit CD-ROM, SK2T-0730-xx. Each reissue of the collection kit is indicated by an updated order number suffix (the -xx part). For example, collection kit SK2T-0730-06 is more up-to-date than SK2T-0730-05. The collection kit is also clearly dated on the cover.

Updates to the softcopy are clearly marked by revision codes (usually a “#” character) to the left of the changes.
Index

Special Characters
“good morning” message program 691

Numerics
62XM gate
    BIND_XM_CLIENT function 57
    INIT_XM_CLIENT function 56

A
ABAB gate
    CREATE_ABEND_RECORD function 23
    INQUIRE_ABEND_RECORD function 28
    START_ABEND function 27
    TAKE_TRANSACTION_DUMP function 29
    UPDATE_ABEND_RECORD function 25
    ABEND_TRANSACTION function, XMER gate 1277
abnormal termination
    system recovery program (SRP) 1139
    transaction failure program (TFP) 1257
    ABNORMALLY_TERMINATE_TASK function, KEDS gate 741
ACB (access control block) 668
    access method control block, VSAM 553
    access method control block, VTAM 1211
    access method control block (ACB) 668
    access method control block (ACB), VSAM 553
    access method control block (ACB), VTAM 1211
    access methods, terminal control 1201
ACP (abnormal condition program) 1257
    node 849
    ACQUIRE_ACTIVITY function, BAAC gate 218
    ACQUIRE_PROCESS function, BAPR gate 211
    ACQUIRE_PROGRAM function, LDLD gate 758
    ACTION_CORBASERVER function, EJCG gate 461
    ACTION_DJAR function, EJDG gate 471
    ACTIVATE_MODE function, DSTIT gate 313
    ACTIVATE_OBJECT function, EJOS gate 490
    activate scan (DFHZACT) 114
    ACTIVATE_TRAP function, TRSR gate 1245
    adapter, FEPI 657 (continued)
    logic flow 659
    ADD_ACTIVITY function, BAAC gate 213
    ADD_BEAN function, EJBG gate 454
    ADD_BEAN function, EJMI gate 484
    ADD_CORBASERVER function, EJCG gate 461
    ADD_DJAR function, EJDG gate 472
    ADD_DOMAIN function, DMDM gate 371
    ADD_DOMAIN function, KEDD gate 734
    ADD_ENTRY function, EJDI gate 477
    ADD_FUNCTION function, CCCC gate 250
    ADD_GATE function, KEDD gate 736
    ADD_LINK function, RMLN gate 929
    ADD_LOCK function, LMLM gate 773
    ADD_LOGICAL_SERVER function, IILS gate 693
    ADD_METHOD function, EJMI gate 484
    ADD_PENDING_REQUEST function, SHPR gate 1007
    ADD_PROCESS function, BAPR gate 209
    ADD_REALTACH_ACQUIRED function, BAAC gate 219
    ADD_REPLACE_TERM_MODEL, AITM format 130
    ADD_REPLACE_DCTEMPLATE function, DHTM gate 360
    ADD_REPLACE_PARTNER function, PRPT format 872
    ADD_REPLACE_PROCESSTYPE function, BATT gate 204
    ADD_REPLACE_QMODEL function, IIMM gate 695
    ADD_REPLACE_TCLASS function, XMCL gate 1267
    ADD_REPLACE_TCIPSERVICE function, SOAD gate 1070
    ADD_REPLACE_TDQUEUE function, TDTM gate 64
    ADD_REPLACE_TRANDEF function, XMXD gate 1292
    ADD_SUBEVENT function, EEMM gate 497
    ADD_SUBORDINATE function, OTSU gate 860
    ADD_SUBPOOL function, SMAD gate 1098
    ADD_SUSPEND function, DSSR gate 317
    ADD_SYMBOL_LIST function, DHTM gate 359
    ADD_SYSTEM_DUMPCODE function, DUDT gate 390
    ADD_TCB function, DSTIT gate 314
    ADD_TCLASS function, XMCL gate 1267
    ADD_TIMER_REQUEST function, BAAC gate 218
    ADD_TRAN_DUMPCODE function, DUDT gate 382
    ADD_TRANSACTION_SECURITY function, XSXM gate 1043
    ADD_TRANSACTION_USER function, USXM gate 1351
    ADD_USER_WITH_PASSWORD function, USAD gate 1342
    ADDRESS_DATA function, BAGD format 230
    address space modules 717
    MVS cross-memory program (DFHXMP) 717
    ADFHAPD1 distribution library 1420
    ADFHC370 distribution library 1420
    ADFHCLIB distribution library 1420
    ADFHCOB distribution library 1420
    ADFHENV distribution library 1420
    ADFHINST distribution library 1420
    ADFHLANG distribution library 1420
    ADFHMAC distribution library 1420
    ADFHMOD distribution library 1420
    COBOL elements 1420
    ADFHMSCS distribution library 1420
    ADFHMSRC distribution library 1420
    ADFHPARM distribution library 1420
    ADFHPL1 distribution library 1420
    ADFHPLI distribution library 1420
    ADFHPLIB distribution library 1420
    ADFHPROC distribution library 1420
    ADFHSAMP distribution library 1420
    ADFHAPD2 distribution elements 1420
    C/370 elements 1420
    COBOL elements 1420
    PL/I elements 1420
    AFCB (authorized function control block) 399
    AID (automatic initiate descriptor)
        AP domain termination program (STP) 112
        chain 112
    AILN format
        COMPLETE_INIT function 127
        START_INIT function 127
    ALIQ format
        END_BROWSE function 130
        GET_NEXT function 129
        INQUIRE_TERM_MODEL function 129

© Copyright IBM Corp. 1989, 2001
CICS Transaction Server: CICS Diagnosis Reference
B

BAAC gate
ACQUIRE_ACTIVITY function 218
ADD_ACTIVITY function 213
ADD_REATTACH_ACQUIRED function 219
ADD_TIMER_REQUEST function 218
CANCEL_ACTIVITY function 215
CHECK_ACTIVITY function 216
DELETE_ACTIVITY function 217
LINK_ACTIVITY function 214
RESET_ACTIVITY function 217
RESUME_ACTIVITY function 215
RETURN_END_ACTIVITY function 217
RUN_ACTIVITY function 214
SUSPEND_ACTIVITY function 215

BABR gate
COMMIT_BROWSE function 226
ENDBR_ACTIVITY function 220
ENDBR_CONTAINER function 223
ENDBR_PROCESS function 225
GETNEXT_ACTIVITY function 220
GETNEXT_CONTAINER function 222
GETNEXT_PROCESS function 224
INQUIRE_ACTIVATION function 226
INQUIRE_ACTIVITY function 220
INQUIRE_CONTAINER function 223
INQUIRE_PROCESS function 225
STARTBR_ACTIVITY function 219
STARTBR_CONTAINER function 222
STARTBR_PROCESS function 224
backout logging 541
BACKOUT_UOW function, RMUW gate 928
backup while open (BWO) 566, 584, 594

BACR gate
DELETE_CONTAINER function 226
GET_CONTAINER INTO function 227
GET_CONTAINER_SET function 228
PUT_CONTAINER function 228
BAD format
ADDRESS_DATA function 230
DEstroy_TOKEN function 229
INQUIRE_DATA_LENGTH function 229
RELEASE_DATA function 230

BAM (business application manager) domain 203

BAPR gate
ACQUIRE_PROCESS function 211
ADD_PROCESS function 209
CANCEL_PROCESS function 211
CHECK_PROCESS function 212
LINK_PROCESS function 210
RESET_PROCESS function 213
RESUME_PROCESS function 212
RUN_PROCESS function 209
SUSPEND_PROCESS function 211
basic direct access method (BDAM) 539
BATT gate
ADD_REPLACE_PROCESSTYPE function 204
COMMIT_PROCESSTYPE_TABLE function 208
DISCARD_PROCESSTYPE function 207
END_BROWSE_PROCESSTYPE function 206
GET_NEXT_PROCESSTYPE function 206
INQUIRE_PROCESSTYPE function 205
SET_PROCESSTYPE function 207
START_BROWSE_PROCESSTYPE function 206
BAXM gate
BIND.getActivity_REQUEST function 208
INIT_ACTIVITY_REQUEST function 208
BDAM
ENDBR request processing 551
READ request processing 545
READNEXT request processing 550
RESETBR request processing 550
REWRITE request processing 548
STARTBR request processing 550
UNLOCK request processing 548
WRITE request processing 547
BDAM (basic direct access method) 539
BDAM request processor, file control (DFHFCBD) 567
BEGIN_TRAN function, OTTR gate 858
BIND.getActivity_REQUEST function, BAXM gate 208
BIND_FACILITY function, ICXM gate 54
BIND_FACILITY function, TDXM gate 73
BIND_FACILITY function, TFDM gate 96
BIND_SECONDARY function, TDXM gate 73
BIND_UOW_TO_TXN function, RMUW gate 929
BIND_XM_CLIENT function, 62XM gate 57
BIND_XM_CLIENT function, MRXM gate 56
BIND_XM_CLIENT function, TFXM gate 55
BMS (basic mapping support) 133
(partition handling program (PHP) 153
route list resolution program (RLR) 156
standard version, modules used 140
terminal page processor (TPP) 158
terminal page retrieval program (TPR) 162
terminal page scheduling program (TPS) 165
undelivered messages cleanup program (TPQ) 160
BPS (builder parameter set) 167
BRAT gate
ATTACH function 52
BRIO gate
INQUIR_CONTEXT function 53
BROWSE_ALL_GET_NEXT function, LGBA gate 780
BROWSE_CHAINS_GET_NEXT function, LGCC gate 784
browse token, table manager 1151
browse token table (BTT) 414
BSAM (basic sequential access method) 1195
and testing facility 1198
BTT (browse token table) 414
build/delete terminals 197
builder parameter list 196
builder parameter set (BPS) 167
builders 167
description 167
purpose 178
builders for 3277 remote terminal calling sequence 195
built-in functions
description 201
field edit 201
phonetic conversion 201
business application manager (BAM) domain 203
BWO (backup while open) 566, 584, 594

C

CALL macro
DL/I interface 344
CALLLDLI macro
DL/I interface 344
calling sequence builders for 3277 remote terminal 195
CANCEL_ACTIVITY function, BAAC gate 215
CANCEL_AID function, TFAL gate 75
CANCEL_AIDS_FOR_CONNECTION function, TFAL gate 76
CANCEL_AIDS_FOR_TERMINAL function, TFAL gate 77
CANCEL function, APTC gate 16, 43
CANCEL function, TSR gate 1227
CANCEL_PROCESS function, BAPR gate 211
CANCEL_TASK function, DSAT gate 306

