JES3 Customization
JES3 Customization
| CSECT Name: IATUX20 (Create and Write Job Headers for Job Output) | 82 |
| CSECT Name: IATUX21 (Create and Write Data Set Headers for Output Data Sets) | 85 |
| CSECT Name: IATUX22 (Examine/Alter the Forms Alignment) | 87 |
| CSECT Name: IATUX23 (Create and Write Job Trailers for Job Output) | 88 |
| CSECT Name: IATUX24 (Examine the Net Id and the Devices Requested) | 90 |
| CSECT Name: IATUX25 (Examine/Modify Volume Serial Number) | 91 |
| CSECT Name: IATUX26 (Examine MVS Scheduler Control Blocks) | 93 |
| CSECT Name: IATUX27 (Examine/Alter the JDAB, JCT, and JMR) | 94 |
| CSECT Name: IATUX28 (Examine the JOB JCL Statement) | 95 |
| CSECT Name: IATUX29 (Examine the Accounting Information) | 96 |
| CSECT Name: IATUX30 (Examine Authority Level for TSO/E Terminal Commands) | 99 |
| CSECT Name: IATUX32 (Override the DYNALDSN Initialization Statement) | 102 |
| CSECT Name: IATUX33 (Modify JCL EXEC Statement and JES3 Control Statement) | 103 |
| CSECT Name: IATUX34 (Modify JCL DD Statement) | 104 |
| CSECT Name: IATUX35 (Validity Check Network Commands) | 106 |
| CSECT Name: IATUX36 (Collect Accounting Information) | 107 |
| CSECT Name: IATUX37 (Modify the JES3 Networking Data Set Header for Local Execution) | 109 |
| CSECT Name: IATUX38 (Change the SYSOUT Class and Destination for Networking Data Sets) | 110 |
| CSECT Name: IATUX39 (Modify the Data Set Header for a SYSOUT Data Set) | 112 |
| CSECT Name: IATUX40 (Modify Job Header for a Network Stream Containing a Job) | 114 |
| CSECT Name: IATUX41 (Determine the Disposition of a Job that Exceeds the Job JCL Limit) | 115 |
| CSECT Name: IATUX42 (TSO Interactive Data Transmission Facility Screening and Notification) | 116 |
| CSECT Name: IATUX43 (Modify Job Header for a Network Stream Containing SYSOUT Data) | 118 |
| CSECT Name: IATUX44 (Modify JCL Statements) | 120 |
| CSECT Name: IATUX45 (Change Job Information for Data Sets Processed by an Output Writer FSS) | 121 |
| CSECT Name: IATUX46 (Select Processors Eligible for C/I Processing) | 123 |
| CSECT Name: IATUX48 (Override Operator Modification of Output Data Sets) | 125 |
| CSECT Name: IATUX49 (Override the Address Space Selected for C/I Processing) | 126 |
| CSECT Name: IATUX50 (JES3 Unknown BSID Modifier Exit) | 128 |
| CSECT Name: IATUX57 (Select a Single WTO Routing Code for JES3) | 129 |
| CSECT Name: IATUX58 (Modify Security Information Before JES3 Security Processing) | 131 |
| CSECT Name: IATUX59 (Modify Security Information After JES3 Security Processing) | 138 |
| CSECT Name: IATUX60 (Determine Action to Take When a TSO User Is Unable to Receive a Data Set) | 145 |
| CSECT Name: IATUX61 (Cancel Jobs Going on the MDS Error Queue) | 146 |
| CSECT Name: IATUX62 (Verify a Mount Request) | 148 |
| CSECT Name: IATUX63 (Provide SSI Subsystem Installation String Information) | 150 |
| CSECT Name: IATUX66 (Determine Transmission Priority for a SNA/NJE Stream) | 152 |
| CSECT Name: IATUX67 (Determine Action When Remote Data Set Is Rejected by RACF) | 154 |
| CSECT Name: IATUX68 (Modify Local NJE Job Trailers) | 155 |
Chapter 2. JES3 Macro Instructions

Introduction

Coding Macro Instructions

Executable Macro Instructions

ABACKR (Reposition Pointer in Multirecord File)

ABLOCK (Add Record to a Multirecord File)

ABNCODE (Generate Global Symbol for Format Module)

ABNCVDEC (Convert Data to Decimal)

ABNCVHEX (Convert Data to Hexadecimal)

ABNGET (Get Storage Address of a Specified Area)

ABNPUT (Write a Line to the Dump Data Set)

ABNVRFY (Verify Specified Address)

ACALL (Call a Routine)

ACLOSE (Close a Multirecord File)

ACVX (Convert Binary Word to Hex)

ADEBLOCK (Retrieve a Record from a Multirecord File)

ADELETE (Delete a Module)

ADEQ (Release a JES3 Resource)

AENQ (Obtain a JES3 Resource)

AGETBUF (Get Buffer from JSAM Buffer Pool)

AGETMAIN (Get a Contiguous Storage Area)

AJOBNUM (Find or Return Job Number)

ALOAD (Load a Module into Storage)

ALOCATE (Locate Space for a Logical Record)

ANOTE (Note Location in a Multirecord File)

AOPEN (Open a Multirecord File)

AOPEND (Open a Multirecord Output File)

APOINT (Reposition in Multirecord File)

APURGE (Purge a Single Record File)

APUTBUF (Return a JSAM Buffer to the JSAM Buffer Pool)

APUTMAIN (Release a Contiguous Storage Area)

ARELEASE (Restore Spool Address in FDB)

ARETURN (Generate a Return Sequence)

ATEST (Test Availability of JES3 Resource)

ATIME (Standard Form: Request Asynchronous Processing)

ATRACK (Allocate a Track Group)

AWAIT (Await Completion of One or More Events)

AWRITE (Write a Single-Record File)

CONCNVRT (Convert Console Destination Class)

CONREVRT (Convert Code to Console Destination)

CONSPROF (Set Console Profile Mask in FCT)

DEQMSG (Dequeue Message Buffers from FCT)

DEVSCAN (Analyze Operands of IN and OUT Keyword Parameters)

DLOCOFF (Mark a Destination Queue Entry Inactive)

DLOCON (Mark a Destination Queue Entry Active)

DSQLOC (Locate Destination Queue Entry)

FAILDSP (Fail the Specified DSP)

GETUNIT (Obtain One or More Devices)

IATXADD (Add a Slot to the Active DJST Chain)
IATXAMDV (Display an Ambiguous Device Number) ........................................... 233
IATXARL (Build, Scan, or Delete an Allocation Requirements List) .................. 234
IATXARQ (Obtain, Release, or Query Queue Access) ....................................... 235
IATXASR (Obtain or return an OSE or WOSE buffer) ...................................... 236
IATXATDE (ATTACH/DETACH an ATDE) .......................................................... 237
IATXATF (Attach FCT to Active Chain) ............................................................ 238
IATXBDSN (Create Qualified Data Set Names) ................................................. 238
IATXBFM (Free Storage in Subpool 231, 241, or 245) .................................. 239
IATXBGM (Obtain Storage in Subpool 231, 241, or 245) ................................ 240
IATXBKIO (Access Spool Data Sets Using the Block Spooler Interface) ............ 242
IATXBPL (Create or Extend Cell Pool) ............................................................ 244
IATXBQN (Create Qualified Names) ............................................................... 247
IATXCIO (Communicate with a C/I FSS Address Space) .................................. 249
IATXCKPT (Request Checkpoint Record Access Method) .................................. 251
IATXCNDB (Modify Console Destination Block) .............................................. 254
IATXNT (Set a Negative Use or Allocation Count to Zero) ............................... 263
IATXCOPYF (Copy One Multi-Record File to Another) .................................... 264
IATXCRPL (Reset a VTAM Request Parameter List) ......................................... 266
IATXCSF (Execute Code Containing MVS WAIT Macros) ................................ 267
IATXCUE (Insert Reentrant Installation Exit Linkage) ..................................... 270
IATXCVB (Convert EBCDIC Number to Binary) .............................................. 272
IATXCVD (Convert Binary to EBCDIC Decimal) .............................................. 273
IATXDATX (Process DATCC with/without extension) ....................................... 275
IATXDDE (Dequeue a Slot from the Active DJST Chain) ................................ 276
IATXDQF (Dequeue a Block of Information) .................................................... 277
IATXDEQ (Free a RESQUEUE Entry) .............................................................. 279
IATXDPL (Delete Cell Pool/Extent) ............................................................... 280
IATXDST (Validate a Destination) ................................................................. 281
IATXELA (Add ECF to the ECF List) ............................................................... 283
IATXELD (Delete ECF from ECF List) ............................................................. 284
IATXELS (Scan ECF List for Satisfied ECF) ................................................... 285
IATXENQ (Enqueue a Block of Information) ................................................... 287
IATXENT (Build an Entry in the Entry Table) ................................................ 288
IATXERCK (Check Return Codes from VTAM Macros) .................................... 289
IATXERCV (Correct Chaining FDBs) ............................................................. 290
IATXFIFO (First In/First Out Queue Manipulation) ......................................... 291
IATXFLCB (Find an LCB) ................................................................................. 293
IATXFREQ (Free a RESQUEUE Entry) ............................................................. 294
IATXFSS (Functional Subsystem Services) ...................................................... 295
IATXFUV (Free a Save Area) ........................................................................... 299
IATXGCL (Allocate Cells From Cell Pool) ....................................................... 300
IATXGENF (General Purpose FCT Functions) .................................................. 302
IATXGET (Get a DJST Slot) ............................................................................. 304
IATXGFC (Obtain an FCT for a DSP) .............................................................. 305
IATXGFM (Execute Form: Issue an Error Message from a FSS Address Space) .................................................................................................................. 307
IATXGFM (List Form: Issue an Error Message from a FSS Address Space) .... 309
IATXGOSE (Add an Output Service Summary Entry) ....................................... 310
IATXGRQ (Get RESQUEUE Entry) ................................................................. 311
IATXGSE (Get a Save Area) ............................................................................. 313
IATXHOME (Check Node Names) .................................................................... 314
<table>
<thead>
<tr>
<th>Function Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>IATXICA (Access Method Control Block Processing)</td>
<td>315</td>
</tr>
<tr>
<td>IATXIOE (Recover From I/O CHECK Macro Error)</td>
<td>316</td>
</tr>
<tr>
<td>IATXIOX (Check an SRF I/O Completion)</td>
<td>316</td>
</tr>
<tr>
<td>IATXISMG (Write a Message to the JESMSG Data Set During Input Service)</td>
<td>317</td>
</tr>
<tr>
<td>IATXIWT (Generate Specified Message)</td>
<td>318</td>
</tr>
<tr>
<td>IATXJBNO (Perform a job number operation)</td>
<td>320</td>
</tr>
<tr>
<td>IATXJTS (Search a Job's TATs)</td>
<td>323</td>
</tr>
<tr>
<td>IATXJCT (Add, Alter, Examine, Delete, or Release a JCT)</td>
<td>326</td>
</tr>
<tr>
<td>IATXJDS (Job Data Set Control Block Access Routines)</td>
<td>328</td>
</tr>
<tr>
<td>IATXJLOK (JES3 BSC Lock Service)</td>
<td>332</td>
</tr>
<tr>
<td>IATXJMR (Read or Write the JMR)</td>
<td>334</td>
</tr>
<tr>
<td>IATXJOB (Convert Job Numbers and Job Identifiers)</td>
<td>336</td>
</tr>
<tr>
<td>IATXJQOE (Obtain the Address of a JQE)</td>
<td>340</td>
</tr>
<tr>
<td>IATXLOC (Locate a Slot in the DJST)</td>
<td>345</td>
</tr>
<tr>
<td>IATXLP (BDT Language Processor Interface Macro)</td>
<td>346</td>
</tr>
<tr>
<td>IATXLPJ3 (JES3 Local Post)</td>
<td>347</td>
</tr>
<tr>
<td>IATXLRTG (SNA Logical Record Get)</td>
<td>349</td>
</tr>
<tr>
<td>IATXLRPT (SNA Logical Record Put)</td>
<td>350</td>
</tr>
<tr>
<td>IATXMBFE (Map the SEND Macro Options)</td>
<td>351</td>
</tr>
<tr>
<td>IATXMG (Obtain Storage Address of a Specified Area)</td>
<td>352</td>
</tr>
<tr>
<td>IATXMD (Generate Message Identifier)</td>
<td>354</td>
</tr>