Index 1589
CHECK_NON_CICS_RESOURCE function, XSRC gate 1041
CHECK_PREFIX function, TSBR gate 1187
CHECK_PROCESS function, BAPR gate 212
CHECK_STORAGE function, SMCK gate 1100
CHECK_SURROGATE_USER function, XSRC gate 1040
CHECK_TIMER function, EMEM gate 498
CHECK_TRANID_IN_USE function, TFDL gate 78
checkpoint and restart 1196
CIB (command input buffer) 1207
CICS-AD/Cycle Language
Environment/370 interface 235
establishing connection 237
function calls 235
parameter lists 240
storage acquisition 238
storage for transaction 237
CICS business logic interface 1413
CICS-C/370 interface 257
CICSTSAD interface 1413
CICS-DB2 DB2TRAN block (D2TRN) 272
CICS-DB2 DB2Global block (D2GLB) 271
CICS-DB2 global work area (D2GWA) 272
CICS-DB2 life of task block (D2LOT) 272
CICS-DB2 static storage (D2SS) 271
CICS-DB2 support 265
CICS-DBCTL interface 291
CICS Web Interface 1413
class of service, LU6.2 1119
CLEAR_LABELS function, PGHM gate 894
CLEAR_MATCH function, DSAT gate 305
CLEAR_PENDING function, RMNM gate 944
CLOSE_ALL_EXTRA_TD_QUEUES function, TDOQ gate 63
close destination program, DFHCLSL 118
CLOSE function, APTC gate 15, 43, 44
CLOSE function, LGGL gate 789
CLOSE function, SOCK gate 1062
CLOSE_OBJECT_STORE function, EJOS gate 491
CLOSE_TRANSIENT_DATA function, TDOQ gate 63
CLT (command list table) 531
CMAC transaction 828
CMF transaction 682
CMS transaction 833, 834
cold start 197
COLLECT_RESOURCE_STATS function, STST format 1088
COLLECT_STATISTICS function, STST format 1088
command input buffer (CIB) 1207
command-language translator 917
command list table (CLT) 531
COMMIT function, OTTR gate 859
COMMIT_ONE_PHASE function, OTTR gate 859
commit process
single-phase 1164
two-phase 1163
COMMIT_PROCESS_TYPE_TABLE function, BATT gate 208
COMMIT_RQMODELS function, IIMM gate 496
COMMIT_SURROGATE function, RTSU gate 57
COMMIT_TDQDEFS function, TDTM gate 73
COMMIT_UOW function, RMUW gate 924
common programming interface
(CPI) 1001
COMMUNICATE function, BABB gate 226
communication with remote system 675
Communications interface, SAA 1001
COMPLETE_INIT function, AIIN format 127
COMPLETE_INIT function, CPIN format 1005
COMPLETE_INIT function, PRIN format 872
components of CICS
organization 5
concurrency control 539
concurrent tasks 542
CONFIRM_ALL_BEANS function, EJBG gate 455
CONNECT for IRC 717
CONNECT function, LGBL gate 799
CONNECT function, LGST gate 809
collection control block (CCB) 713
collection control element (CCE) 1207
collection message handling 1119
collection session update model 540
collection blocks
for BMS, illustrated 138
for file control, illustrated 552
for interregion communication (IRC), illustrated 709
for relay transaction, illustrated 1331
for subsystem interface, illustrated 1122
for table manager, illustrated 1152
for terminal control, illustrated 1216
for user exit interface, illustrated 1362
conversation
session recovery 718
CONVERSE function, MEME gate 822
CONVERT_TO_DECIMAL_TIME function, KETI gate 751
CONVERT_TO_STCK_FORMAT function, KETI gate 751
COUNT_FOR_CS function, EJDG gate 473
coupling facility data table 540
coupling facility data tables server 542
CP (common programming interface) 1001
CPIN format
CPMI transaction 674
CPIN format
CPI (common programming interface) 1001
count function 185
creation/deletion state machine 185
CREATE_UOW function, RMUW
CREATE_TCB function, KEDS gate 742
CREATE_TASK function, KEDS gate 741
CREATE_PASSTICKET function, XSPW
CREATE function, RZSO gate 979
CREATE_DOCUMENT function, DHDH
CREATE_DIRECTORY function, DDDI
CREATE_CHAIN_TOKEN function
CREATE_ABEND_RECORD function
create, EXEC CICS 198
CRB (cross-region block) 709
CPSP format
CPMI transaction 674
CSZI transaction 657
CSPS transaction 146
CSPQ transaction 146, 161
CSPS transaction 146
CS2I transaction 657
CVMI transaction 674
cross-system coupling facility (XCF) 709
creation/deletion state machine 709
cross-region block (CRB) 709
cross-system coupling facility (XCF) used for interregion communication 709
cross-system dumping (cont.) 1324
deblocking (cont.)
READ request processing 545
RESETBR request 550
STARTBR request 550
deblocking for BDAM data sets 539
DEBREC option
READ request processing 545
READDNEXT request processing 550
RESETBR request 550
STARTBR request 550
DECB (data event control block) 1202
defined work element (DWE) 1132
DEFINE_ATOMIC_EVENT function, EMEM gate 498
DEFINE_COMPOSITE_EVENT function, EMEM gate 499
DEFINE_PROGRAM function, PGDD gate 760
DEFINE_PROGRAM function, PGDD gate 883
DEFINE_TIMER function, EMEM gate 499
del_transaction_security function, XSXM gate 1043
DELETE_ACTIVITY function, BAAC gate 217
DELETE_ALL_BEANS function, EJBG gate 456
DELETE_ALL_DJARS function, EJDG gate 475
DELETE_ALL function, LGCC gate 785
DELETE_ALL_OPEN_TCBs function, DSIT gate 316
DELETE_BEAN function, EJBG gate 456
DELETE_BOOKMARK function, DHDD gate 356
DELETE_CERTIFICATE_DATA function, SOIS gate 1070
DELETE_CERTIFICATE_DATA function, SOSE gate 1079
DELETE_CONTAINER function, BACR gate 226
DELETE_CORBASEVER function, EJCG gate 463
DELETE_DATA function, DHDD gate 355
DELETE_DJAR function, EJDG gate 474
DELETE_DOCTEMPLATE function, DHDD gate 354
DELETE_ENTRY function, DDI gate 296
DELETE_EVENT function, EMEM gate 500
DELETE function, CCC gate 250
DELETE function, TSQR gate 1173
DELETE function, TSSH gate 1182
DELETE_GATE function, KEDD gate 736
DELETE_HISTORY function, LGCC gate 786
DELETE_LINK function, RMLM gate 931
delete lock 549
DELETE_LOCK function, LGCC gate 775
D
D2CSB (CICS-DB2 subtask block) 272
D2ENT (CICS-DB2 DB2ENTRY block) 271
D2GLB (CICS-DB2 global block) 271
D2GWA (CICS-DB2 global work area) 272
D2LOT (CICS-DB2 life of task block) 272
d2SS (CICS-DB2 static storage) 271
d2TRN (CICS-DB2 DB2TRAN block) 272
daisy chaining
APPC 1322
LU6.2 1322
data control block (DCB) 554
data event control block (DECB) 1202
data for function shipping, formatting 678
data format
transaction-routed 1329
data interchange block (DIB) 294
data services
transient 1337
data set name block (DSNB) 555
data streams for transaction routing 1327
data table request processor, file control (DFHFCDS) 571
data tables, processing using 543
database resource adapter (DRA) 281
database support 265, 281, 343, 975
DB2 265
DBCTL (database control) 281
PSB scheduling 290
PSB termination 290
system definition 290
DBCTL call processor 286
DBCTL global block (DBG) 291
DBCTL scheduling block (DSB) 291
DBCTL user-replaceable program 286
DCB (data control block) 554
DD (directory manager) domain 295
DDBR gate
END_BROWSE function 299
GET_NEXT_ENTRY function 298
START_BROWSE function 298
DDDI gate
ADD_ENTRY function 296
CREATE_DIRECTORY function 295
DELETE_ENTRY function 296
REPLACE_DATA function 297
DDLO gate
LOCATE function 297
DEACTIVATE_TRAP function, TRSR gate 1245
debkey option
READ request processing 545
READNEXT request processing 551
RESETBR request 550
STARTBR request 550
deblocking
debkey option 545
debrec option 545
DELETE_LOGICAL_SERVER function, IILS gate 694
DELETE_OPEN_TCB function, DSIT gate 315
DELETE_PARTNER function, PRFT format 873
DELETE_PENDING_REQUEST function, SHPR gate 1088
DELETE_PROGRAM function, LDLD gate 763
DELETE_PROGRAM function, PGDD gate 885
DELETE_PARTNER function, PRPT format 873
DELETE_PENDING_REQUEST function, SHPR gate 1088
DELETE_PROGRAM function, LDLD gate 763
DELETE_PROGRAM function, PGDD gate 885
DELETE_RQMODEL function, IIMM gate 696
DELETE_SUBPOOL function, SMAD gate 1099
DELETE_SUBSPACE_TCBS function, DSAT gate 307
DELETE_SUSPEND function, DSSR gate 317
DELETE_SYSTEM_DUMPCODE function, DUDT gate 391
DELETE_TCPIPSERVICE function, SOAD gate 1071
DELETE_TERM_MODEL, AITM format 131
DELETE_TIMER function, EMEM gate 500
DELETE_TRAN_DUMPCODE function, DUDT gate 384
DELETE_TRANSACTION_USER function, USXM gate 1352
DELETE_TRANSIENT_DATA function, AITM format 131
DELETE_USER function, USAD gate 1345
DELETE_USER_SECURITY function, XSD gate 1021
deletion of autoinstalled APPC devices 119
deletion of autoinstalled consoles 120
DELIVER_BACKOUT_DATA function, RMRO gate 957
DELIVER_FORGET function, RMDE gate 961
DELIVER_IC_RECOVERY_DATA function, TISIC gate 1188
DELIVER_RECOVERY function, RMDE gate 960
DENEREGISTER function, KEAR gate 733
DENEREGISTER function, SORD gate 1063
DENEREGISTER_TCLASS_USAGE function, XMCL gate 1272
DESTROY_TOKEN function, BAGD format 229
DETACH processing in application-owning region 1318
DETACH_TERMINATED_OWN_TCBS function, KEDS gate 743
device independence 135
DFH99BC 424
DFH99CC 425
CICS Transaction Server: CICS Diagnosis Reference
| DFHSZFDRD 669 | DFHSZVSL 671 | DFHTFIQ 104 |
| DFHSZFSRD 669 | DFHSZVSQ 671 | DFHTFP 1257, 1259 |
| DFHSZIDX 669 | DFHSZVSR 671 | DFHTFRF 104 |
| DFHSZPCP 669 | DFHSZVSY 671 | DFHTIDM 1229 |
| DFHSZPDX 669 | DFHSZWSL 671 | DFHTIDUF 1137, 1229 |
| DFHSZPID 669 | DFHSZXADA 671 | DFHTIEM 1162 |
| DFHSZPIX 669 | DFHSZXFR 671 | DFHTISR 1229 |
| DFHSZPOA 669 | DFHSZXLG 671 | DFHTITRI 1137, 1229, 1255 |
| DFHSZPOD 669 | DFHSZXLST 671 | DFHTMDF 1137 |
| DFHSZPOR 670 | DFHSZXNS 671 | DFHTMP 1153, 1213, 1557 |
| DFHSZPOX 670 | DFHSZXPM 671 | DFHTOAx 1214 |
| DFHSZPOY 670 | DFHSZXRA 671 | DFHTOBPS 1214 |
| DFHSZQPS 670 | DFHSZXSC 671 | DFHTON 1558 |
| DFHSZQX 670 | DFHSZXT 671 | DFHTONK 169, 173 |
| DFHSZPSB 670 | DFHSZYLG 671 | DFHTOR 998, 1214, 1558 |
| DFHSZSC 670 | DFHSZYQR 671 | DFHTORP 1558 |
| DFHSZSD 670 | DFHSZR 671 | DFHTPE 137 |
| DFHSZSH 670 | DFHSZSC 671 | DFHTPP 139, 158 |
| DFHSZSQ 670 | DFHSYSR 671 | DFHTPPS 1558 |
| DFHSZSR 670 | DFHSYSY 671 | DFHTPPAS 1558 |
| DFHSZSS 670 | DFHSZZAG 671 | DFHTQ 139, 160, 1558 |
| DFHSZPX 670 | DFHSZZEF 671 | DFHTPR 139, 162, 1558 |
| DFHSZPTE 670 | DFHSZZNG 671 | DFHTPS 140, 165, 1559 |
| DFHSZROCA 670 | DFHSZZBG 671 | DFHTR530 1559 |
| DFHSZRC 670 | DFHTACP 1191, 1194, 1554 | DFHTRADS 1248 |
| DFHSZRD 670 | DFHTAJP 723, 724, 1555 | DFHTRAO 1248 |
| DFHSZRG 670 | DFHTBS 169, 172 | DFHTRAP 1248, 1559 |
| DFHSZRN 670 | DFHTBS00 1555 | DFHTRDM 1248 |
| DFHSZRP 670 | DFHTBSB 1555 | DFHTRDS 1248 |
| DFHSZRS 670 | DFHTBSBP 171, 1555 | DFHTRDUF 1137, 1254 |
| DFHSZRTD 670 | DFHTBSD 1555 | DFHTREN 1248 |
| DFHSZREQ 670 | DFHTBSDP 1555 | DFHTRFBD 1137, 1254 |
| DFHSZRFC 670 | DFHTBSL 1555 | DFHTREFE 1137, 1254 |
| DFHSZRG 670 | DFHTBSLP 1555 | DFHTRFFD 1137, 1254 |
| DFHSZRIA 670 | DFHTBSQ 1555 | DFHTRFPP 1137, 1254 |
| DFHSZRIC 670 | DFHTBSQ 1555 | DFHTRIB 1137, 1254 |
| DFHSZRID 670 | DFHTBSR 1555 | DFHTRIP 1232, 1255, 1559 |
| DFHSZRIP 670 | DFHTBSRP 1555 | DFHTRORP 1254 |
| DFHSZRII 670 | DFHTBS 169, 173, 191, 1211, 1214 | DFHTRPG 1254 |
| DFHSZRIN 670 | DFHTBSSP 1555 | DFHTRPT 1248 |
| DFHSZRIO 670 | DFHTC macro 1320 | DFHTRPFX 1233, 1234, 1240, 1248 |
| DFHSZRIIF 670 | DFHTCBP 1556 | DFHTSR 1248 |
| DFHSZRIIQ 670 | DFHTCDUF 1137 | DFHTRS 1248 |
| DFHSZRIIS 670 | DFHTCP 1198, 1220, 1556 | DFHTRJRI 1137, 1255 |
| DFHSZRIT 670 | DFHTCRR 169, 1214, 1556 | DFHTRZCP 1559 |
| DFHSZRIW 670 | DFHTCRRPC 1556 | DFHTRZIP 1559 |
| DFHSZRC 670 | DFHTCRRPL 1556 | DFHTRZPP 1560 |
| DFHSZRN 670 | DFHTCRRPS 1556 | DFHTRZXP 1214 |
| DFHSZRP 670 | DFHTCRRPU 1556 | DFHTRX 1560 |
| DFHSZRPW 670 | DFHTCT 1209 | DFHTRX 1240 |
| DFHSZRRQ 671 | DFHTDA 1340, 1556 | DFHTRX 1260 |
| DFHSZRRQW 671 | DFHTDB 1340, 1357 | DFHTRXZ 1190 |
| DFHSZRRD 671 | DFHTDDUF 1137 | DFHTSD 1189 |
| DFHSZRT 671 | DFHTDE 1340 | DFHTSD 1190 |
| DFHSZRSI 671 | DFHTDEXL 1557 | DFHTSDUC 1190 |
| DFHSZRS 671 | DFHTDOC 1340 | DFHTSDF 1137, 1190 |
| DFHSZRTS 671 | DFHTDP 1340, 1557 | DFHTSDUS 1190 |
| DFHSZRTM 671 | DFHTDQ 1557 | DFHTSIR 1190, 1255 |
| DFHSZRXD 671 | DFHTDRM 1340, 1557 | DFHTSP 1560 |
| DFHSZRRZ 671 | DFHTDRP 1557 | DFHTSP 1189 |
| DFHSZSPI 671 | DFHTDSUC 1340 | DFHTSR 1189 |
| DFHSZSVN 671 | DFHTDTM 1340, 1557 | DFHTSS 1189 |
| DFHSZVGF 671 | DFHTDTI 1137 | DFHTSRR 1190 |
| DFHSZVS 671 | DFHTDX 1557 | DFHTSS 1190 |
| DFHSZVQA 671 | DFHTDXM 104 | DFHTST 1190 |
| DFHSZVR 671 | DFHTED 1223, 1557 | DFHTFD 137 |
| DFHSZVCR 671 | DFHTF 104 | DFHTUS 1252, 1560 |
| DFHSZVSC 671 | DFHTFBF 104 | DFHTCV 1560 |
CREATE_DOCUMENT function 350
DELETE_BOOKMARK function 356
DELETE_DATA function 355
DELETE_DOCUMENT function 354
INQUIRE_DOCUMENT function 357
INSERT_BOOKMARK function 352
INSERT_DATA function 351
REPLACE_DATA function 353
RETRIEVE_WITH_CTLINFO function 356
RETRIEVE_WITHOUT_CTLINFO function 357
DHRP gate
RECOVER_DEFINITIONS function 365
DHSL gate
SET_SYMBOL_VALUE_BY_API function 358
SET_SYMBOL_VALUE_BY_SSI function 358
DHSL gate (continued)
ADD_REPLACE_DOCTEMPLATE function 360
DELETE_DOCTEMPLATE function 363
END_BROWSE function 364
GET_NEXT function 363
INITIALIZE_DOCTEMPLATES function 360
INQUIRE_DOCTEMPLATE function 362
INQUIRE_TEMPLATE_STATUS function 362
READ_TEMPLATE function 361
START_BROWSE function 363
DIB (data interchange block) 294
DIB (DL/I interface block) 345
DIP (data interchange program) 293
storage control 294
temporary-storage control 294
terminal control 294
trace control 294
directory manager (DD) domain 295
DISABLE routine of DFHUEM 1361
DISABLE_STATISTICS function, STST gate 1086
discard, EXEC CICS 198
DISCARD_AIDS function, TFAL gate 78
DISCARD_DJAR function, EJCP gate 466
DISCARD function, LGJN gate 805
DISCARD_MEROD_INFO function, EJMI gate 485
DISCARD_PROCESSSTYPE function, BATT gate 207
DISCARD_TDQDEF function, TDTM gate 72
DISCONNECT_ALL function, LGLB gate 801
DISCONNECT for IRC 717
DISCONNECT function, LGLB gate 800
DISCONNECT function, LGST gate 810
disconnection flow for terminals,
LU-initiated 118
dispatcher (DS) domain 301
distributed program link (DPL) 327
distributed transaction processing (DTP)
logical unit type 6.1 (LU6.1) protocol 331
session failures 331
system failures 331
distribution tapes, modules
supplied 1419
DL/I
PSB scheduling 290, 975
PSB termination 290, 975
remote 975
system definition 975
DL/I database support 281, 343, 975
DL/I interface CALL macro 344
DL/I interface (continued)
CALLOCDLI macro 344
EXEC DL/I command 344
IMS service modules 344
program specification block (PSB) 344
DL/I interface block (DIB) 345
DL/I interface parameter list (DLP) 291, 345
DM (domain manager) domain 369
DMDM gate
ADD_DOMAIN function 371
QUIESCE_SYSTEM function 372
SET_PHASE function 372
WAIT_PHASE function 373
DMIQ gate
END_BROWSE function 375
GET_NEXT function 374
INQ_DOMAIN_BY_ID function 376
INQ_DOMAIN_BY_NAME function 375
INQ_DOMAIN_BY_TOKEN function 376
START_BROWSE function 374
document handler (DH) domain 349
domain calls 4
generics 7
responses 8
specific 7
tokens 8
domain gates 6
domain manager (DM) domain 369
domains 3
application (AP) 9, 21
APTC & APID domain (AP) 13
business application manager (BAM) 203
directory manager (DD) 295
dispenser (DS) 301
document handler (DH) 349
domain manager (DM) 369
dump (DU) 381
event manager (EM) 497
global catalog (GC) 249
IIOP domain (II) 693
kernel (KE) 731
loader (LD) 757
local catalog (LC) 249
lock manager (LM) 773
log manager (LG) 779
message (ME) 819
monitoring (MN) 837
object transaction service (OT) 857
parameter manager (PA) 865
program manager (PG) 881
domains 3 (continued)
Recovery Manager (RM) 919
scheduler services (SH) 1007
security manager (XS) 1017
socket (SO) 1059
statistics (ST) 1083
storage manager (SM) 1097
temporary storage domain (TS) 1167
timer (TI) 1225
trace (TR) 1233
transaction manager (XM) 727, 1261
user (US) 1341
Web (WB) 1401
DRA (database resource adapter) 281
DS (dispatcher) domain 301
DSAT format
PURGE_INHIBIT_QUERY function 325
TASK_REPLY function 325
DSAT gate
ATTACH function 302
CANCEL_TASK function 306
CHANGE_MODE function 303
CHANGE_PRIORITY function 305
CLEAR_MATCH function 305
DELETE_SUBSPACE_TCBS function 307
FREE_SUBSPACE_TCBS function 306
NOTIFY_DELETE_TCBS function 305
SET_PRIORITY function 305
TCB_POOL_MANAGEMENT function 307
dSB (data stream build) 141
interfaces, illustrated 142
page and text build (PBP) 142
terminal page processor (TPP) 142
dSB (DBCTL scheduling block) 291
dSB gate
END_BROWSE function 308
GET_NEXT function 308
INQUIRE_TASK function 310, 311
INQUIRE_TCBS function 311
SET_TCBS function 311
START_BROWSE function 308
dSiT gate
ACTIVATE_MODE function 313
ADD_TCBS function 314
DELETE_ALL_OPEN_TCBS function 316
DELETE_OPEN_TCBS function 315
DELETE_TCBS function 315
FREE_TCBS function 316
INQUIRE_DISPATCHER function 312
PROCESS_DEAD_TCBS function 316
SET_DISPATCHER function 312
dSNM block manager, file control (DFHFCDN) 569
dSNB (data set name block) 555
dSNBCUEXT 273
dSNCT (data set name block table) 321
finding in dump 1151
dSSR gate
ADD_SUSPEND function 317
DELETE_SUSPEND function 317
INQUIRE_SUSPEND_TOKEN function 317
dSSR gate (continued)
RESUME function 319
SUSPEND function 318
WAIT_MVS function 319
WAIT_OLDLC function 322
WAIT_OLDW function 321
dU (dump) domain 381
dUA (dump domain anchor block) 413
dUQT gate
ADD_SYSTEM_DUMPCODE function 390
ADD_TRAN_DUMPCODE function 382
DELETE_SYSTEM_DUMPCODE function 391
DELETE_TRAN_DUMPCODE function 384
ENDBR_SYSTEM_DUMPCODE function 396
ENDBR_TRAN_DUMPCODE function 389
GETNEXT_SYSTEM_DUMPCODE function 395
GETNEXT_TRAN_DUMPCODE function 388
INQUIRE_SYSTEM_DUMPCODE function 392
INQUIRE_TRAN_DUMPCODE function 385
SET_SYSTEM_DUMPCODE function 393
SET_TRAN_DUMPCODE function 386
STARTBR_SYSTEM_DUMPCODE function 394
STARTBR_TRAN_DUMPCODE function 387
dUDU gate
SYSTEM_DUMP function 400
TRANSACTION_DUMP function 397
dump (DU) domain 381
dUMP_DATA function, EJDU gate 479
dump domain anchor block (DUA) 413
dump domain open block 413
dUMP_STACK function, EJDU gate 480
dump utility program (DFHDU530) 419
dUMPDS_CLOSE function, DUSR gate 403
dUMPDS_OPEN function, DUSR gate 403
dUMPDS_SWITCH function, DUSR gate 403
dumps
finding FCT, or TCT in 1151
finding in dump 1151
system 381, 731
transaction 381
dumps, FEPI
interpreted areas 663
common area 665
connections 667
conversations 667
DQEs 668
nodes 666
pools 665
property sets 665
Index 1601
dumps, FEPI (continued)
static area 665
targets 666
DUSR gate
CROSS_SYSTEM_DUMP_AVAIL
function 402
DUMPDS_CLOSE function 403
DUMPDS_OPEN function 403
DUMPDS_SWITCH function 403
INQUIRE_CURRENT_DUMPDS
function 405
INQUIRE_DUMPDS_AUTOSWITCH
function 405
INQUIRE_DUMPDS_OPEN_STATUS
function 405
INQUIRE_INITIAL_DUMPDS
function 406
INQUIRE_RETRY_TIME
function 408
INQUIRE_SYSTEM_DUMP
function 407
SET_DUMPDS_AUTOSWITCH
function 405
SET_DUMPTABLE_DEFAULTS
function 407
SET_INITIAL_DUMPDS
function 406
SET_RETRY_TIME function 408
SET_SYSTEM_DUMP function 408
DWE (deferred work element) 1132
dynamic allocation sample program
(DYNALLOC) 423
dynamic log
as used by file control 541
for restartable transactions 1309
dynamic transaction backout
READ UPDATE request 545
WRITE request 546
dynamic transaction backout (DTB)
transaction restart 1309