<tr>
<td>IATXMLWO (Multi-line WTO Message)</td>
<td>355</td>
</tr>
<tr>
<td>IATXNMNX (Determine Minimum and Maximum Values, and Accumulate Total Values)</td>
<td>357</td>
</tr>
<tr>
<td>IATXMP (Main Processor Control Table Services)</td>
<td>358</td>
</tr>
<tr>
<td>IATXMVDA (Access or Release a JES3 Table in Common Storage)</td>
<td>361</td>
</tr>
<tr>
<td>IATXMVXM (Move Data Among CSA, Private Area, and JES3AUX)</td>
<td>362</td>
</tr>
<tr>
<td>IATXNGRS (Send a Negative Response)</td>
<td>363</td>
</tr>
<tr>
<td>IATXNTS (Search the Node Table)</td>
<td>364</td>
</tr>
<tr>
<td>IATXOSBM (Scan the MOSE/OSS Chain for Work to Go to BDT)</td>
<td>366</td>
</tr>
<tr>
<td>IATXOSCI (Output Service Input Data Set Close)</td>
<td>368</td>
</tr>
<tr>
<td>IATXOSCO (End Processing on Output Device)</td>
<td>369</td>
</tr>
<tr>
<td>IATXOSG (Obtain an Input Data Record)</td>
<td>370</td>
</tr>
<tr>
<td>IATXOSOI (Output Service Input Data Set Open)</td>
<td>372</td>
</tr>
<tr>
<td>IATXOSOO (Open and Initialize Output Device)</td>
<td>373</td>
</tr>
<tr>
<td>IATXOSP (Write Out Queued Output Requests)</td>
<td>374</td>
</tr>
<tr>
<td>IATXOSPC (Access Output Service Elements for Process-SYSOUT Requests)</td>
<td>377</td>
</tr>
<tr>
<td>IATXOSPM (Call the PPQ Manager or Writer Functions)</td>
<td>378</td>
</tr>
<tr>
<td>IATXOSSC (Access Output Service Elements for Subsystem Requests)</td>
<td>382</td>
</tr>
<tr>
<td>IATXOSUP (Set or Reset RQOSESUP)</td>
<td>383</td>
</tr>
<tr>
<td>IATXOSWS (Access Output Service Elements)</td>
<td>385</td>
</tr>
<tr>
<td>IATXPDQ (Call the Pending Data Set Queue Manager)</td>
<td>387</td>
</tr>
<tr>
<td>IATXPGX (Page-Fix or Page-Free in JES3AUX)</td>
<td>391</td>
</tr>
<tr>
<td>IATXPHEX (Internal Data Translator)</td>
<td>392</td>
</tr>
<tr>
<td>IATXPJ3 (Post JES3)</td>
<td>393</td>
</tr>
<tr>
<td>IATXPOSE (Update an Output Service Summary Entry)</td>
<td>394</td>
</tr>
<tr>
<td>IATXPOST (Post a Function Using the COMPARE AND SWAP Instruction)</td>
<td>395</td>
</tr>
<tr>
<td>IATXPRES (Set Bytes in a Specified Area)</td>
<td>396</td>
</tr>
<tr>
<td>IATXPRMD (Define Operation for Process Mode Table)</td>
<td>397</td>
</tr>
<tr>
<td>IATXPRRT (Print Specified Data Areas)</td>
<td>400</td>
</tr>
<tr>
<td>IATXPS (Request Presentation Services)</td>
<td>402</td>
</tr>
<tr>
<td>IATXPTCH (Generate Patch Area)</td>
<td>403</td>
</tr>
<tr>
<td>IATXPTCK (Checkpoint the Partition TAT)</td>
<td>404</td>
</tr>
<tr>
<td>IATXRABC (Create Record Allocation Block)</td>
<td>405</td>
</tr>
<tr>
<td>IATXRABD (Purge Record Allocation Block)</td>
<td>406</td>
</tr>
<tr>
<td>IATXRABP (Move Partial Track Group Between RAB and JOB TAT)</td>
<td>407</td>
</tr>
<tr>
<td>Function Name</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>IATXRCL</td>
<td>Return Cell to Cell Pool</td>
</tr>
<tr>
<td>IATXRCLVL</td>
<td>Validate a Routing Code or Destination Class</td>
</tr>
<tr>
<td>IATXRDCCH</td>
<td>Read a Chained, Single-Record File into Storage</td>
</tr>
<tr>
<td>IATXRELC</td>
<td>Restore Spool Address in FDB</td>
</tr>
<tr>
<td>IATXRMEP</td>
<td>Branch to the JMF SSI Response Sampling Routine</td>
</tr>
<tr>
<td>IATXRST</td>
<td>Activate I/O Activity for a SNA RJP Session</td>
</tr>
<tr>
<td>IATXRUGT</td>
<td>Obtain an RU</td>
</tr>
<tr>
<td>IATXRUPT</td>
<td>Obtain an Empty RU</td>
</tr>
<tr>
<td>IATXSAS</td>
<td>Locate a Spool Data Set Entry or Validate a Spool Record Address</td>
</tr>
<tr>
<td>IATXSCH</td>
<td>Schedule a Job to a CI, POSTSCAN, or DISABLE DSP</td>
</tr>
<tr>
<td>IATXSCN1</td>
<td>Scan Input Commands</td>
</tr>
<tr>
<td>IATXSCN2</td>
<td>Scan Input Command Sublists</td>
</tr>
<tr>
<td>IATXSAS</td>
<td>Initialize the Spool Data Management List</td>
</tr>
<tr>
<td>IATXSEC</td>
<td>Security Authorization Services</td>
</tr>
<tr>
<td>IATXSIO</td>
<td>Initialize Spool I/O Operation</td>
</tr>
<tr>
<td>IATXSIO</td>
<td>(Initiate Spool I/O Operation)</td>
</tr>
<tr>
<td>IATXSIO</td>
<td>(Write SMF Records)</td>
</tr>
<tr>
<td>IATXSIO</td>
<td>(Set Up Parameter Linkage for the SNA RJP State Manager)</td>
</tr>
<tr>
<td>IATXSIO</td>
<td>(Obtain a Work Area for SNARJP Termination)</td>
</tr>
<tr>
<td>IATXSIO</td>
<td>(Determine the Caller’s State and Release Storage)</td>
</tr>
<tr>
<td>IATXSIO</td>
<td>(Interface to the SNARJP FAILDSP Processor)</td>
</tr>
<tr>
<td>IATXSIO</td>
<td>(Determine Caller’s State and Obtain Storage)</td>
</tr>
<tr>
<td>IATXSIO</td>
<td>(Increase or Decrease the LCB Use Count)</td>
</tr>
<tr>
<td>IATXSIO</td>
<td>(Send a SNA RJP Message)</td>
</tr>
<tr>
<td>IATXSIO</td>
<td>(Set Up Interface to SNARJP Termination Status Manager)</td>
</tr>
<tr>
<td>IATXSIO</td>
<td>(Create an Entry in the SNA RJP Trace Table)</td>
</tr>
<tr>
<td>IATXSIO</td>
<td>(Checkpoint Status of Spool Data Sets)</td>
</tr>
<tr>
<td>IATXSIO</td>
<td>(Enqueue SETPRT Requests)</td>
</tr>
<tr>
<td>IATXSIO</td>
<td>(Add or Delete Entry from Storage Management Queue)</td>
</tr>
<tr>
<td>IATXSIO</td>
<td>(Post/Test DSPs Waiting for Specialized Rescheduling)</td>
</tr>
<tr>
<td>IATXSIO</td>
<td>(Gather JES3 Statistics)</td>
</tr>
<tr>
<td>IATXSIO</td>
<td>(Set Dispatching Mode)</td>
</tr>
<tr>
<td>IATXSIO</td>
<td>(Search for Allocated STT Extents)</td>
</tr>
<tr>
<td>IATXSIO</td>
<td>(Access SYSUNITS Table)</td>
</tr>
<tr>
<td>IATXSIO</td>
<td>(Test FCT Mode or Task Control)</td>
</tr>
<tr>
<td>IATXSIO</td>
<td>(Obtain Local Time of Day)</td>
</tr>
<tr>
<td>IATXSIO</td>
<td>(Make Entry in a JES3 Trace Table)</td>
</tr>
<tr>
<td>IATXSIO</td>
<td>(End a SNA RJP Session)</td>
</tr>
<tr>
<td>IATXSIO</td>
<td>(Allocate User Memory Buffer)</td>
</tr>
<tr>
<td>IATXSIO</td>
<td>(Validate JES3 Spool Record Address)</td>
</tr>
<tr>
<td>IATXSIO</td>
<td>(Validate the Contents of an FDB)</td>
</tr>
<tr>
<td>IATXSIO</td>
<td>(Write Messages About Job Validation to the JES3OUT Data Set)</td>
</tr>
<tr>
<td>IATXSIO</td>
<td>(Save Information About Job Validation Spool Errors)</td>
</tr>
<tr>
<td>IATXSIO</td>
<td>(Save Information About Spool Records Associated with a Job)</td>
</tr>
<tr>
<td>IATXSIO</td>
<td>(Wait for ECF Post)</td>
</tr>
<tr>
<td>IATXSIO</td>
<td>(SNA RJP Work Station Open)</td>
</tr>
<tr>
<td>IATXSIO</td>
<td>(Recover From WRITE Macro Error)</td>
</tr>
<tr>
<td>IATXSIO</td>
<td>(SNA RJP Work Station Close)</td>
</tr>
<tr>
<td>IATXSIO</td>
<td>(Inner Macro to Determine if Operand Is in Register Notation)</td>
</tr>
<tr>
<td>IATXSIO</td>
<td>(Inner Macro to Create Routine Linkage)</td>
</tr>
<tr>
<td>IATXSIO</td>
<td>(Inner Macro for Decimal to Hexadecimal Conversion)</td>
</tr>