E
EDF (execution diagnostic facility) 527
EJB (EXEC interface block) 511
EIC (EXEC interface communication area) 512
EIP (EXEC interface program) 511
EIS (EXEC interface storage) 513
EJ domain 451
EJCB, EJCG, EJDG, EJBG Gates
(E) 451
EJ domain (E) 451
EJBB gate
END_BROWSE 454
GET_NEXT 453
START_BROWSE 452
EJBG gate
ADD_BEAN 454
CONFIRM_ALL_BEANS 455
DELETE_ALL_BEANS 456
DELETE_BEAN 456
GET_BEAN_DD 457
INQUIRE_BEAN 458
EJCB gate
END_BROWSE 460
GET_NEXT 459
EJCB gate (continued)
START_BROWSE 458
EJCG gate
ACTION_CORBASERVER
function 461
ADD_CORBASERVER function 461
AMEND_CORBASERVER
function 462
INQUIRE_CORBASERVER
function 463
RESOLVE_CORBASERVER
function 464
SET_ALL_STATE function 465
WAIT_FOR_CORBASERVER
function 465
EJCP gate
DISCARD_DJAR function 466
INSTALL_DJAR function 466
PRE_INSTALL_DJAR function 467
PUBLISH_CORBASERVER
function 467
PUBLISH_DJAR function 468
RETRACT_CORBASERVER
function 468
RETRACT_DJAR function 469
EJCB gate
END_BROWSE_OBJECT 486
GET_NEXT_OBJECT 487
INQUIRE_OBJECT 488
INQUIRE_STORES 488
RETRIEVE_STATISTICS 489
START_BROWSE_OBJECT 490
EJO gate
ACTIVATE_OBJECT 490
CLOSE_OBJECT_STORE 491
OPEN_OBJECT_STORE 491
REMOVE_OBJECT 492
REMOVE_STORE 493
STORE_OBJECT 493
elements list 1419
types of elements 1419
EM (event manager) domain 497
EMBR gate
END_BROWSE_EVENT function 506
END_BROWSE_TIMER function 508
GET_NEXT_EVENT function 505
GET_NEXT_TIMER function 507
INQUIRE_EVENT function 504
INQUIRE_TIMER function 506
START_BROWSE_EVENT
function 505
START_BROWSE_TIMER
function 507
EMEM gate
ADD_SUBEVENT function 497
CHECK_TIMER function 498
DEFINE_ATOMIC_EVENT
function 498
DEFINE_COMPOSITE_EVENT
function 499
DEFINE_TIMER function 499
DELETE_EVENT function 500
DELETE_TIMER function 500
FIRE_EVENT function 501
FORCE_TIMER function 501
INQUIRE_STATUS function 502
REMOVE_SUBEVENT function 502
RETRIEVE_REATTACH_EVENT
function 503
RETRIEVE_SUBEVENT function 503
TEST_EVENT function 504
emergency restart 197
after 197
ENABLE routine of DFHUEM 1361
END_BACKOUT function, RMRO
gate 958
END_BROWSE_ALL function, LGBA
gate 781
END_BROWSE_CHAINS function, LGCC
gate 785
END_BROWSE_EVENT function, EMBR
gate 506
END_BROWSE function, AIQ
format 130
END_BROWSE function, CCCC
gate 253
END_BROWSE function, DDBR
gate 299
END_BROWSE function, DHTM
gate 364
END_BROWSE function, DMIQ
gate 375

1602
CICS Transaction Server: CICS Diagnosis Reference
End_Browse function, DSBR gate 308
End_Browse function, EJBB gate 454
End_Browse function, EJCB gate 460
End_Browse function, EJDB gate 471
End_Browse function, IIMM gate 705
End_Browse function, LDLD gate 766
End_Browse function, LGJN gate 794
End_Browse function, LGLD gate 803
End_Browse function, LGJN gate 809
End_Browse function, LGT gate 1074
End_Browse function, TSBR gate 1187
End_Browse function, TSSB gate 1183
End_Browse function, WBP gate 1402
End_Browse_Object function, EJOB gate 486
End_Browse_PROCESSTYPE function, EJOB gate 1155, 1160,
End_Browse_PROGDEF function, PCIS gate 394
End_Browse_TCLASS function, XMCL gate 1271
End_Browse_TDQDEF function, TDM gate 71
End_Browse_TIMER function, EMBR gate 508
End_Browse_TRANDEF function, XMBD gate 1266
End_Browse_TRANSACTION function, XMIQ gate 1286
End_Browse_TXN_TOKEN function, XMIQ gate 1287
End_Browse_BROWSE function, LGCB gate 782
End_Browser_BROWSE function, RMDE gate 961
End_Browser_BROWSE function, RMLN gate 943
End_Browser_BROWSE function, PRCM format 871
End_Browser_BROWSE function, PRCM format 394
End_Browser_BROWSE function, USXM gate 1352
End_Browser_BROWSE function, RMUW gate 928
End_Browser_BROWSE function, RMWT gate 954
End_Browser_BROWSE function, EJB gate 220
End_Browser_CONTAINER function, BABR gate 223
End_Browser_PROCESS function, BABR gate 225
End_Browser_SYSTEM_DUMPCODE function, DUDT gate 396
End_Browser_TRAN_DUMPCODE function, DUDT gate 396
Environment, function shipping 676
EPB (exit program block) 1155, 1160, 1359
EPL (exit program link) 1359
ERROR function, LGGL gate 813
ESDS (entry-sequenced data set) 549
unnecessary DELETE request 549
WRITE request processing 546
ESTABLISH_LANGUAGE function, APLH gate 33
ESTABLISH_LANGUAGE function, APLI gate 36
ESTABLISH_LANGUAGE function, APLJ gate 38
ESTAE exit routine 379, 715, 755, 1127, 1128
event manager (EM) domain 497
EXCEPTION_DATA_PUT function, MNMN gate 837
exchange log name (XLN) 1375
EXCI (external CICS interface) 533
design overview 533
EXEC CALL interface 533
EXEC CICS interface 534
program interfaces 533
EXEC CALL interface of EXCI 533
EXEC CICS CREATE 198
EXEC CICS DELETE command 549
EXEC CICS DISCARD 198
EXEC CICS ENDBR command 551
EXEC CICS interface of EXCI 534
EXEC CICS READ command 544
EXEC CICS READNEXT command 550
EXEC CICS READPREV command 550
EXEC CICS RESETBR command 550
EXEC CICS STARTBR command 550
EXEC CICS UNLOCK command 548
EXEC CICS WRITE command 545
EXEC DLI command 344
EXEC interface block (EIB) 511
EXEC interface communication area (EIC) 512
EXEC interface module, file control (DFHEIFC) 565
EXEC interface program (EIP) 511
EXEC interface storage (EIS) 513
execute diagnostic facility (EDF) 527
exit interface, user
control blocks, illustrated 1362
exit program block (EPB) 1155, 1160, 1359
exit program link (EPL) 1359
Exit
VTAM 124, 1399
EXPLICIT_OPEN function, LGJN gate 796
EXPORT_CERTIFICATE_DATA function, SOIS gate 1069
EXPORT_CERTIFICATE_DATA function, SOSE gate 1078
EXPORT_SYMBOL_LIST function, DUSL gate 359
extended recovery facility (XRF) 531
external CICS interface (EXCI) 533
design overview 533
EXEC CALL interface 533
EXEC CICS interface 534
programming interfaces 533
EXTRACT-EXIT routine of DPUHEM 1362
INDEX

Index 1603
file control 539 (continued)
  RLS quiesce initiation (DFHFCQI) 588
  RLS quiesce processor (DFHFCQU) 588
  RLS quiesce send transaction (DFHFCQS) 588
  RLS record management processor (DFHFCRS) 596
  RLS restart (DFHFCRR) 595
  RLS VSAM interface processor (DFHFCRV) 596
  share control block manager (DFHFCRL) 592
  shared data table function ship program (DFHFCDSX) 571
  shared resources pool processor (DFHFCPL) 579
  shutdown program (DFHFCSD) 596
  STARTBR request 550
  static storage (FC static) 556
  statistics program (DFHFCST) 597
  table entry (FRTE) 558
  table manager (DFHFCMT) 580
  UNLOCK request 548
  user exits 655
  VSAM interface program (DFHFCVP) 598
  VSAM request processor (DFHFCVS) 599
  WRITE request 545
  file input/output area (FIOA) 552, 559
  file lasting access block (FLAB) 552
  file request anchor block (FRAB) 552, 560
  file request handler, file control (DFHFCFR) 572
  file request thread element (FRTE) 552, 561
  file state program, file control (DFHFCPS) 574
  files
    DELETE request 549
    LSR (local shared resources) 543
    READ request 543
    FIND_PROFILE, XMFED gate 1277
    FIND_REQUEST_STREAM function, IRKH gate 697
    FIND_TRANSACTION OWNER function, TFAL gate 78
    FIOA (file input/output area) 552, 559
    FREEMAIN after REWRITE request 548
    FREEMAIN after WRITE request 547
    FREEMAIN during UNLOCK request processing 548
    FREEMAIN in ENDBR request processing 551
    READ request 545
    RESETBR request 550
    STARTBR request 550
    WRITE request 547, 548
    FIRE_EVENT function, EMEM gate 501
    FLAB (file lasting access block) 552
    FLATTEN process
      FLATTEN 191
      FLATTEN_TRANSACTION_USER function, USXM gate 1353
      FLATTEN_USER function, USFL gate 1349
      FLATTEN_USER_SECURITY function, XSF gate 1025
      FLLBs (file control locks locator blocks)
      file control locks locator blocks (FLLBs) 552
      flow for sign-on to consoles
      flow for sign-on to consoles 117
      FORCE_DATA function, LGWF gate 810
      FORCE function, LGGL gate 789
      FORCE function, RMRE gate 951
      FORCE_JNL function, LGGL gate 791
      FORCE_PURGE_INHIBIT_QUERY function, XMPP format 1306
      FORCE_START function, PAGP gate 865
      FORCE_TIMER function, EMEM gate 501
      FORCE_UOW function, RMUW gate 925
      FORGET function, OTCO gate 862
      FORGET function, OTSU gate 861
      formats, domain call
        generic 7
        ownership of 7
        specific 7
      formatting data for function shipping 678
      forward recovery logging 541
      FRAB (file request anchor block) 552, 560
      FRAB and FLAB processor, file control (DFHFCPL) 572
      FREE_HANDLE_TABLES function, PGHM gate 894
      FREE processing in application-owning region 1320
      FREE processing in terminal-owning region 1325
      FREE_SUBSPACE_TCBS function, DSAT gate 306
      FREE_SURROGATE function, RTSU gate 58
      FREE_TCB function, DSIT gate 316
      FREE_TCBS function, KEDS gate 744
      FREE_TXN_ENVIRONMENT function, XMXE gate 1303
      FREEMAIN_ALL TERMINAL function, SMMC gate 1107
      FREEMAIN_ALL function, SMFG gate 1102
      FREEMAIN_ALL function, SMMC gate 1106
      FRTE (file request thread element) 552, 561
      function management header
        type 5 1055
      function shipping 673
      communication with remote system 675
      data transformations 678
      distributed program link 327
      handling of DL/I requests 684
      receiving a reply at a remote system 683
      receiving a request at a remote system 682
      function shipping 673 (continued)
      handling of CICS requests 679
      (continued)
      sending a reply at a remote system 683
      sending a request to a remote system 680
      handling of DL/I requests 684
      receiving a DL/I reply from a remote system 686
      receiving a DL/I request at a remote system 686
      sending a DL/I reply at a remote system 686
      sending a DL/I request to a remote system 686
      initialization 675
      local and remote resource names 674
      programming functions with 674
      protocols 676
      resynchronization protocol 676
      sender error recovery protocol 676
      shutdown protocol 676
      symmetrical bracket protocol 676
      syncpoint functions
        ABORT 687
        COMMIT 687
        PREPARE 688
        SPR (syncpoint request) 687
      terminal control 686
      terminal control functions
        ALLOCATE 687
        FREE 687
        POINT 687
        TERM=YES operand 687
        XFRDS 679
      function shipping interface module, file control (DFHFCFR) 591
      functional layout of FEPI 657
      functions of CICS, organization 5, 10
      functions provided by gates 6
      FWA (file work area)
        FREEMAIN during REWRITE processing 548
        FREEMAIN during UNLOCK request processing 548
        FREEMAIN during WRITE request 547
        FREEMAIN in BDAM ENDBR request processing 551
        FREEMAIN in VSAM ENDBR request processing 551
        READ request 544
        RESETBR request 550
        STARTBR request 550
      G
gates, domain
      functions provided by 6
      generic 6
      specific 6
      GC (global catalog) domain 249
      GCD (global catalog data set) 249
      general request processing 543
      generalized trace facility (GTF) 1233
GET_NEXT_TIMER function, EMBR 1033
GET_NEXT_TDQDEF function, TDTM 1034
generic formats 7
generic gates 6
GENERIC option of DELETE request 549
generic resource
VTAM 1365
Generic Resource - autoinstall 116
Generic resource and AT1 1367
GET_BEAN_DD function, EJBG gate 1183
GET_CONTAINER_INT0 function, BACR gate 227
GET_CONTAINER_SET function, BACR gate 228
GET function, CCCC gate 250
GET function, TSPT gate 1175
GET_HTTP_RESPONSE function, WBAP gate 1403
GET_INITIAL_DATA function, IIRP gate 698
GET_MESSAGE_BODY function, WBAP gate 1402
GETMESSAGE function, TFAI gate 79
GET_METHOD_INFO function, EJMI gate 485
GET_NEXT_ENTRY function, DDBR gate 298
GET_NEXT_EVENT function, EMBR gate 505
GET_NEXT function, AIIQ format 129
GET_NEXT function, CCCC gate 253
GET_NEXT function, DHTM gate 363
GET_NEXT function, DMQ gate 374
GET_NEXT function, DSBR gate 308
GET_NEXT function, EJBB gate 453
GET_NEXT function, EJCB gate 459
GET_NEXT function, EJDB gate 470
GET_NEXT function, IMM gate 704
GET_NEXT function, LGN gate 793
GET_NEXT function, LGLD gate 802
GET_NEXT function, LGST gate 808
GET_NEXT function, SOTB gate 1073
GET_NEXT function, TSBR gate 1186
GET_NEXT function, TSSB gate 1183
GET_NEXT_INSTANCE function, LLDL gate 765
GET_NEXT_LINK function, RMLN gate 941
GET_NEXT_OBJECT function, EJOB gate 487
GET_NEXT_PARTNER function, PRCM format 870
GET_NEXT_PROCESSTYPE function, BATT gate 206
GET_NEXT_PROGRAM function, LLDL gate 764
GET_NEXT_PROGRAM function, PGIS gate 902
GET_NEXT_TCLASS function, XMCL gate 1271
GET_NEXT_TDQDEF function, TDTM gate 69
GET_NEXT_TIMER function, EMBR gate 507
GET_NEXT_TRANDEF function, XMBD gate 1264
GET_NEXT_TRANSACTION, XMIQ gate 1282
GET_NEXT_TXN_TOKEN function, XMIQ gate 1286
GET_NEXT_UOW function, RMUW gate 926
GET_NEXT_WORK_TOKEN function, RMWT gate 954
GET_PARAMETERS function, PAGP gate 866
GET_RECOVERY_STATUS function, RITU gate 58
GET_RELEASE function, TSPT gate 1176
GET_RELEASE_SET function, TSPT gate 1176
GET_SET function, TSPT gate 1175
GET_TXN_ENVIRONMENT function, XMXE gate 1302
GET_UPDAT function, CCCC gate 251
GETMAIN function, SMG gate 1101
GETMAIN function, MMC gate 1105
GET_NEXT_ACTIVITY function, BABR gate 220
GET_NEXT_CONTAINER function, BABR gate 222
GET_NEXT_PROCESS function, BABR gate 224
GET_NEXT_SYSTEM_DUMPCODE function, DUDT gate 395
GET_NEXT_TRAN_DUMPCODE function, DUDT gate 388
GL_FORCE function, LGLB gate 801
GL_WRITE function, LGLB gate 800
global catalog (GC) domain 249
global catalog data set (GCD) 249
global trap working storage (TRGTW) 1247
global user exits 1161
XFCREQ 543, 545
GTF (generalized trace facility) 1233
GTF buffer 1247

hash table 1147
high-performance option (HPO) 1207
horizontal tabs
and device independence 135
HPO (high-performance option) 1207

ICE (interval control element) 724
chain, and AP domain termination program (STP) 112
ICP (interval control program) mapping control program (MCP) 146
terminal page retrieval program (TPR) 164
undelivered messages cleanup program (TPQ) 161
ICXM gate
BIND_FACILITY function 54
ICXM gate (continued)
INQUIRE_FACILITY function 54
RELEASE_FACILITY function 54
IDENTIFY_PROGRAM function, LGDL gate 767
IEFJSCVT 1121
IEFJSSVT 1121
IGNORE_CONDITIONS function, PGHM gate 889
II (IOP) domain 693
IIIS gate
ADD_LOGICAL_SERVER function 693
DELETE_LOGICAL_SERVER function 694
PUBLG_LOGICAL_SERVER function 694
RETRACT_LOGICAL_SERVER function 695
IIIM gate
ADD_REPLACE_RQMODEL function 695
COMMIE_RQMODELS function 696
DELETE_RQMODEL function 696
IIOP domain (II) domain 693
III (IIOP) domain 693
III (non-3270 input mapping) 143
interfaces, illustrated 143
mapping control program (MCP) 143, 147
storage control 143
terminal control 143
IIIRH gate
FIND_REQUEST_STREAM function 697
PARSE function 697
IIIRP gate
GET_INITIAL_DATA function 698
INITIALISE function 702
INVOKE function 700
RECEIVE_REPLY function 700
RECEIVE_REQUEST function 699
SEND_REPLY function 701
TERMINATE function 702
IIIRQ gate
END_BROWSE function 705
GET_NEXT function 704
INQUIRE_RQMODEL function 703
MATCH_RQMODEL function 705
START_BROWSE function 703
IIIRR gate
PROCESS_REQUESTS function 706
IMMCLOSE function, SORD gate 1064
IMPLICIT_OPEN function, LCNJ gate 797
IMPORT_CERTIFICATE_DATA function, SOIS gate 1069
IMPORT_CERTIFICATE_DATA function, SSOE gate 1078
IMPORT_SYMBOL_LIST function, DHIISL gate 360
IMPORT_TRAN function, OTTR gate 857
IMS service modules
DL/I interface 344
in-doubts, resolution of 1164
INBOUND_FLOW function, RMLN gate 935

Index 1607
KEDS gate (continued)
CREATE_TASK function 741
CREATE_TCB function 742
DETACH_TERMINATEDOWN_TCBS function 743
END_TASK function 743
FREE_TCB function 744
POP_TASK function 744
PROCESS_KERNEL_ERROR function 747
PUSH_TASK function 744
READ_TIME function 744
RESET_TIME function 745
RESTORE_TIMER function 746
SEND_DEFERRED_ABEND function 746
START_PURGE_PROTECTION function 746
START_RUNAWAY_TIMER function 747
STOP_PURGE_PROTECTION function 747
STOP_RUNAWAY_TIMER function 745
KEGD gate
INQUIRE_KERNEL function 747
SET_KERNEL function 748
kernel (KE) domain 731
kernel linkage routines 4
KETI format
NOTIFY_RESET function 753
KETI gate
CONVERT_TO_DECIMAL_TIME function 751
CONVERT_TO_STICK_FORMAT function 751
INQ_LOCAL_DATE_TIME_DECIMAL function 750
INQUIRE_DATE_FORMAT function 750
REQUEST_NOTIFY_OF_A_RESET function 749
RESET_LOCAL_TIME function 749
SET_DATE_FORMAT function 750
KGM gate
TRANSFORMATION_INITIALISATION function 752
KEYPOINT_DATA function, RMKD gate 948
keypoint list element (KPLE) block 562
KPLE (keypoint list element) block 562
KS (key-sequence data set)
GENERAL option of DELETE request 549
WRITE request for KSDS file 546