<tr>
<td>IATXSIO</td>
<td>(Inner Macro to Load Parameter Register)</td>
</tr>
<tr>
<td>IATXSIO</td>
<td>(Inner Macro for Checking for Mutually Exclusive Parameters)</td>
</tr>
<tr>
<td>IATXSIO</td>
<td>(Inner Macro to Generate Standard MNOTEs)</td>
</tr>
<tr>
<td>IATXSIO</td>
<td>(Inner Macro to Create Move Instructions)</td>
</tr>
<tr>
<td>IATXSIO</td>
<td>(Inner Macro for Illegal Parameter Verification)</td>
</tr>
<tr>
<td>IATXSIO</td>
<td>(Parameter Checker)</td>
</tr>
<tr>
<td>Macro Name</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>IATZTYPE</td>
<td>(Inner Macro to Generate Data Areas)</td>
</tr>
<tr>
<td>ICARDRD</td>
<td>(Read Initialization Statement)</td>
</tr>
<tr>
<td>ICONVBIN</td>
<td>(Convert to Binary)</td>
</tr>
<tr>
<td>ICONVHEX</td>
<td>(Convert to Hexadecimal Format)</td>
</tr>
<tr>
<td>INITMWLE</td>
<td>(Scan for Ending Right Parenthesis)</td>
</tr>
<tr>
<td>INTERCOM</td>
<td>(Simulate Input of Operator Message)</td>
</tr>
<tr>
<td>IPURGE</td>
<td>(Recover Spool Space)</td>
</tr>
<tr>
<td>ISCAN1</td>
<td>(Locate Fields on Initialization Statement)</td>
</tr>
<tr>
<td>ISCAN2</td>
<td>(Locate Subfield of Keyword Parameter)</td>
</tr>
<tr>
<td>ISORT</td>
<td>(Sort Internal Table Entries)</td>
</tr>
<tr>
<td>ITREAD</td>
<td>(Read the Intermediate Text for JES3 Initialization)</td>
</tr>
<tr>
<td>ITWRITE</td>
<td>(Write the Intermediate Text for JES3 Initialization)</td>
</tr>
<tr>
<td>IWASPOUT</td>
<td>(Write Initialization Error Message)</td>
</tr>
<tr>
<td>JDSADD</td>
<td>(Add New Entry to JDS)</td>
</tr>
<tr>
<td>JDSGET</td>
<td>(Access Entry in JDS)</td>
</tr>
<tr>
<td>JDSHOLD</td>
<td>(Hold JDS Entry)</td>
</tr>
<tr>
<td>JDSPPOINT</td>
<td>(Reobtain JDS Entry)</td>
</tr>
<tr>
<td>JDSPUT</td>
<td>(Checkpoint Change in Current JDS Entry)</td>
</tr>
<tr>
<td>JDSREL</td>
<td>(Release Control of JDS or Negate Entry Hold)</td>
</tr>
<tr>
<td>JESCKPNT</td>
<td>(Save JES3 Checkpoint Data)</td>
</tr>
<tr>
<td>JESCLOSE</td>
<td>(Close Unit Record or Tape Device)</td>
</tr>
<tr>
<td>JESEXCP</td>
<td>(Request I/O for Device)</td>
</tr>
<tr>
<td>JESMSG</td>
<td>(Place Message in JESMSG Data Set)</td>
</tr>
<tr>
<td>JESOPEN</td>
<td>(Open a Unit Record or Tape Device)</td>
</tr>
<tr>
<td>JESREAD</td>
<td>(Read Single Record File)</td>
</tr>
<tr>
<td>JESTAE</td>
<td>(Define Abnormal Exit Routine)</td>
</tr>
<tr>
<td>JNADD</td>
<td>(Add Entry to JNCB Chain)</td>
</tr>
<tr>
<td>JNCBHLD</td>
<td>(Hold Specified JNCB Within JNCB Chain)</td>
</tr>
<tr>
<td>JNCBREL</td>
<td>(Release a JNCB from Hold)</td>
</tr>
<tr>
<td>JNDEL</td>
<td>(Delete a JNCB from the JNCB Chain)</td>
</tr>
<tr>
<td>JNGET</td>
<td>(Access Next JNCB on the Chain)</td>
</tr>
<tr>
<td>JSERV</td>
<td>(Issue SSISERV Request)</td>
</tr>
<tr>
<td>LOGIN</td>
<td>(Establish Console Communication Linkage)</td>
</tr>
<tr>
<td>LOGOUT</td>
<td>(End Console Communication Linkage)</td>
</tr>
<tr>
<td>MDSALLOC</td>
<td>(Allocate MDS Resources)</td>
</tr>
<tr>
<td>MDSDSN</td>
<td>(Build or Delete a SETDSN Entry)</td>
</tr>
<tr>
<td>MDSERRQ</td>
<td>(Invoke Installation Exit 61)</td>
</tr>
<tr>
<td>MDSID</td>
<td>(Create MDS Trace Table Entry)</td>
</tr>
<tr>
<td>MDSJERR</td>
<td>(Analyze Allocation Error or Locate Error Message for JST Entry)</td>
</tr>
<tr>
<td>MDSJGET</td>
<td>(Get JST Entry)</td>
</tr>
<tr>
<td>MDSJST</td>
<td>(Read, Write, and Release JST Buffers)</td>
</tr>
<tr>
<td>MDSMSG</td>
<td>(Construct MDS Output Messages)</td>
</tr>
<tr>
<td>MDSVGET</td>
<td>(Get JVT Entry)</td>
</tr>
<tr>
<td>MDSVLM</td>
<td>(Build, Scan, or Delete SETVOL Entry)</td>
</tr>
<tr>
<td>MESSAGE</td>
<td>(Standard Form: Issue a Message from a JES3 DSP)</td>
</tr>
<tr>
<td>MESSAGE</td>
<td>(Execute Form: Issue a Message)</td>
</tr>
<tr>
<td>MESSAGE</td>
<td>(List Form: Issue a Message)</td>
</tr>
<tr>
<td>MNTRDEF</td>
<td>(Create Monitor Definition Entry)</td>
</tr>
<tr>
<td>MOVEDATA</td>
<td>(Move Data Exceeding 256 Bytes)</td>
</tr>
<tr>
<td>MTBL</td>
<td>(Build a Main Service Search Table Entry)</td>
</tr>
<tr>
<td>NCBAADD</td>
<td>(Add a Net Control Block to the JNCB)</td>
</tr>
<tr>
<td>NCBTAFND</td>
<td>(Locate a Net Control Block Within the JNCB)</td>
</tr>
<tr>
<td>NCBTAGET</td>
<td>(Locate Next Net Control Block Within the JNCB)</td>
</tr>
<tr>
<td>NCBTAPUT</td>
<td>(Write NCB Buffer)</td>
</tr>
<tr>
<td>NCBTAREL</td>
<td>(Release NCB Buffer)</td>
</tr>
<tr>
<td>NCKADD</td>
<td>(Checkpoint a DJC JNCB)</td>
</tr>
<tr>
<td>Macro Name</td>
<td>Page</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>NCKDEL (Delete DJCJNCB Checkpoint Record)</td>
<td>567</td>
</tr>
<tr>
<td>PURCHAIN (Purge Chained Single Record Files)</td>
<td>567</td>
</tr>
<tr>
<td>PUTUNIT (Return One or More JES3 Devices for Reallocation)</td>
<td>569</td>
</tr>
<tr>
<td>RQTAADD (Add a RESQUEUE to the RESQUEUE Chain and Subchain)</td>
<td>569</td>
</tr>
<tr>
<td>RQTADEL (Delete a RESQUEUE from a RESQUEUE Chain and Subchain)</td>
<td>570</td>
</tr>
<tr>
<td>RQTAGEN (Inner Macro to Manipulate RESQUEUE Table)</td>
<td>571</td>
</tr>
<tr>
<td>RQTAPUT (Move a RESQUEUE to Another Chain)</td>
<td>572</td>
</tr>
<tr>
<td>SPINOFF (Process Output Prior to Job Completion)</td>
<td>573</td>
</tr>
<tr>
<td>SSISERV (Provide Interface for SSI and JES3)</td>
<td>574</td>
</tr>
<tr>
<td>SUPFLAG (Set or Reset Flag in SUPUNITS)</td>
<td>578</td>
</tr>
<tr>
<td>VIOLATE (Execute Instructions Violating Storage Protection)</td>
<td>579</td>
</tr>
<tr>
<td>WRITEOSE (Write an OSE to spool)</td>
<td>579</td>
</tr>
<tr>
<td>WRTCCHAIN (Write Chained Single Record Files)</td>
<td>581</td>
</tr>
<tr>
<td>ZEROCORE (Clear Area to Binary Zeros)</td>
<td>583</td>
</tr>
<tr>
<td>Table Generating and Other Macro Instructions</td>
<td>583</td>
</tr>
<tr>
<td>IATYASM (Identify Level of MVS Macros)</td>
<td>583</td>
</tr>
<tr>
<td>IATYAWDR (Generate an AWAIT Reason Code)</td>
<td>584</td>
</tr>
<tr>
<td>IATYAWR (JES3 AWAIT Reason Codes)</td>
<td>585</td>
</tr>
<tr>
<td>IATYCNS (Generate Console Service Tables)</td>
<td>587</td>
</tr>
<tr>
<td>IATYDJF (Define a Field Within a JES3 Spool-Resident Data Area)</td>
<td>588</td>
</tr>
<tr>
<td>IATYDJR (Define a JES3 Spool-Resident Data Area)</td>
<td>593</td>
</tr>
<tr>
<td>IATYDSD (Generate a DSP Dictionary Entry)</td>
<td>595</td>
</tr>
<tr>
<td>IATYFCD (Generate Function Control Table)</td>
<td>598</td>
</tr>
<tr>
<td>IATYMOD (Identify Module)</td>
<td>602</td>
</tr>
<tr>
<td>IATYRSC (JES3 Resource Table)</td>
<td>603</td>
</tr>
<tr>
<td>IATYSRF (Map the SRF Header for a Spool Control Block)</td>
<td>603</td>
</tr>
</tbody>
</table>

**Appendix A. Information for JES3 Security Calls**

**Appendix B. Accessibility**

- Using assistive technologies
- Keyboard navigation of the user interface
- z/OS information

**Notices**

- Programming Interface Information
- Policy for unsupported hardware
- Trademarks

**Index**
Figures

1. Example of Using SMP to Implement a JES3 Installation Exit . . . . . . . . . . . . . . .4
2. IATXSEC Processing . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Types of IATXSEC Calls</td>
</tr>
<tr>
<td>2</td>
<td>JES3-Provided Global Assembler Variable (&amp;J3PLVL) for Currently Supported JES3 Releases</td>
</tr>
<tr>
<td>3</td>
<td>Return Codes Passed to JSS</td>
</tr>
<tr>
<td>4</td>
<td>Color or Intensity of Messages by Message Type</td>
</tr>
<tr>
<td>5</td>
<td>Selection of Final Routing Code for WTO/WTOR Messages</td>
</tr>
<tr>
<td>6</td>
<td>Default Algorithm for Selecting a Single Routing Code to Present to JES3</td>
</tr>
<tr>
<td>7</td>
<td>Macros Used for Issuing Messages to Different Destinations</td>
</tr>
<tr>
<td>8</td>
<td>JES3 Services Available Under Auxiliary Task</td>
</tr>
<tr>
<td>9</td>
<td>JES3 Executable Macros Available in a C/I FSS Address Space</td>
</tr>
<tr>
<td>10</td>
<td>Return Codes for a Strict B1 JES3 Security Classification</td>
</tr>
<tr>
<td>11</td>
<td>JES3 Installation Exits Listed by JES3 Function (with Mnemonic Name)</td>
</tr>
<tr>
<td>12</td>
<td>Obtaining Access to JCT or JDAB</td>
</tr>
<tr>
<td>13</td>
<td>RACF Classes Used to Protect JES3 Resources</td>
</tr>
<tr>
<td>14</td>
<td>RACF Classes Used to Protect JES3 Resources</td>
</tr>
<tr>
<td>15</td>
<td>Syntax examples</td>
</tr>
<tr>
<td>16</td>
<td>JES3 Executable Macro Instructions by JES3 Function</td>
</tr>
<tr>
<td>17</td>
<td>Return Codes Returned in Register 15 for a TYPE=CI Request</td>
</tr>
<tr>
<td>18</td>
<td>Parameter Interdependencies for IATXSDM</td>
</tr>
<tr>
<td>19</td>
<td>SAF Classes Used to Protect JES3 Resources</td>
</tr>
<tr>
<td>20</td>
<td>Relationships Between MESSAGE Parameters and WTO Parameters</td>
</tr>
<tr>
<td>21</td>
<td>Common Information Supplied for Security Calls to SAF</td>
</tr>
<tr>
<td>22</td>
<td>Support Information for IATUX58</td>
</tr>
</tbody>
</table>
**About This Document**

This document supports z/OS® (5694-A01). This document provides information to help system programmers modify JES3 processing in installations that include z/OS systems. It includes instructions on how to write JES3 installation exit routines and dynamic support programs (DSPs), and how to use JES3 macro instructions to aid in writing the exit routines and DSPs.

**Who Should Use This Document**

This document is intended for anyone who modifies the JES3 system. Usually, this person is a systems programmer. This document assumes that the reader is familiar with the conventions for writing code that runs in a 31-bit addressing environment. If you are not familiar with 31-bit coding techniques, see the z/OS MVS Programming: Assembler Services Guide.

**How to Use This Document**

This document contains three topics:

- **Chapter 1, “JES3 Dynamic Support Programs and Installation Exit Routines,”** describes how to write user dynamic support programs and installation exit routines.
- **Installation Exits** contains descriptions of all JES3 installation exits available with JES3. At the beginning of this topic, you will find a chart that shows which JES3 functions invoke which JES3 exits.
- **Introduction** describes the JES3 macro instructions and how to use them when writing installation exit routines and DSPs.

**Where to Find More Information**

This document references the following publications using the shortened version of the document title. The following table lists the shortened titles, complete titles, and order numbers of the documents not listed in z/OS Information Roadmap.

<table>
<thead>
<tr>
<th>Title</th>
<th>Short Title</th>
<th>Order Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS/VS, DOS/VSE and VM/370 Assembler Language</td>
<td></td>
<td>GC33-4010</td>
</tr>
</tbody>
</table>

**The z/OS Basic Skills Information Center**

The z/OS Basic Skills Information Center is a Web-based information resource intended to help users learn the basic concepts of z/OS, the operating system that runs most of the IBM mainframe computers in use today. The Information Center is designed to introduce a new generation of Information Technology professionals to basic concepts and help them prepare for a career as a z/OS professional, such as a z/OS system programmer.

Specifically, the z/OS Basic Skills Information Center is intended to achieve the following objectives:

- Provide basic education and information about z/OS without charge
- Shorten the time it takes for people to become productive on the mainframe
• Make it easier for new people to learn z/OS.

To access the z/OS Basic Skills Information Center, open your Web browser to the following Web site, which is available to all users (no login required):

http://publib.boulder.ibm.com/infocenter/zos/basics/index.jsp

## Additional Information

Additional information about z/OS elements can be found in the following documents.

<table>
<thead>
<tr>
<th>Title</th>
<th>Order Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>z/OS Introduction and Release Guide</strong></td>
<td>GA22-7502</td>
<td>Describes the contents and benefits of z/OS as well as the planned packaging and delivery of this new product.</td>
</tr>
<tr>
<td><strong>z/OS Planning for Installation</strong></td>
<td>GA22-7504</td>
<td>Contains information that lets users:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Understand the content of z/OS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Plan to get z/OS up and running</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Install the code</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Take the appropriate migration actions</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Test the z/OS system</td>
</tr>
<tr>
<td><strong>z/OS Information Roadmap</strong></td>
<td>SA22-7500</td>
<td>Describes the information associated with z/OS including z/OS documents and documents for the participating elements.</td>
</tr>
<tr>
<td><strong>z/OS Summary of Message and Interface Changes</strong></td>
<td>SA22-7505</td>
<td>Describes the changes to messages for individual elements of z/OS. Note: This document is provided in softcopy only on the message bookshelf of the z/OS collection kit.</td>
</tr>
</tbody>
</table>

## Determining If a Publication Is Current

As needed, IBM updates its publications with new and changed information. For a given publication, updates to the hardcopy and associated BookManager softcopy are usually available at the same time. Sometimes, however, the updates to hardcopy and softcopy are available at different times. Here’s how to determine if you are looking at the most current copy of a publication:

1. At the end of a publication’s order number there is a dash followed by two digits, often referred to as the dash level. A publication with a higher dash level is more current than one with a lower dash level. For example, in the publication order number GC28-1747-07, the dash level 07 means that the publication is more current than previous levels, such as 05 or 04.

2. If a hardcopy publication and a softcopy publication have the same dash level, it is possible that the softcopy publication is more current than the hardcopy publication. Check the dates shown in the Summary of Changes. The softcopy publication might have a more recently dated Summary of Changes than the hardcopy publication.

3. To compare softcopy publications, you can check the last two characters of the publication's filename (also called the book name). The higher the number, the more recent the publication. Also, next to the publication titles in the CD-ROM booklet and the readme files, there is an asterisk that indicates whether a publication is new or changed.
Summary of Changes

Summary of Changes
for SA22-7542-08
z/OS Version 1 Release 11

The document contains information previously presented in z/OS JES3 Customization, SA22-7542-07, which supports z/OS Version 1 Release 10.

New Information
• None.

Changed Information
• IATUX72 - Added information about SSI 70. See "CSECT Name: IATUX72 (Examine/Modify a Temporary OSE or an OSE Moved to Writer Queue)" on page 164.

You may notice changes in the style and structure of some content in this document—for example, headings that use uppercase for the first letter of initial words only, and procedures that have a different look and format. The changes are ongoing improvements to the consistency and retrievability of information in our documents.

This document contains terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

Summary of Changes
for SA22-7542-07
z/OS Version 1 Release 10

The document contains information previously presented in z/OS JES3 Customization, SA22-7542-06, which supports z/OS Version 1 Release 9.

New Information
• None.

Changed Information
• ACALL - Added macro parameter and description. See "ACALL (Call a Routine)" on page 185.
• IATUX63 - Added and changed programming consideration parameters. See "CSECT Name: IATUX63 (Provide SSI Subsystem Installation String Information)" on page 150.
• IATXVFDB - Added and changed macro description, syntax, and parameters. See "IATXVFDB (Validate the Contents of an FDB)" on page 481.

This document contains terminology, maintenance, and editorial changes, including changes to improve consistency and retrievability.

Summary of Changes
for SA22-7542-06
z/OS Version 1 Release 9
The document contains information previously presented in z/OS JES3 Customization, SA22-7542-05, which supports z/OS Version 1 Release 8.

New Information
• IATXASR - New macro. See “IATXASR (Obtain or return an OSE or WOSE buffer)” on page 236.

Changed Information
• IATXGOSE - Added macro parameter and description. See “IATXGOSE (Add an Output Service Summary Entry)” on page 310.
• IATXPOSE - Added macro parameter and description. See “IATXPOSE (Update an Output Service Summary Entry)” on page 394.
• IATXPDQ - Changed macro parameter and description. See “IATXPDQ (Call the Pending Data Set Queue Manager)” on page 387.
• IATXERCV - Changed macro parameter and description. See “IATXERCV (Correct Chaining FDBs)” on page 290.