L
LACB (logon address control block) 713
LC (local catalog) domain 249
LCB (logon control block) 713
LCBE (logon control block entry) 713
LCDS (local catalog data set) 249
LD (loader) domain 757
LDLD gate
ACQUIRE_PROGRAM function 758
LDLD gate (continued)
CATALOG_PROGRAMS function 769
DEFINE_PROGRAM function 760
DELETE_PROGRAM function 763
END_BROWSE function 766
GET_NEXT_INSTANCE function 765
GET_NEXT_PROGRAM function 764
IDENTIFY_PROGRAM function 767
INQUIRE_OPTIONS function 769
INQUIRE_PROGRAM function 762
REFRESH_PROGRAM function 760
RELEASE_PROGRAM function 759
SET_OPTIONS function 768
START_BROWSE function 763
LG (log manager) domain 779
LGbA gate
START_BROWSE_ALL function 781
LGbA gate
BROWSE_ALL_GET_NEXT function 780
END_BROWSE_ALL function 781
LGCB gate
CHAIN_BROWSE_GET_NEXT function 782
END_CHAIN_BROWSE function 782
START_CHAIN_BROWSE function 782
LGCC gate
BROWSE_CHAINS_GET_NEXT function 784
CREATE_CHAIN_TOKEN function 783
DELETE_ALL function 785
DELETE_HISTORY function 786
END_BROWSE_CHAINS function 785
INQUIRE_KEYPOINT_FREQUENCY function 786
RELEASE_STORE_CHAIN_TOKEN function 786
RESTORE_CHAIN_TOKEN function 784
SET_HISTORY function 785
SET_INHIBIT_DELETE function 787
SET_KEYPOINT_FREQUENCY function 786
START_BROWSE_CHAINS function 784
SYSEN function 783
LGGL gate
CLOSE function 789
ERROR function 813
FORCE function 789
FORCE_JNL function 791
INITIALIZE function 792
OPEN function 787
UOW_TIME function 791
WRITE function 788
WRITE_JNL function 790
LGJN gate
DISCARD function 795
END_BROWSE function 794
EXPLICIT_OPEN function 796
GET_NEXT function 793
IMPLICIT_OPEN function 797
INITIALIZE function 798
LGJN gate (continued)
INQUIRE function 792
PROCESS_STATISTICS function 799
SET function 795
START_BROWSE function 793
STREAM_FAIL function 798
LGMB gate
CONNECT function 799
DISCONNECT_ALL function 801
DISCONNECT function 800
GL_FORCE function 801
GL_WRITE function 800
LGMD gate
DISCARD function 805
END_BROWSE function 803
GET_NEXT function 802
INITIALIZE function 805
INQUIRE function 802
INSTALL function 804
MATCH function 804
START_BROWSE function 802
LGMM gate
MOVE_CHAIN function 805
LGPA gate
INQUIRE_PARAMETERS function 806
SET_PARAMETERS function 806
LGLR gate
LOGSTREAM_STATS function 807
LGST gate
CONNECT function 809
DISCONNECT function 810
END_BROWSE function 809
GET_NEXT function 808
INITIALIZE function 810
INQUIRE function 807
START_BROWSE function 808
LGLWF gate
FORCE_DATA function 810
WRITE function 811
limited resources, LU6.2 1377
LINK_ACTIVITY function, BAAC gate 214
LINK_EXEC function, PGL gate 908
LINK function, PGK gate 909
LINK_PLT function, PGL gate 910
LINK_PROCESS function, BAPR gate 210
LINK_URI function, PGLU gate 910
linkage routines, kernel 4
LISTEN function, APTC gate 16, 45
LISTEN function, SOCK gate 1062
LM (lock manager) domain 773
LMLM gate
ADD_LOCK function 773
DELETE_LOCK function 775
LOCK function 773
TEST_LOCK_OWNER function 775
UNLOCK function 774
LOAD_EXEC function, PGLD gate 905
LOAD function, PGLD gate 906
loader (LD) domain 757
local catalog (LC) domain 249
local catalog data set (LCD) 249
local resource names in function shipping 674
local shared resources (LSR) 543, 553
LOCATE_AID function, TFAL gate 81
LOCATE_AND_LOCK_PARTNER function, PRFS format 871
LOCATE_AND_LOCK_TCLASS function, XMLM gate 1273
LOCATE_AND_LOCK_TRANDEF function, XMLM gate 1289
LOCATE function, DDLO gate 297
LOCATE_REMDEL_AID function, TFAL gate 81
LOCATE_SHIPPABLE_AID function, TFAL gate 82
LOCATE_TERM_MODEL function, AIIQ gate 478
LOCATE_SHIPPABLE_AID function, AIIQ gate 807
LOGSTREAM_STATS function, LGSR gate 705
LOGON for IRC 717
logon control block entry (LCBE) 715
logon control block (LCB) 715
logon address control block (LACB) 713
logon exit, DFH2ZLGX 114
LOGON for IRC 717
LOGSTREAM_STATS function, LGSR gate 807
LOOKUP_ENTRY function, EJDI gate 478
LOOKUP_ENTRY function, EJDI gate 807
lower objects first, rule 146
LSR (local shared resources) 543, 553
LU1 initiated autoinstall disconnection 137
LU-initiated autoinstall disconnection 137
LU6.2 137
LU6.1 688
LU6.1 688
LU6.2 137
class of service 1373
command processing in application-owning region 1320
command processing in terminal-owning region 1325
daisy chaining 1322
exchange log name (XLN) 1375
limited resources 1377
modules 1377
RECEIVE processing 1376
SEND processing 1376
session management 1373
session states 1376
transaction routing for APPC devices 1333
VTAM 1373

M
M32 (3270 mapping) 149
interfaces, illustrated 150
M32 (3270 mapping) 149 (continued)
message routing 135
message switching 833
BMS 834
input TIOA 834
program control 834
ROUTE operand 834
storage control 834
task control area (TCA) 834
temporary-storage control 834
terminal list table (TLT) 834
Message translation utility
ADFHCLIB distribution library 1420
ADFHMLIB distribution library 1420
ADFHMSRC distribution library 1420
ADFHPCFM distribution library 1420
ADFHPLIB distribution library 1420
ADFHPROC distribution library 1420
ADFHSDCK distribution library 1420
messages
DFHIR3762 718
MGP (message generation program) 831
microfiche, optional source listings 1420
mirror transaction 674
ML1 (LU1 printer with extended attributes mapping program) 147
interfaces, illustrated 148
mapping control program (MCP) 148
page and text build (PBP) 147, 153
partition handling program (PHP) 147, 155
program manager 146
route list resolution program (RLR) 146, 158
storage control 146
task control 146
temporary-storage control 146
terminal page retrieval program (TPR) 164
transient data control 146
undelivered messages cleanup program (TPQ) 161
MCX (BMS fast path module) mapping control program (MCP) 147
ME (message) domain 819
MEBM gate
INQUIRE_MESSAGE_DEFINITION function 820
INQUIRE_MESSAGE_LENGTH function 820
RETRIEVE_MESSAGE function 819
MEME gate
CONVERSE function 822
INQUIRE_MESSAGE function 827
INQUIRE_MESSAGE_LENGTH function 824
RETRIEVE_MESSAGE function 823
SEND_MESSAGE function 821
VALIDATE_LANGUAGE_CODE function 825
VALIDATE_LANGUAGE_SUFFIX function 826
MESR gate
SET_MESSAGE_OPTIONS function 827
message (ME) domain 819
message control block (MCB) 138
message generation program (MGP) 831