This document contains terminology, maintenance, and editorial changes, including changes to improve consistency and retrievability.
Chapter 1. JES3 Dynamic Support Programs and Installation Exit Routines

Introduction

This topic describes how to write dynamic support programs (DSPs) and installation exit routines to customize your JES3 subsystem. Installation Exits describes the installation exits supplied with your JES3 system. Introduction describes the JES3 macros.

JES3 is a general job entry subsystem of MVS™ and sometimes cannot satisfy all installation-specific needs at a given installation. If you modify JES3 code to accomplish your specific functions, you then are susceptible to the migration and maintenance implications that result from installing new versions of JES3. JES3 exits allow you to modify JES3 processing without directly affecting JES3 code. In this way you keep your modifications independent of JES3 code, making migration to new JES3 versions easier and making maintenance less troublesome.

CAUTION:

You should not view the JES3 exits defined by IBM® as unchanging; however, IBM will attempt to make any needed changes as transparent as possible to the user.

Only an experienced system programmer should attempt to make use of the JES3 exits. The writing of an exit routine and the installation of a new exit require a thorough knowledge of system programming and of JES3 programming conventions. If, without having this knowledge, you attempt to write exit routines or to install new exit points, you run the risk both of seriously degrading the performance of your system and causing complete system failure.

Therefore, defining exits and writing exit routines is intended to be accomplished primarily by experienced system programmers; the reader is assumed to have an advanced knowledge of JES3.

In certain areas of JES3 processing, you may still want to customize JES3 to a greater extent than that made possible by the standard options, JES3 initialization statements and their parameters, JES3 operator commands, JES3 control statements, and JES3 exits. While you are not constrained from making more extensive changes by directly altering the JES3 source code, direct alteration can eventually result in problems. For instance, you would have to adapt IBM maintenance code before applying it to your altered JES3 source code and, during migration, there would be no simple way of transferring your alterations to a new release. Alteration of JES3 code is not recommended.

If you intend to modify an IBM-supplied JES3 module, take into consideration the addressing mode and residency mode requirements of the module before making your change.

All JES3 installation exits and JES3 modules, data areas (with very few exceptions) reside in virtual storage above 16 megabytes and execute in 31-bit addressing mode. This book describes coding techniques that assume 31-bit addressing mode.
to be the rule and not the exception. When certain segments of code need to refer to data below 16 megabytes, this book will clearly state so.

Note that JES3 control blocks and interfaces, and, where applicable, the converter/interpreter text for converted JCL, are subject to change.

Options for Executing User-written Programs

Four options are available for executing user-written programs. These options are:

1. Executing the program as a normal MVS job. In this case, you have the greatest control over the JCL for the job, and the job is simple to submit. On the other hand, you must submit the job as a batch job using TSO submit or through a reader.

2. Executing the program as an MVS started task. In this case, the program is simple to call from a console. On the other hand, the process of overriding the JCL may be cumbersome. You must ensure that the program’s JCL resides in a system procedure library.

3. Executing the program as a JES3 DSP. In this case, the program uses JES3 functions, and JES3 executes the program directly. However, the program must operate within the JES3 environment, and a failing program might affect JES3. The following sections of this topic fully explain this third option.

4. Writing your program as an exit routine so that its purpose matches the intent of one of the IBM-supplied installation exit locations in JES3 code. In this case, your program will execute whenever JES3 processing invokes the particular exit.

In the first two cases, a failing job or procedure would abnormally end and would not affect system performance. However, in the third and fourth cases, a failure might affect JES3, which means that you should make it a practice to establish a JESTAE recovery environment. Also, because the DSP would compete for resources with JES3, it could reduce throughput in your complex.

From an overall system viewpoint, method 1 is the preferred way of executing user code. From a JES3 modification viewpoint, method 4 is the preferred way to add user code. Methods 2 and 3 require modification of IBM code and should be avoided if possible.

The conventions you need to follow to write and install dynamic support programs and installation exit routines are described in the rest of these topics.

User-written Dynamic Support Programs

JES3 dynamic support programs (DSPs) control JES3 processing. Some primary DSPs (such as MAIN) directly relate to a job’s execution. Other DSPs (such as Dump Job and Disk Reader DSPs) provide functions or services that can be called by a system operator.

In addition to the JES3 DSPs, you can create your own DSPs and add them to the system. Such DSPs run enabled in supervisor state, and under protection key 1. They become part of JES3 and JES3 expects them to use the same programming conventions as JES3 DSPs supplied by IBM.
Assembling and Link-editing a DSP

To assemble your own DSP after having written your code, submit it as you would any program to be assembled. If, in addition to adding your own code, you need to change existing source statements in a JES3 module, use the System Modification Program. The IBM-supplied JES3 source statements used to create JES3 modules reside in the SYS1.SIATSRC data set. SMP will invoke the MVS utility program, IEBUPDTE, to move and update the module into a scratch data set before assembly. To learn how to use SMP, see z/OS System Modification Program (SMP) System Programmer’s Guide. An example of the typical SMP control statements for adding your own code to a JES3 installation exit module is shown in Figure 1 on page 4.

Any program running in a z/OS environment must be assembled using the High Level Assembler with the z/OS MVS macro libraries.

All JES3 macro instructions reside in the SYS1.SIATMAC data set. During assembly, the MVS macro library SYS1.MACLIB must be concatenated with SYS1.SIATMAC in the user’s JCL to provide access to all macros.

If you have additional macros of your own, be sure those macros match the proper level of JES3. Mismatches in macro libraries can easily result in an incorrect level of a macro being included in a module.

After you have assembled your module, you must get it into a proper library for execution using a linkage editor. All JES3 program modules reside in the SYS1.SIATLIB, SYS1.SIATLPA, and SYS1.SIATLINK data sets. You must link-edit your DSP module into one of these libraries.

Any program that you write that has explicit AMODE or RMODE attributes must be link-edited with the z/OS DFSMSdfp™ linkage editor.

Note that the sample shown in Figure 1 on page 4 is merely an example of how to use SMP to add code to one of the JES3 installation exit modules. The Assembler code is not real. The SMP statements are the ones you could use for making this change to your system. However, there are other SMP statements that you could also add. Consult the appropriate SMP books to learn the proper use of SMP before using SMP to add or change JES3 installation exit routines.
Adding a DSP to JES3

Before your DSP can be executed (after you've assembled it and link-edited it), you need to include the name of the DSP in the DSP dictionary. To do so, add one or more of each of the following two macros in module IATGRPT:

- **IATYDSD** – Generates one entry for the DSP in the DSP dictionary.
- **IATYFCD** – Generates an entry in the permanent function control table (FCT).

You need one IATYFCD macro for each FCT entry you want.

Figure 1. Example of Using SMP to Implement a JES3 Installation Exit
Note: If you want the DSP to have a resident FCT, include the IATYFCD macro in IATGRPT, but do not include IATYDSD.

If you are updating the DSP dictionary, add the new names to the end of the dictionary and perform a warm start or a hot start. If you add new DSP names to some position other than the end of the DSP dictionary, you must cold start the JES3 system because the DSP numbers reside in control blocks in spool data sets.

Notes:
1. Changing the priority of an existing JES3 FCT can affect implied serialization between DSPs and thereby cause unpredictable results.
2. If maintenance is applied that adds a DSP dictionary entry to IATGRPT, you must insure that any previously added user entries are above the entry being added by the maintenance. It is important that your user-defined entries maintain the same position in the DSP dictionary. This will insure that any jobs that are to be scheduled for a user DSP number will be processed by the correct DSP.

The IATYDSP macro maps an entry in the DSP dictionary, and also an entry in the device requirements table (DRT). Each entry in the DSP dictionary contains such information as:
• The program name
• The internally assigned DSP number
• Priority
• The callable or not-callable indicator
• The holdable or held indicators
• A pointer to the device requirements table which defines the device types required for the program
• The data CSECT name, if one is to be loaded

Following the reassembly of the DSP dictionary (in module IATGRPT or, in an FSS address space, IATGRPTF), you must relink-edit the module with the JES3 nucleus (IATNUC) or FSS nucleus (IATNUCF).

Starting and Stopping a DSP
After you’ve assembled and link-edited your DSP, and have identified it to JES3, the DSP is ready to be executed.

For callable DSPs, use the following operating sequence:
1. Issue the *CALL command to make the DSP available to the system (that is, added to the FCT chain).
2. Issue the *START command to start processing the DSP.

To stop a DSP, issue the *CANCEL command.

Securing Your Installation Exits and DSPs
Your installation may enhance JES3 supplied code by implementing JES3 installation exits or by writing your own dynamic support programs. You should use the security features that protect your installation exits and installation-written DSPs.

The IATXSEC macro allows JES3 to use your installation’s security product; this macro specifies a security request and sends it to the Security Authorization Facility (SAF). SAF forwards the request to the your installation’s security product. The
security product determines if the request can be authorized and sends a response to JES3 through SAF. Figure 2 illustrates the processing for requesting processing from your installation’s security product.

Figure 2. IATXSEC Processing

Note: If your installation has RACF® 1.8 installed, RACF returns to JES3 to indicate JES3 should use its own authorization facilities to determine if the request should be authorized. If your installation has RACF 1.9 or a subsequent release installed, RACF uses its definitions to determine if the request should be authorized. JES3 may perform additional authorization processing.

You can use installation exits to change security processing in your installation.

Types of Security Calls in JES3

Table 1 describes the types of IATXSEC calls your installation can issue and the required parameters for each type of IATXSEC call.

<table>
<thead>
<tr>
<th>Value on REQUEST= Keyword</th>
<th>Reason for Issuing Request</th>
<th>Related IATXSEC Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUDIT</td>
<td>Use to create an audit record.</td>
<td>LOGSTR=Yes</td>
</tr>
<tr>
<td>AUTH</td>
<td>Use to request authority for a user, job, or DSP to use a resource.</td>
<td>ATTR=Yes, CLASS=Yes, ENTITY=Yes, LOG=No, LOGSTR=Yes, RECVR=No, RTOKEN=No, UTOKEN=No</td>
</tr>
<tr>
<td>Value on REQUEST=Keyword</td>
<td>Reason for Issuing Request</td>
<td>Related IATXSEC Parameters</td>
</tr>
<tr>
<td>--------------------------</td>
<td>----------------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>EXTRACT</td>
<td>Encrypts a token.</td>
<td>CLASS=No</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ENCRYPT=Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ENTITY=No</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FIELDS=No</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TYPE=No</td>
</tr>
<tr>
<td>TOKENBLD</td>
<td>Modifies a token if it does not have the proper authority level to access a resource.</td>
<td>GROUP=No</td>
</tr>
<tr>
<td></td>
<td></td>
<td>POE=No</td>
</tr>
<tr>
<td></td>
<td></td>
<td>REMOTE=No</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SECLABL=No</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SESSION=No</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SGROUP=No</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SNODE=No</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SUSERID=No</td>
</tr>
<tr>
<td></td>
<td></td>
<td>USERID=No</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UTOKEN=Yes</td>
</tr>
<tr>
<td>TOKENMAP</td>
<td>Converts a token from its encrypted format (external format) to the format used by RACF.</td>
<td>TOKNIN=Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TOKNOUT=Yes</td>
</tr>
<tr>
<td>TOKENXTR</td>
<td>Converts a token for various releases of RACF.</td>
<td>TOKNOUT=Yes</td>
</tr>
<tr>
<td>VERIFYX</td>
<td>Use to verify a user, job, or DSP can perform a particular task.</td>
<td>ENCRYPT=No</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GROUP=No</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JOBNAME=No</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LOGSTR=Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NEWPASS=No</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PASSCHK=No</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PASSWRD=No</td>
</tr>
<tr>
<td></td>
<td></td>
<td>POE=No</td>
</tr>
<tr>
<td></td>
<td></td>
<td>REMOTE=No</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SECLABL=No</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SESSION=No</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SGROUP=No</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SNODE=No</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SUSERID=No</td>
</tr>
<tr>
<td></td>
<td></td>
<td>USERID=No</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TOKNIN=No</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TOKNOUT=Yes</td>
</tr>
</tbody>
</table>
Issuing a Security Call in JES3

IBM has enhanced JES3’s security by adding calls to your installation’s security product. You need only to improve your installation’s security by:

1. Examining your installation’s exits and installation-written DSPs for any instances where you may want to take security precautions.
2. Using Table 1 on page 6 to determine the type request you need on the IATXSEC macro.
3. Associating a user-defined function code with the IATXSEC macro. For each type of request, IBM has reserved four user-defined function codes in IATYSSX.
4. Establishing addressability to the IATYSEC parameter list in register 1.
5. Establishing addressability to the IATYSSVT in register 0.
6. Determining the environment where you are issuing the security request to determine the value you need to specify on the MODE= keyword. If you are issuing the security request in an installation exit, use Return Codes for JES3 Installation Exits on page 58 to determine where the installation exit is running. If you are issuing the security request from a user-written DSP, specify USER on the MODE= keyword.
7. Determining which parameters you require on the IATXSEC macro based on the type of security call you made. Table 1 on page 6 identifies the optional parameters you may want to specify on the IATXSEC macro.

Installation-Written Job Validation and Job Restart Routines

This section gives an overview of how JES3 attempts to restart its processing during a restart caused by some event (such as a system failure or operator action). After the overview of JES3 restart processing, this section describes how you can tailor the JES3 restart process by providing your own job validation/restart routines to handle restarts for your own jobs.

Overview of JES3 Validation/restart Processing

During a JES3 global restart (all types except cold starts), some JES3 initialization modules examine the work (or jobs) that was processing when the system was restarted. The restart process brings the JES3 system back up to the point where jobs that were executing before the restart can continue to run.

During initialization, two JES3 functions ensure that work can restart. These two functions are called job validation and job restart.

Job validation ensures that the spool data representing the job is valid and usable. Job validation:

- Reads the spool records that represent jobs to JES3 and ensures that their contents are valid.
- Reads the job-related spool records and verifies that they belong to the individual job being validated and that their contents are valid.
- Ensures that data used by JES3 DSPs is valid.

Job restart involves restoring the job-related control blocks to a state in which the job can continue. Job restart:

- Reallocates the resources that are assigned to the job, such as the job record, the job number, and the spool space used by the job.
- Builds the control block structures used for scheduling the job and for continued execution of the job.
By validating jobs remaining in the system over a restart, JES3 makes it possible for the operator to obtain diagnostic information for incorrect jobs, to cancel any incorrect valid jobs, and to end JES3 initialization if necessary.

**Job Validation Using the Scheduler Elements**

During a restart, JES3 validates the jobs remaining in the system by processing each job individually according to the scheduler elements (SEs) that make up the job. JES3 provides all the validation routines required for your installation to validate *standard jobs* and jobs that are submitted through a card reader, TSO, or batch. JES3 supplies validation routines for scheduler elements representing these DSPs:

- Input service-related DSPs (such as, CR, TR, DR, ISDRVR, INTRDR)
- Converter/interpreter (CI)
- Main service (MAIN)
- Main device scheduling (MDS)
- Output service (OUTSERV)

**Validating Control Blocks During a JES3 Restart**

JES3 validates the following spool control blocks on any restart (except a cold start or local start) of the system:

- The job control table (JCT)
- The job description and accounting block (JDAB)
- The job management record (JMR)
- The job data set (JDS)
- The job track allocation tables (JBTATs, JBTs, or TATs)
- The data set track allocation tables (DSTATs), if any exist

In addition, during an “analysis” restart, the following job-related spool control blocks will be validated:

- The job summary tables (JSTs) and the dynamic job summary tables (DJSTs).
- The output service elements (OSEs) and work output service elements (WOSEs).
- The PROCLIB update blocks (PDBs), if any exist.
- The job volume table (JVT)
- The format parameter buffers (FRPs)

**Restarting Individual Jobs**

After validating the control blocks that are common to all jobs, JES3 examines control blocks related to particular scheduler elements associated with individual jobs. For each job that remains in the system over a restart, JES3:

1. Validates the job control table (JCT). JES3 ensures the JCT contains all the scheduler elements required for the job, and that the scheduler elements are in the proper sequence.
2. Determines if the job was active when JES3 ended. If the job was active, JES3 indicates which scheduler element was processing the job when JES3 ended. (Later, JES3 will restart the job in the active scheduler element after JES3 is restarted.)
3. Determines if the job should continue validation processing. JES3 will end validation processing for the job if:
   - The job had an incorrect SE.
   - The job was flagged for deletion by the operator.
4. Accesses the job track allocation tables and data set track allocation tables (JBT) to validate and recover the spool space allocated to the job.
5. Performs validation processing for each scheduler element in the job.