Index 1611
MRXM gate (continued)
  INIT_XM_CLIENT function 55
  multiregion operation (MRO) 847
MVS image
  MRO links between images, in a
  sysplex 709
MXT_CHANGE_NOTIFY function,
  XMNT format 1304
MXT_NOTIFY function, XMNT
  format 1304
N
NACP (node abnormal condition
  program) 849, 1197
terminal control 1202
VTAM 849
NEP (node error program) 855, 1197
  skeleton sample 855
  terminal control 1202
  user-written 855
VTAM 855
NIB (node initialization block) 114, 167,
  668, 1211
  node initialization block (NIB) 114, 167,
  668, 1211
  non-RLS lock handler, file control
  (DFHFCNQ) 587
NOTIFY_DELETE_TCB function, DSAT
  gate 305
NOTIFY function, TISR format 1228
NOTIFY_REFRESH function, APLHI
  gate 35
NOTIFY_RESET function, KETI
  format 753
O
OCO (object code only) 1, 5
object transaction service (OT) domain 857
OCO components 5
offsite recovery completion, file control
  (DFHFCOR) 587
open/close program, file control
  (DFHFPCN) 583
OPEN function, APTC gate 13, 46
OPEN function, LGGL gate 787
OPEN_OBJECT_STORE function, EJOS
  gate 491
OPEN_TRANSIENT_DATA function,
  TDOC gate 62
operator error
  abnormal condition program
  (ACP) 1257
optional source listings, CD-ROM and
  microfiche 1420
OSPWA (output services processor work
  area) 138
partition handling program
  (PHP) 155
OT (Object Transaction Service) domain 857
OTCO gate
  FORGET function 862
OTCO gate (continued)
  SET_COORDINATOR function 862
OTSU gate
  ADD_SUBORDINATE function 860
  FORGET function 861
  RESYNC function 861
  SET_VOTE function 860
OTTR gate
  BEGIN_TRAN function 858
  COMMIT function 859
  COMMIT_ONE_PHASE function 859
  IMPORT_TRAN function 857
  PREPARE function 859
  SET_ROLLBACK_ONLY
  function 860
  ownership of formats 7
PA
PA (parameter manager) domain 865
PAGP gate
  FORCE_START function 865
  GET_PARAMETERS function 866
  INQUIRE_START function 866
  parallel sessions
  allocation 687
parameter manager (PA) domain 865
PARSE function, IIRH gate 697
partner resource manager 869, 999, 1003
  partner resource table (PRT) 869
paths
  REWRITE processing 548
pattern structure 177
patterns and subpatterns 178
PAUSE_AUXILIARY_TRACE function,
  TRSR gate 1242
PBP (page and text build) 151
  3270 mapping (M32) 150
  data stream build (DSB) 142
  interfaces, illustrated 152
  LU1 printer with extended attributes
  mapping program (ML1) 148
  mapping control program
  (MCP) 147, 153
  program manager 153
  storage control 153
PCP (program control program) 881
PG (program manager) domain 881
PGAQ gate
  INQUIRE_AUTOINSTALL
  function 882
  SET_AUTOINSTALL function 883
PGDD gate
  DEFINE_PROGRAM function 883
  DELETE_PROGRAM function 885
PGEX gate
  INITIALIZE_EXIT function 886
  TERMINATE_EXIT function 887
PGHM gate
  CLEAR_LABELS function 894
  FREE_HANDLE_TABLES
  function 894
  IGNORE_CONDITIONS
  function 889
  INQ_ABEND function 892
  INQ_AID function 890
  INQ_CONDITIONS function 888
  POP_HANDLE function 893
  PUSH_HANDLE function 893
  SET_ABEND function 891
  SET_AID function 889
  SET_CONDITIONS function 887
PGIS gate
  END_BROWSE_PROGRAM
  function 904
  GET_NEXT_PROGRAM function 902
  INQUIRE_CURRENT_PROGRAM
  function 897
  INQUIRE_PROGRAM function 894
  REFRESH_PROGRAM function 905
  SET_PROGRAM function 900
  START_BROWSE_PROGRAM
  function 901
PGLD gate
  LOAD_EXEC function 905
  LOAD function 906
  RELEASE_EXEC function 907
  RELEASE function 907
PGLK gate
  LINK function 909
  LINK_PLT function 910
PERFORM_RESTART_DRDREGE function,
  SHRQ gate 1010
PERFORM_SHUNT function, RMLK
  gate 965
PERFORM_SHUNT function, RMRO
  gate 959
PERFORM_SHUTDOWN function, SHRQ
  gate 1010
PERFORM_SYSTEM_ACTION function,
  KEDD gate 739
PERFORM_UNSHUNT function, RMLK
  gate 966
PERFORM_UNSHUNT function, RMRO
  gate 959
PERFORMANCE_DATA_PUT function,
  MNMN gate 838
Persisten Sessions
  diagnosing problems 1393
  sessions unbind during restart 1393
  persistent sessions restart flow 1386
PG (program manager) domain 881
PGQ gate
  INQUIRE_AUTOINSTALL
  function 882
  SET_AUTOINSTALL function 883
PGDD gate
  DEFINE_PROGRAM function 883
  DELETE_PROGRAM function 885
PGEX gate
  INITIALIZE_EXIT function 886
  TERMINATE_EXIT function 887
PGHM gate
  CLEAR_LABELS function 894
  FREE_HANDLE_TABLES
  function 894
  IGNORE_CONDITIONS
  function 889
  INQ_ABEND function 892
  INQ_AID function 890
  INQ_CONDITIONS function 888
  POP_HANDLE function 893
  PUSH_HANDLE function 893
  SET_ABEND function 891
  SET_AID function 889
  SET_CONDITIONS function 887
PGIS gate
  END_BROWSE_PROGRAM
  function 904
  GET_NEXT_PROGRAM function 902
  INQUIRE_CURRENT_PROGRAM
  function 897
  INQUIRE_PROGRAM function 894
  REFRESH_PROGRAM function 905
  SET_PROGRAM function 900
  START_BROWSE_PROGRAM
  function 901
PGLU gate
LINK_URM function 910
PGMINFO1 area 238, 245
PGMINFO2 area 246
PGG gate
INITIAL_LINK function 911
PGRE gate
PREPARE_RETURN function 912
PGXE gate
PREPARE_XCTL_EXEC function 912
PGXM gate
INITIALIZE_TRANSACTION function 913
TERMINATE_TRANSACTION function 913
PHONETIC conversion subroutine 201
PHP (partition handling program) 153
interfaces, illustrated 155
mapping control program (MCP) 147, 155
output processor work area (OSPWA) 155
program manager 155
storage control 156
terminal control table terminal entry (TCTTE) 156
terminal output macro (TOM) 155
terminal partition extension (TPE) 155
PHPPIN 154
PHPPSC 154
PHPPSI 153
PHPPXE 154
PLT (program list table)
AP domain termination program (STP) 111
POP_TASK function, KEDS gate 744
PRCM format
END_PARTNER_BROWSE function 871
GET_NEXT_PARTNER function 870
INQUIRE_PARTNER function 869
START_PARTNER_BROWSE function 870
PRE_INITIALIZE function, DMDM format 378
PRE_INSTALL_DJAR function, EJCP gate 467
PREPARE function, OTTR gate 859
PREPARE_RETURN function, PGRE gate 912
PREPARE_SURROGATE function, RTSU gate 59
PREPARE_XCTL_EXEC function, PGXE gate 912
PRFS format
LOCATE_AND_LOCK_PARTNER function 871
PRIN format
COMPLETE_INIT function 872
START_INIT function 871
Problem solving for generic resource
generic resource 1369
problem solving 1369
process overview
adapter 657
EXEC stubs 657
FEPI as CICS transaction 657
logic flow
FEPI application programming commands 657
FEPI system programming commands 658
within adapter 659
Resource Manager 657
PROCESS_REQUESTS function, IIRR gate 706
PROCESS_STATISTICS function, LGJN gate 799
processing using data tables 543
processing using VSAM 543
processors 1159
PROFILE function, APID gate 18, 31
program check interrupt
system recovery program (SRP) 1139
program isolation deadlock 1309
program manager (PG) domain 881
program preparation utilities
command-language translator 917
program termination block (PTB) 238, 247
programming functions with function shipping 674
protocols, function shipping 676
protocols, LU6.1 function shipping 676
PRPT format
ADD_REPLACE_PARTNER function 872
DELETE_PARTNER function 873
PRT (partner resource table) 869
PSB (program specification block) 869
PSB scheduling, DBCTL 290
PSB scheduling, DL/I 290, 975
PSB termination, DBCTL 290
PSB termination, DL/I 290, 975
PTB (program termination block) 238, 247
PUBLISH_CORBASERVER function, EJCP gate 467
PUBLISH_DJAR function, EJCP gate 468
PUBLISH_LOGICAL_SERVER function, IILS gate 694
PURGE_ALLOCATE_AIDS function, TFAL gate 83
PURGE_INHIBIT_QUERY function, DSAT format 325
PURGE_TRANSACTION, XMIQ gate 1289
PUT_REPLACE function, CCCC gate 252
PUT_REPLACE function, TSPT gate 1174
Q
QUERY function 1215
QUERY_NETNAME function, APID gate 17, 31
transients, I/O data
extrapartition 1336, 1339
indirect 1336
intrapartition 1335, 1337
queue, table manager 1151
QUIESC_DOMAIN function, DMDM format 379
QUIESC function, EJGE gate 481
QUIESC receive transaction, file control (DFHFCQR) 588
QUIESC_SYSTEM function, DMDM gate 372
R
RABN (Resource definition Atom Block Name) 172
RACB (receive-any control block) 1211
RACE (receive-any RPL pool) 1218
range table 1147
RDO (resource definition online) 997
CEDA transaction 997
terminal control autoinstallation 997
RDBU (Resource Definition Update Block) 172
READ_HEADER function, WBAP gate 1404
read integrity 541
READ_INTO function, TSQR gate 1169
READ_INTO function, TSSH gate 1179
read locks, table manager 1150
READ_NEXT function, WBAP gate 1402
READ_NEXT_INTO function, TSQR gate 1171
READ_NEXT_INTO function, TSSH gate 1180
READ_NEXT_SET function, TSQR gate 1172
READ_NEXT_SET function, TSSH gate 1181
READ_SET function, TSQR gate 1170
READ_SET function, TSSH gate 1180
READ_TEMPLATE function, DHTM gate 361
READ_TIME function, KEDS gate 744
READ_TRANSIENT_DATA function, APTD gate 49
READY function, KEAR gate 733
REATTACH_REPLY function, RMUW gate 929
REBUILDRESOURCE_CLASSES function, XSRG gate 1042
receive-any control block (RACB) 1211
receive-any RPL pool (RACE) 1218
RECEIVE function, APTC gate 14, 46
RECEIVE function, SOCK gate 1060
SCP (storage control program) (continued)
partition handling program
(PHP) 156
route list resolution program
(RLR) 158
terminal control 1201
terminal page processor (TPR) 159
terminal page retrieval program
(TPR) 164
transaction failure program
(TFP) 1258
undelivered messages cleanup
program (TPQ) 161
SCTE (subsystem control table
extension) 713, 1121
SDT (system dump table) 413
secondary index 1148
SECURE_SOC_CLOSE function, SOSE
gate 1077
SECURE_SOC_INIT function, SOSE
gate 1075
SECURE_SOC_READ function, SOSE
gate 1076
SECURE_SOC_RESET function, SOSE
gate 1077
SECURE_SOC_WRITE function, SOSE
gate 1076
security manager (XS) domain 1017
SEND_DEFERRED_ABEND function,
KEDD gate 746
SEND_DO_COMMIT function, RMLK
gate 963
SEND function, APTC gate 14, 47
SEND function, SOCK gate 1060
SEND function, WBSR gate 1407
SEND_MESSAGE function, MEME
gate 821
SEND processing, LU6.2 1376
SEND_REPLY function, IIRP gate 701
SEND_RESPONSE function, WBSR gate 1404
SEND_SSL_DATA function, SOCK
gate 1060
sequential retrieval 540
service request block (SRB) 1202
SEVRREQ attribute of file
DELETE request 549
READ request 543
session management, LU6.2 1373
session states, LU6.2 1376
sessions
recovery 718
SET_ABEND function, PGHM gate 891
SET_AID function, PGHM gate 889
SET_ALL_STATE function, EJCG
gate 465
SET_ALL_STATE function, EJCG
gate 476
SET_ANCHOR function, KEDD
gate 735
SET_AUTOINSTALL function, PGAQ
gate 883
SET_AUX_TRACE_AUTOSWITCH
function, TRSR gate 1243
SET_BOUND_REQUEST function, SHPR
gate 1008
SET_BUFFERS function, TSSR gate 1185
SET_CLIENT_DATA function, RMLK
gate 946
SET_CONDITIONS function, PGHM
gate 887
SET_COORDINATOR function, OTCO
gate 862
SET_DATE_FORMAT function, KETI
gate 750
SET_DEFAULT_RECOVERY function,
KEDD gate 741
SET_DEFERRED_ABEND function,
XMER gate 1275
SET_DEFERRED_MESSAGE function,
XMER gate 1274
SET_DISPATCHER function, DSIT
gate 312
SET_DOMAIN_TRACE function, KEDD
gate 738
SET_DSA_LIMIT function, SMSR
gate 1111
SET_DTRTRAN function, XMSR
gate 1297
SET_DUMPDS_AUTOSWITCH function,
DUSR gate 405
SET_DUMPTABLE_DEFAULTS function,
DUSR gate 407
SET_EXIT_PROGRAM function, SHRT
gate 1009
SET_EXIT_STATUS function, APUE
format 100
SET function, LGJN gate 795
SET function, SOIS gate 1067
SET_GATE function, RMDM gate 948
SET_GCS function, RMDM gate 945
SET_GLOBAL_TRACE function, KEDD
gate 737
SET_HISTORY function, LGCC gate 785
SET_INHIBIT_DELETE function, LGCC
gate 787
SET_INITIAL_DUMPDS function, DUSR
gate 406
SET_INTERNAL_TABLE_SIZE function,
TRSR gate 1240
SET_KERNEL function, KEDD gate 748
SET_KEYPOINT_FREQUENCY function,
LGCC gate 786
SET_LINK function, RMLN gate 933
SET_LOCAL_LU_NAME function,
RMDM gate 948
SET_LOGNAME function, RMNM
gate 944
SET_MARK function, RMNM gate 939
SET_MCX_SUFFIX function, MNSR
gate 839
SET_MESSAGE_OPTIONS function,
MSR gate 827
SET_MONITORING_OPTIONS function,
MSR gate 840
SET_MXT function, XMSR gate 1291
SET_NETWORK_IDENTIFIER function,
XSIS gate 1032
SET_OPTIONS function, LLDDL gate 768
SET_PARAMETERS function, LGPA
gate 806
SET_PARAMETERS function, RMDM
gate 948
SET_PARAMETERS function, SOIS
gate 1084
SET_PHASE function, DMDM gate 372
SET_PRIORITY function, DSAT gate 305
SET_PROCESSTYPE function, BATT
gate 207
SET_PROGRAM function, PGIS
gate 900
SET_RECOVERY_STATUS function,
RMLN gate 937
SET_REENTRANT_PROGRAM function,
SMSR gate 1114
SET_RETRY_TIME function, DUSR
gate 408
SET_ROLLBACK_ONLY function, OTTR
gate 860
SET_SECURITY_DOMAIN function, XSIS
gate 1030
SET_SESSION function, APTC gate 16,
47
SET_START_TYPE function, TSSR
gate 1185
SET_STARTUP function, RMDM
gate 947
SET_STATISTICS_OPTIONS function,
STSC gate 1084
SET_STORAGE_PROTECT function,
SMSR gate 1112
SET_STORAGE_RECOVERY function,
SMSR gate 1110
SET_STRINGS function, TSSR gate 1185
SET_SYMBOL_VALUE_BY_API function,
DHSI gate 358
SET_SYMBOL_VALUE_BYSSI function,
DHSI gate 358
SET_SYSTEM_DUMP function, DUSR
gate 408
SET_SYSTEM_DUMPCODE function,
DUDT gate 393
SET_SYSTEM function, SQAQ gate 62
SET_TASK_TRACE function, KEDD
gate 739
SET_TCB function, DSRB gate 311
SET_TCLASS function, XMCL gate 1269
SET_TCPIP_SERVICE function, SOTB
gate 1074
SET_TQUEUE function, TDMT gate 71
SET_TERMINAL_FACILITY function,
TFIQ gate 98
SET_TRAN_DUMPCODE function,
DUDT gate 386
SET_TRANDEF function, XMXD
gate 1295
SET_TRANSACTION function, XMQI gate 1281
SET_TRANSACTIONIALIZATION
function, SMSR gate 1110
SET_TRANSACTION_TOKEN function,
XMQI gate 1288
SET_TRAP_OFF function, KEDD
gate 740
SET_TRAP_ON function, KEDD
gate 740
SET_USER_DOMAIN_PARMS function,
USIS gate 1351
SET_VOTE function, OTSU gate 860
SH (scheduler services) domain 1007
share control block manager, file control
(DFHFCRL) 592
shared data table function ship program, file control (DFHFCDTX) 571
shared data table services 542
shared resources control (SHRCTL) block 562
shared resources pool processor, file control (DFHFCL) 579
shipping TCTTE for transaction routing 120
SHPR gate
ADD_PENDING_REQUEST function 1007
DELETE_PENDING_REQUEST function 1008
SET_BOUND_REQUEST function 1008
SHRCTL (shared resources control) block 562
SHRQ gate
PERFORM_REGULAR_DREDGE function 1010
PERFORM_RESTART_DREDGE function 1010
PERFORM_SHUTDOWN function 1010
SHRR gate
RECEIVE_REQUEST function 1011
RETRY_REQUEST function 1011
ROUTE_REQUEST function 1010
SHRT gate
INQUIRE_EXIT_PROGRAM function 1009
SET_EXIT_PROGRAM function 1009
shutdown 198
shutdown program, file control (DFHFCSD) 596
side information (partner) 1003
sign-on (SN) routine 1047
sign-on to consoles flow 117
SIGNOFF_ATI_SESSION function, DFHSNAS subroutine 1049
SIGNOFF.Attach_Header function, DFHSNUS subroutine 1056
SIGNOFF_PRESERV_USERID function, DFHSPNU subroutine 1050
SIGNOFF_SESSION_USERID function, DFHSNSU subroutine 1052
SIGNOFF_SURROGATE function, DFHSNSG subroutine 1051
SIGNOFF_TERMINAL_USER function, DFHSNTU subroutine 1054
SIGNON_ATI_SESSION function, DFHSNAS subroutine 1048
SIGNON.Attach_Header function, DFHSNUS subroutine 1055
SIGNON_PRESERV_USERID function, DFHSPNU subroutine 1049
SIGNON_SESSION_USERID function, DFHSNSU subroutine 1051
SIGNON_TERMINAL_USER function, DFHSNTU subroutine 1053
single-phase commit process 1164
SJ (JVM) domain 727
SJIN gate
xxxxxx function 727
SJIS gate
xxxxxx function 728
SLCB (subsystem logon control block) 714
SLOWDOWN_PURGE function, TFAL gate 93
SM (storage manager) domain 1097
SMAD gate
ADD_SUBPOOL function 1098
DELETE_SUBPOOL function 1099
SMAR gate
ALLOCATE_TRANSACTION_STG function 1099
RELEASE_TRANSACTION_STG function 1100
SMCK gate
CHECK_STORAGE function 1100
RECOVER_STORAGE function 1101
SMGF gate
FREEMAIN function 1102
GETMAIN function 1101
INQUIRE_ELEMENT_LENGTH function 1103
SMMC gate
FREEMAIN_ALL_TERMINAL function 1107
FREEMAIN function 1106
GETMAIN function 1105
INITIALIZE function 1105
INQUIRE_ELEMENT_LENGTH function 1104
INQUIRE_TASK_STORAGE function 1104
SMNT format
STORAGE_NOTIFY function 1116
SMMR gate
INQUIRE_ACCESS function 1113
INQUIRE_ACCESS_TOKEN function 1114
INQUIRE_DSA_LIMIT function 1111
INQUIRE_DSA_SIZE function 1109
INQUIRE_ISOLATION_TOKEN function 1108
INQUIRE_REENTRANT_PROGRAM function 1108
INQUIRE_SHORT_ON_STORAGE function 1109
INQUIRE_STORAGE_PROTECT function 1112
SET_DSA_LIMIT function 1111
SET_REENTRANT_PROGRAM function 1114
SET_STORAGE_PROTECT function 1112
SET_STORAGE_RECOVERY function 1110
SET_TRANSACTION_ISOLATION function 1110
SWITCH_SUBSPACE function 1110
UPDATE_SUBSPACE_TCB_INFO function 1114
SN (sign-on) routine 1047
SO (socket) domain 1059
SOAD gate
ADD_REPLACE_TCPIPSERVICE function 1070
DELETE_TCPIPSERVICE function 1071
SOCK gate
CLOSE function 1062
LISTEN function 1062
RECEIVE function 1060
SEND function 1060
SEND_SSL_DATA function 1069
socket (SO) domain 1059
SOIS gate
DELETE_CERTIFICATE_DATA function 1070
EXPORT_CERTIFICATE_DATA function 1069
IMPORT_CERTIFICATE_DATA function 1069
INITIALIZE_ENVIRONMENT function 1065
INQUIRE function 1065
INQUIRE_STATISTICS function 1067
SET function 1067
SET_PARAMETERS function 1064
VERIFY function 1068
SOLICIT_INQUIRES function, TSIC gate 1188
SORD gate
DREGISTER function 1063
IMMEXECLOSE function 1064
REGISTER function 1062
SOSE gate
DELETE_CERTIFICATE_DATA function 1079
EXPORT_CERTIFICATE_DATA function 1078
IMPORT_CERTIFICATE_DATA function 1078
INITIALIZE_SSL function 1075
SECURE_SOC_CLOSE function 1077
SECURE_SOC_INIT function 1075
SECURE_SOC_READ function 1076
SECURE_SOC_RESET function 1077
SECURE_SOC_WRITE function 1076
TERMINATE_SSL function 1077
SOTB gate
END_BROWSE function 1074
GET_NEXT function 1073
INQUIRE_TCPIPSERVICE function 1071
SET_TCPIPSERVICE function 1074
START_BROWSE function 1072
specific gates 6
SPIE exit routine 1127
SPP (syncpoint program) 686, 1131, 1157, 1163, 1375
SRB (service request block) 1202
SRP (system recovery program) abnormal termination 1139
program check interrupt 1139
runaway task 1139
system abend 1139
system recovery table (SRT) 1139
SRT (system recovery table) 1139
SSA (static storage area) 108
SSCT (subsystem communication table) 1119, 1121
SSVT (subsystem vector table) 1119, 1121
Index 1617
START_BROWSE_TRANSACTION
START_BROWSE_TRANDEF function, EMBR gate 957
START_BROWSE_TIMER function, EMBR gate 781
START_BROWSE/all function, LGBA gate 781
START_BROWSE_EVENT function, EMBR gate 505
START_BROWSE_function, AIIQ format 129
START_BROWSE_function, CCCC gate 252
START_BROWSE_function, DDBR gate 298
START_BROWSE_function, DHTM gate 37
START_BROWSE_function, DMIQ gate 374
START_BROWSE_function, DSBR gate 308
START_BROWSE_function, EJBB gate 452
START_BROWSE_function, EJCB gate 458
START_BROWSE_function, EJDB gate 469
START_BROWSE_function, IIMM gate 703
START_BROWSE_function, LDLD gate 763
START_BROWSE_function, LGJN gate 793
START_BROWSE_function, LGLD gate 802
START_BROWSE_function, LGST gate 808
START_BROWSE_function, SOTB gate 1072
START_BROWSE_function, TSBR gate 1186
START_BROWSE_function, TSSB gate 1183
START_BROWSE_function, WBAP gate 1401
START_BROWSE_OBJECT_function, EJOB gate 490
START_BROWSE_PROCESS_TYPE function, BATT gate 206
START_BROWSE_PROGRAM_function, PGIS gate 901
START_BROWSE_TCLASS_function, XMCL gate 1270
START_BROWSE_TDQDEF_function, TDTM gate 68
START_BROWSE_TIMER_function, EMBR gate 507
START_BROWSE_TRANDEF_function, XMID gate 1263
START_BROWSE_TRANSACTION_function, XMIQ gate 1282
START_BROWSE_TXN_TOKEN_function, XMIQ gate 1286
START_CHAIN_BROWSE_function, LGCB gate 782
START_DELIVERY_function, RMDE gate 959
START_GTF_FUNCTION, TRSR gate 1244
START_INIT_function, AIN format 127
START_INIT_function, CPIN format 1004
START_INIT_function, PRIN format 871
START_INTERNAL_TRACE_function, TRSR gate 1241
START_LINK_BROWSE_function, RMLN gate 940
START_PARTNER_BROWSE_function, PRCM format 870
START_PROGRAM_function, APLH gate 34
START_PROGRAM_function, APLI gate 37
START_PROGRAM_function, APLJ gate 39
START_PURGE_PROTECTION_function, KEDS gate 746
START_RUNAWAY_TIMER_function, KEDS gate 745
START_UOW_BROWSE_function, RMUW gate 925
START_WORK_TOKEN_BROWSE_function, RMWT gate 953
START_WRITE_function, CCCC gate 254
STARTBR_ACTIVITY_function, BABR gate 219
STARTBR_CONTAINER_function, BABR gate 222
STARTBR_PROCESS_function, BABR gate 224
STARTBR_SYSTEM_DUMPCODE_function, DUDT gate 394
STARTBR_TRAN_DUMPCODE_function, DUDT gate 387
static storage, file control (FC static) 556
static storage area (SSA) 108
statistics
CICS-DB2 273
statistics (ST) domain 1083
statistics collection 1091
statistics, utility program (DFHFCST) 597
statistics, utility program (DFHSTUP) 1091
STOP_AUXILIARY_TRACE_function, TRSR gate 1242
STOP_GTF_FUNCTION, TRSR gate 1244
STOP_INTERNAL_TRACE_function, TRSR gate 1241
STOP_PURGE_PROTECTION_function, KEDS gate 747
STOP_RUNAWAY_TIMER_function, KEDS gate 745
storage control macro-compatibility interface 1095
storage manager (SM) domain 1097
STORE_NOTIFY_function, SMNT format 1116
STORE_OBJECT_function, EJOS gate 493
STP (system termination program) 111
AID chain 112
CEMT SHUTDOWN request 111
common system area (CSA) 112
ICE chain 112
invocation from DFHEIPSH 111
program control 111
program list table (PLT) 111
transaction list table (XLT) 111
STREAM_FAIL_function, LGJN gate 798
STST format
COLLECT_RESOURCE_STATS_function 1088
COLLECT_STATISTICS_function 1088
STST gate
DISABLE_STATISTICS_function 1086
INQ_STATISTICS_OPTIONS_function 1083
RECORD_STATISTICS_function 1085
REQUEST_STATISTICS_function 1084
SET_STATISTICS_OPTIONS_function 1084
STATISTICS_COLLECTION_function 1085
subsystem anchor block (SAB) 1121
subsystem communication table (SSCT) 1119, 1121
subsystem connection address control block (SCCAB) 714
subsystem connection control block (SCCB) 714
subsystem control table extension (SCTE) 713, 1121
subsystem definition 1119
subsystem interface 1119
console message handling 1119
control blocks, illustrated 1122
subsystem logon control block (SLCB) 714
subsystem user definition block (SUDB) 713
subsystem vector table (SSVT) 1119, 1121
subtask control 1125
SUDB (subsystem user definition block) 713
surrogate session entry 1316
surrogate TCTTE 1332
svc 53
REWRITE request processing 548
UNLOCK request processing 548
WRITE request processing 547
SWITCH_AUXILIARY_EXTENTS_function, TRSR gate 1243
SWITCH_SUBSPACE_function, SMSR gate 1110
synchronization of tasks 723
synchronization processing, initiating 345
syncpoint 172
function shipping 687
SYNCPOINT_REQUEST, CPSP format 1005
SYISINI function, LGCC gate 783
sysplex, MVS
cross-system coupling facility (XCF) for MRO links across MVS images 709
system abend
system recovery program (SRP) 1139
system control
AP domain KC subcomponent 107
AP domain termination program (STP) 111
autoinstall terminal model manager 127
CICS-DB2 Attachment facility 265
DL/I database support 281, 343, 975
EXEC interface program (EIP) 511
file control 539
interval control 723
partner resource manager 869
program control 875
storage control macro-compatibility interface 1095
subtask control 1125
syncpoint program (SPP) 1131
table manager 1147
task-related user exit control 1155
terminal control 1195
trace control macro-compatibility interface 1231
transient data control 1335
user exit control 1359
system definition, DBCTL 290
system definition, DL/I 975
system dump formatting program 1135
SYSTEM_DUMP function, DUDU gate 400
system dump table (SDT) 413
system dumps 381, 731
system EIB 512
system entries, TCT (terminal control table) 676
system programming commands, FEPI logic flow 658
system reliability
node abnormal condition program (VTAM) 849
node error program (VTAM) 855
program error program (PEP) 879
system recovery program (SRP) 1139
task-related user exit recovery 1163
terminal abnormal condition program (TCAM, BSAM) 1191
terminal error program 1223
transaction failure program (TFP/ACP) 1257
transaction restart program, DFHERST 1309
system services
dynamic allocation sample program (DYNALLOC) 423
field engineering program 537
master terminal 817
message switching 833
operator terminal 817
resource definition online (RDO) 997
security interface 1017
subsystem interface 1119
supervisory terminal 817
system spooling interface 1145
system spooling interface 1145
system utilities
command-language translator 917
CSD utility program 277
dump utility program 419
statistics utility program 1091
system dump formatting program 1135
trace utility program 1252
T
table management program (TMP) 1213
table manager, file control (DFHFCMT) 580
tabs, horizontal
and device independence 135
tabs, vertical
and device independence 135
TACLE (terminal abnormal condition line entry) 1218
terminal control 1202
TACP (terminal abnormal condition program) 1191, 1197
default error handling 1194
message construction matrix 1193
message routines 1191
TCAM, BSAM 1191
terminal control 1202
TAKE_ACTIVITY_KEYPOINT function, RMRE gate 952
TAKE_KEYPOINT function, RMKP gate 961
TAKE_KEYPOINT function, TFAL gate 94
TAKE_TRANSACTION_DUMP function, ABAB gate 29
TAKEOVER function, USFL gate 1350
task abnormal condition abnormal condition program (DFHACP) 1257
transaction failure program (DFHTFPP) 1257
task control area facility control area associated address (TCAFCAA) 1201
task control block (TCB) 753
task interface element (TIE) 1131, 1155, 1161
task-related user exits 1155, 1163
task-related user exits 1155, 1163
task-related user exits 1155, 1163
control 1155
task blocks, illustrated 1160
transaction restart program, DFHERST 1309
transaction-related user exits 1155, 1163
transaction-related user exits 1155, 1163
(continued)
transaction-related user exits 1155, 1163
transaction-related user exits 1155, 1163
resynchronization 1157
state of 1157
TASK_REPLY function, DSAT format 325
TASK_REPLY function, KEDS format 752
task synchronization, time 723
TCA (task control area) message switching 834
terminal control 1202
user 1202
TCAFCAA (task control area facility control area associated address) 1201
TCAM (Telecommunications Access Method) 1195, 1196
TCB (task control block) 753
TCB_POOL_MANAGEMENT function, DSAT gate 307
TCB_REPLY function, KEDS format 753
TCT (terminal control table) 1195, 1211
finding in dump 1151
terminal control table prefix 1218
terminal control table wait list 1218
TCTLE (terminal control table line entry) 1202, 1218
TCTSE (terminal control table system entry) 676
TCTTE (terminal control table terminal entry) 1218
TCTTE generation
DFHZCQ 170
TCTTE layout 192
TCTTE structure 177
TDCC gate
CLOSE_ALL_EXTRA_TD_QUEUES function 63
CLOSE_TRANSIENT_DATA function 63
OPEN_TRANSIENT_DATA function 62
TDIM gate
ADD_REPLACE_TDQUEUE function 64
COMMUT_TDQDEFS function 73
DISCARD_TDQDEF function 72
END_BROWSE_TDQDEF function 71
GET_NEXT_TDQDEF function 69
INQUIRE_TDQDEF function 66
SET_TDQUEUE function 71
START_BROWSE_TDQDEF function 68
TDXN gate
BIND_FACILITY function 73
BIND_SECONDARY function 73
INQUIRE_FACILITY function 74
RELEASE_FACILITY function 74

Index 1619
Telecommunications Access Method (TCAM) 1195, 1196
temporary storage (TS) domain 1167
temporary-storage browse transaction, CEBR 527
temporary-storage control
  data interchange program (DIP) 294
  mapping control program (MCP) 146
  message switching 834
  terminal page processor (TPP) 160
  terminal page retrieval program (TPR) 164
  undelivered messages cleanup program (TPQ) 161
  terminal control table line entry (TCTLE) 1202, 1218
  terminal control table system entry (TCTSE) 676
terminal definition 192
  installing 1209
terminal error program (TEP) 1197, 1223
terminal error recovery 1197
terminal location (DFHZCT1) 1212
terminal location (DFHZLOC) 1212
  TERMINAL_NOW_UNAVAILABLE function, TFAL gate 94
terminal-owning region (TOR) 120
  terminal page scheduling program (TPS) 165
terminal paging 135
terminal query transaction 1215
terminal storage, builders 192
  terminals
    build/delete 197
  Terminals
    autoinstall 113
    LU-initiated disconnection flow 118
terminals, autoinstall logon flow 114
TERMINATE_DOMAIN function, DMDM format 379
TERMINATE_EXIT function, PGEX gate 887
TERMINATE function, EJGE gate 482
TERMINATE function, IIRP gate 702
TERMINATE_RECOVERY function, RMLN gate 939
TERMINATE_SSL function, SOSE gate 1077
TERMINATE_TRANSACTION function, PGXM gate 913
TEST_EVENT function, EMEM gate 504
TEST_LOCK_OWNER function, LMLM gate 775
testing facility, and sequential access method (BSAM) 1198
TFAL gate
  ALLOCATE function 75
  CANCEL_AID function 75
  CANCEL_AIDS_FOR_CONNECTION function 76
  CANCEL_AIDS_FOR_TERMINAL function 77
  CHECK_TRANID_IN_USE function 78
  DISCARD_AIDS function 78
  FIND_TRANSACTION_OWNER function 78
  GET_MESSAGE function 79
  INITIALIZE_AID_POINTERS function 80
  INQUIRE_ALLOCATE_AID function 80
  LOCATE_AID function 81
  LOCATE_REMDEL_AID function 81
  LOCATE_SHIPPABLE_AID function 82
  MATCH_TASK_TO_AID function 82
  PERFORCE_ALLOCATE_AIDS function 83
  RECOVER_START_DATA function 83
TFAL gate (continued)
  REMOVE_DELETE function 84
  REMOVE_EXPIRED_AID function 85
  REMOVE_EXPIRED_REMOTE_AID function 85
  REMOVE_MESSAGE function 86
  REMOVE_REMOTE_DELETES function 87
  REROUTE_SHIPPABLE_AIDS function 87
  RESCHEDULE_BMS function 88
  REQUEST_NOTIFY_INTERVAL function 1225
  REQUEST_NOTIFY_TIME_OF_DAY function 1226
TERM_AVAILABLE_FOR_QUEUE function, USXM gate 1355
terminal control 1195
  3270 mapping (M32) 151
  access method dependent interface 1201
  access methods 1201
  autoinstallation 997
  builder parameter set 167
  common interface 1201
  control blocks, illustrated 1206
  data interchange program (DIP) 294
  flow through device-dependent modules, illustrated 1206
  for function shipping 686
  interfaces, illustrated 1200
  locks 1213
  node abnormal condition program (NACP) 1202
  node error program (NEP) 1202
  non-3270 input mapping (IIP) 143
  service request facilities 1196
  sign-on component 1047
  storage control 1201
  system console support 1196
  task control 1201
  task control area (TCA)
    facility control area associated address 1201
  task control area, user 1202
  TCA (task control area), terminal control 1202
  TCAM/BSAM routines, illustrated 1205
  terminal abnormal condition line entry (TACLE) 1202
  terminal abnormal condition program (TACP) 1202
  terminal page retrieval program (TPR) 164
  transmission facilities
    TCAM 1197
    VTAM 1197
    VTAM/non-VTAM 1197
  WAIT request 1201
  ZCP and TCP common control routines, illustrated 1204
TIMEOFDAY_CONTROL_ID function 55
TIMEOFDAY_CONNECTION_ID function 55
TI (timer) domain 1225
TIE (task interface element) 1131, 1155, 1161
time-dependent task synchronization 723
time-of-day
  retrieval 723
  time-of-day control 723
timer (TI) domain 1225
TIOA (terminal input/output area) 1218
  3270 mapping (M32) 151
  LU1 printer with extended attributes
  mapping program (ML1) 149
TISR format
  NOTIFY function 1228
TISR gate
  CANCEL function 1227
  INQUIRE_EXPIRATION_TOKEN function 1227
  REQUEST_NOTIFY_INTERVAL function 1225
  REQUEST_NOTIFY_TIME_OF_DAY function 1226
TRANSACTION_DUMP function, DUDU gate 397

transaction dump table (TDT) 414

transaction dumps 381

TRANSACTION_INITIALISATION function, KEXM gate 752

TRANSACTION_INITIALIZATION function, APRM gate 40

TRANSACTION_INITIALIZATION function, APXM gate 51

TRANSACTION_INITIALIZATION function, MNXM gate 842

transaction initiation, automatic (ATI) 1336

transaction manager (XM) domain 1261

invocation of abnormal condition program 1259

transaction restart program, DFHREST 1309

transaction-routed data format 1329

transaction routing 198, 1311

data streams for 1327

DFHAPRT 1312

for APPC devices, LU6.2 1333

relay transaction 1312

surrogate TCTTE for 120

TRANSACTION_TERMINATION function, APRM gate 41

TRANSACTION_TERMINATION function, APXM gate 52

TRANSACTION_TERMINATION function, MNXM gate 843

transactions

CAT A 114, 169, 174, 1568

CATB 1569

CATR 1569

CEBR 527

CECI 275

CECS 275

CEDA 997

CEDB 997

CEDC 997

CEMT 817, 1091

CEOT 817

CEST 817

CTS 1569

CMAC 828

CMBX 674

CSFE 537

CSGM 691

CSMI 674

CSM2 674

CSM3 674

CSM5 674

CSMI 674

CSNC 718

CSNE 849

CSPG 164

CSPQ 146, 161

CSPS 146

CSZI 657

CVMI 674

CXT 1324

mirrors 674

TRANSFORMSIT function, APAP gate 30

transformations of data for function shipping 678

transformer program (DFHXTP) 1326

transformer storage area (XFRDS) 679
transient data control

abnormal condition program (ACP) 1258

mapping control program (MCP) 146

terminal page retrieval program (TPR) 164

undelivered messages cleanup program (TPQ) 161
transient data program (TDP)

deferred work element (DWE) 1335

transient data services 1337

translator, command-language 917

TRBL (trace block) 1247

TREN (trace entry) 1247

TRFCA (trace formatting control area) 1254

TRGTw (global trap working storage) 1247

TRPT gate

TRACE_PUT function 1239

TRS gate

ACTIVATE_TRAP function 1245

DEACTIVATE_TRAP function 1245

INQUIRE_AUXILIARY_TRACE function 1243

INQUIRE_GTF_TRACE function 1244

INQUIRE_INTERNAL_TRACE function 1241

PAUSE_AUXILIARY_TRACE function 1242

SET_AUX_TRACE_AUTOSWITCH function 1243

SET_INTERNAL_TABLE_SIZE function 1240

START_AUXILIARY_TRACE function 1241

START_GTF_TRACE function 1244

START_INTERNAL_TRACE function 1241

STOP_AUXILIARY_TRACE function 1242

STOP_GTF_TRACE function 1244

STOP_INTERNAL_TRACE function 1241

SWITCH_AUXILIARY EXTENTS function 1243

TS (temporary storage) domain 1167

TSBR gate

CHECK_PREFIX function 1187

END_BROWSE function 1187

GET_NEXT function 1186

INQUIRE_QUEUE function 1184

START_BROWSE function 1183

READ_INTO function 1187

READ_NEXT_INTO function 1180

READ_NEXT_SET function 1181

READ_SET function 1180

REWRITE function 1178

WRITE function 1178

TSBR gate

SET_BUFFERS function 1185

SET_START_TYPE function 1185

SET_STRINGs function 1185

TTP (terminal type parameter) illustration of 138

terminal page processor (TPP) 160

two-phase commit process 1163

type 3 SVC routine used for interregion communication 709

TYPE_PURGE function, CCCC gate 253

U

UEH (user exit handler) 1362

UEM (user exit manager) 1361

UET (user exit table) 1160, 1359

UIB (user interface block) 346

UNCHAIN_AID function, TFAL gate 95

UNFLATTEN_TRANSACTION_USER function, USXM gate 1356

UNFLATTEN_USER function, USFL gate 1350

UNFLATTEN_USER_SECURITY function, USFL gate 1025

unit of recovery descriptor (URD) 1131, 1163

UNLOCK function, LMLM gate 774

UNLOCK_TCLASS function, XMCL gate 1273

UNLOCK_TERM_MODEL function, AIIQ format 128
WRITE_HEADER function, WBAP
  gate 1405
WRITE_JNL function, LGGL gate 790
WRITE_JOURNAL_DATA function, APJC gate 33
WRITE_NEXT function, CCCC gate 254
WRITE_TRANSIENT_DATA function, APTD gate 48
WTO macro 1411
WTOR macro 1411

X

XFCREQ, global user exit
  READ request 543
  WRITE request 545
XFRDS (transformer storage area) 679
XLN (exchange log name) 1375
XLT (transaction list table)
  AP domain termination program (STP) 111
XM (transaction manager) domain 1261 invocation of abnormal condition program 1259
XMAT gate
  ATTACH function 1262
XMBD gate
  END_BROWSE_TRANDEF function 1266
  GET_NEXT_TRANDEF function 1264
  START_BROWSE_TRANDEF function 1263
XMCL gate
  ADD_REPLACE_TCLASS function 1267
  ADD_TCLASS function 1267
  DELETE_TCLASS function 1270
  DEREGISTER_TCLASS_USAGE function 1272
  END_BROWSE_TCLASS function 1271
  GET_NEXT_TCLASS function 1271
  INQUIRE_ALL_TCLASSES function 1269
  INQUIRE_TCLASS function 1268
  LOCATE_AND_LOCK_TCLASS function 1273
  REGISTER_TCLASS_USAGE function 1272
  SET_TCLASS function 1269
  START_BROWSE_TCLASS function 1270
  UNLOCK_TCLASS function 1273
XMDD gate
  DELETE_TRANDEF function 1274
XMFD format
  TRANDEF_DELETE_QUERY function 1305
  TRANDEF_NOTIFY function 1305
XMER gate
  ABEND_TRANSACTION function 1277
  INQUIRE_DEFERRED_ABEND function 1276
  INQUIRE_DEFERRED_MESSAGE function 1275
XMIQ gate
  END_BROWSE_TRANSACTION function 1286
  END_BROWSE_TXN_TOKEN function 1286
  GET_NEXT_TRANSACTION function 1287
  GET_NEXT_TXN_TOKEN function 1286
  INQUIRE_TRANSACTION function 1277
  INQUIRE_TRANSACTION_TOKEN function 1287
  PURGE_TRANSACTION function 1289
  SET_TRANSACTION function 1288
  START_BROWSE_TRANSACTION function 1282
  START_BROWSE_TXN_TOKEN function 1286
XMLD gate
  LOCATE_AND_LOCK_TRANDEF function 1289
  UNLOCK_TRANDEF function 1290
XMNT format
  MXT_CHANGE_NOTIFY function 1304
  MXT_NOTIFY function 1304
XMPP format
  FORCE_PURGE_INHIBIT_QUERY function 1306
XMRE gate
  INQUIRE_DTRTRAN function 1291
  INQUIRE_MXT function 1290
  SET_DTRTRAN function 1292
  SET_MXT function 1291
XMRE gate (continued)
  ADD_REPLACE_TRANDEF function 1292
  INQUIRE_REMOTE_TRANDEF function 1300
  INQUIRE_TRANDEF function 1296
  SET_TRANDEF function 1295
XMRE gate (continued)
  REPORT_MESSAGE function 1276
  SET_DEFERRED_ABEND function 1275
  SET_DEFERRED_MESSAGE function 1274
XMRE gate (continued)
  INQUIRE_USER_ATTRIBUTES function 1022
  VALIDATE_USERID function 1024
XSFIL gate
  GENERATE_APPC_BIND function 1033
  GENERATE_APPC_RESPONSE function 1034
  VALIDATE_APPC_RESPONSE function 1034
XSIS gate
  INQ_SECURITY_DOMAIN_PARMS function 1027
  INQUIRE_REGION_USER function 1027
  SET_NETWORK_IDENTIFIER function 1032
  SET_SECURITY_DOMAIN function 1030
XSIS gate
  GENERATE_PASSTICKET function 1035
  INQUIRE_PASSWORD_DATA function 1036
  UPDATE_PASSWORD_DATA function 1037
XSIS gate
  CHECK_CICS_COMMAND function 1039
  CHECK_CICS_RESOURCES function 1038
  CHECK_NON_CICS_RESOURCES function 1041
  CHECK_SURROGATE_USER function 1040
  REBUILD_RESOURCE_CLASSES function 1042
XSIS gate
  ADD_TRANSACTION_SECURITY function 1043
  DEL_TRANSACTION_SECURITY function 1043
xxxxxx function, xxxx gate 727, 728
Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user’s responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply in the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore this statement may not apply to you.

This publication could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM United Kingdom Laboratories, MP151, Hursley Park, Winchester, Hampshire, England, SO21 2JN. Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.
The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Programming License Agreement, or any equivalent agreement between us.

Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States, or other countries, or both:


Other company, product, and service names may be trademarks or service marks of others.
Sending your comments to IBM

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM.

Feel free to comment on what you regard as specific errors or omissions, and on the accuracy, organization, subject matter, or completeness of this book.

Please limit your comments to the information in this book and the way in which the information is presented.

To ask questions, make comments about the functions of IBM products or systems, or to request additional publications, contact your IBM representative or your IBM authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate, without incurring any obligation to you.

You can send your comments to IBM in any of the following ways:

• By mail, to this address:
  
  User Technologies Department (MP095)
  IBM United Kingdom Laboratories
  Hursley Park
  WINCHESTER,
  Hampshire
  SO21 2JN
  United Kingdom

• By fax:
  – From outside the U.K., after your international access code use 44–1962–870229
  – From within the U.K., use 01962–870229

• Electronically, use the appropriate network ID:
  – IBM Mail Exchange: GBIBM2Q9 at IBMMAIL
  – IBMLink™: HURSLEY(IDRCF)
  – Internet: idrcf@hursley.ibm.com

Whichever you use, ensure that you include:

• The publication title and order number
• The topic to which your comment applies
• Your name and address/telephone number/fax number/network ID.