During “analysis restarts,” JES3 performs additional validation processing for individual jobs. If the job contains:
• An input service related scheduler element (IS), JES3 validates the job track allocation table (JBT) for the reader.
• A converter/interpreter scheduler element (CI) and the job is a PROCLIB-update job, JES3 validates the PROCLIB update blocks (PDB).
• An output service scheduler element (OUTSERV), JES3 validates output service scheduler elements (OSEs), work output service scheduler elements (WOSEs), and format parameter buffers (FRPs).

Writing Your Own Validation/restart Routine

If your installation uses only standard jobs, you will not need to write a validation/restart routine because JES3 supplies the routines for the five primary DSPs listed on page 9. However, if your installation makes use of user-written DSPs and you do not include a validation/restart routine to protect jobs that invoke these DSPs, then any of these jobs might be lost if JES3 is restarted in the middle of the job’s processing.

To determine if your installation creates any non-standard jobs, examine any JCL that your installation uses to submit jobs. If the JCL includes any */PROCESS statements that specify a user-written DSP, a validation/restart routine may be needed. [z/OS MVS Initialization and Tuning Guide] explains how you can tailor job processing so that not all jobs execute as standard jobs; that is, you can create non-standard jobs.

If you need to write a routine to validate the control blocks associated with a scheduler element created by your installation, you must identify your routine to JES3. To do so, you must update the definition of the scheduler element so that it specifies the validation routine. You define scheduler elements to JES3 using the IATYDSD macro in the DSP dictionary. The JOBVAL= parameter on the IATYDSD macro identifies the validation routine for the scheduler parameter element. A description of the IATYDSD macro appears in IATYDSD (Generate a DSP Dictionary Entry) on page 595.

After you write the routine for validating the scheduler element, you must assemble and link-edit it.

In addition to checking the validity of control blocks created by your own DSPs, you can also check the validity of user fields within JES3 control blocks. To do so, use JES3 installation exit 14. See “CSECT Name: IATUX14 (Validate Fields in Spool Control Blocks During a JES3 Restart)” on page 74 for more information.

Conditions that create a need for you to write a job validation/restart routine for a user-written DSP are:
• A callable DSP that remains active over a JES3 restart creates a spool-resident control block. A validation/restart routine should be written to validate the control blocks and to reallocate the spool space assigned to the control blocks.
• The DSP is scheduled as part of a job (that is, the DSP is invoked because a scheduler element for the DSP is included in the job’s JCT).
• The DSP creates its own job-related spool record (that is, its spool space is obtained from the job’s track allocation table (JBT)) and the user wishes to perform analysis of the control block during a normal JES3 restart or when an “analysis” restart is specified.
• The DSP expects the job’s RESQUEUE to contain information that is not provided by IBM-supplied code (that is, an installation exit or DSP inserts information into the job’s JCT instead of into the job’s RESQUEUE).
Your validation/restart routine should adhere to the following guidelines:

- The name of the module should follow your installation’s naming conventions. If your installation does not follow a particular naming convention, IBM suggests following the naming conventions for JES3.
- Your routine must be MVS reentrant.
- Your routine should contain a JESTAE retry routine. The retry routine should attempt to recover from an ABEND or allow percolation to the next level of JESTAE.
- Messages can be issued to the operator’s console using the IATXVMSG macro with the WTO=Y keyword. IATXVMSG queues all the messages issued during validation/restart processing and issues them when the processing finishes.

Your validation/restart routine should include a section for:

- Validation processing to ensure the control blocks associated with your installation’s scheduler element are valid.
- Restart processing if the job was executing in your installation’s scheduler element when JES3 ended.

The Validation Portion of Your Routine

The validation portion of your routine should validate any spool control blocks that are associated with your installation’s scheduler element. First determine the control blocks associated with your installation’s scheduler elements and when they should be validated. For example, should they be validated on all restarts or only on analysis restarts? Then, for each control block you need to validate:

- Issue an IATXVFDB macro to determine if you can access the control block the FDB represents. The IATXVFDB macro should include the MF=, FDB=, ID=, and DESC= parameters. JES3 creates an entry for the Summary of Spool Records Validated section of a SNAP using the information supplied on the ID= and DESC= parameters.
- Examine the return code in register 0 from IATXVFDB. If the return code is a 0 or 4, the single record file (SRF) can be validated. Otherwise, the SRF the FDB describes cannot be accessed. If the control block is valid and resides in the JES3 single track table (SST), the IATXVFDB service will reallocate the spool record.
- Issue a JESREAD macro for those SRFs that can be validated. (Before attempting to read in the SRFs, be sure that the SRFs are not already in storage as a result of a prior validation module.)
- Validate the information in the SRF. For each SRF, ensure that:
  - The SRF belongs to the job. For the job being validated, the job identifier, if any, is contained in the job validation work area macro IATYJVW, field JVVJOBID. This value should be compared against the job identified in the SRF your validation routine is examining.
  - The control block contains a valid end-of-file indicator.
  - The variable and fixed-length fields are valid and the length of the fields add up to the entire length of the control block.
  - The information that is duplicated in another control block is the same.
- Release the control blocks that were read in by this routine. If information in the control block needed to be updated, write the control block to spool using the AWRITE macro. If the control block was not updated, it can be released from storage using the ARELEASE macro.
- To record errors that occur during validation processing, use either of the following macros:
- IATXVSRE to record the control block in the Spool Record Diagnostic/Error Messages section of the SNAP output for the job.
- IATXVMSC to record the control block in the Job Validation JES3OUT/Console Messages section of the SNAP output for the job.

The Restart Portion of Your Routine

The restart portion of your validation/restart routine should recreate the control blocks that are required to process the job, only if the job was active in your installation’s scheduler element. To determine if the job was active, you can check two fields in the JVW. The first field is JVWACTSE, which contains the address of the active JCT scheduler element entry. The second field is JVWACTDE, which contains the address of the DSP dictionary entry for the active DSP.

The restart processing in your validation/restart routine should:
- Recreate the control blocks required by your installation’s scheduler element to process the job
- Rebuild the RESQUEUE for the job.
- Indicate in the job’s RESQUEUE if additional restart processing is required. The additional restart processing will be invoked when JSS is started. Additional restart processing includes processing that cannot be performed until the JES3 initialization tables are built (that is, after phase 3 of initialization).

Note: If restart processing is not included in the validation/restart routine and restart processing is required for the job, JES3 rebuilds only the RESQUEUE and attempts to process the job.

Programming Considerations for Your Routine

The following information should be considered when writing a job validation/restart routine:

Environment: A user-written validation/restart module will be invoked if both of the following conditions are met:

1. The DSP dictionary entry or the function control table (FCT) entry (in module IATGRPRT) for the user-written DSP specifies the JOBVAL= keyword (this keyword specifies the name of the job validation/restart module to be invoked during initialization when a job contains a scheduler element for the DSP). See the descriptions of the IATYDSD and IATYFCD macros in "Introduction" on page 169 more information.

2. A job that is being restarted contains a scheduler element for the user-written DSP.

Linkage Conventions: The job validation/restart module will be invoked using ACALL from module IATJVDR for each job that contains a scheduler element for the DSP. The validation module must return to JES3 using the ARETURN macro.

Information Supplied on Entry: The address of the validation FCT is passed in register 11.

The address of the job validation data csect (macro IATYJVD) is passed in register 13. The JVD is used to pass information between the job validation driver (IATJVDR) and the validation/restart modules. This information includes:
- The address of the job validation work area (JVW) representing the job that JES3 is currently validating. The JVW contains: the status information about the job, a copy of the job’s JCT, pointers to the job’s JDAB and JMR in storage, and data
which will be used to “snap” information about the job. The address of the job’s
JVW is contained in field JVDJVWAD of the JVD.

- The addresses of the routines for validation and data collection (macros
  IATXVFDB, IATXVMSG, IATXVSRE, and IATXVSRV).

**Information Supplied on Exit:** The contents of the registers on return to IATJVDR
are irrelevant. If the validation/restart module is responsible for building the
RESQUEUE for the job (this occurs if the job was active in the scheduler element
when the JES3 system went down), the RESQUEUE must be chained off the job’s
JVW (JVWRESQ). The RESQUEUE must contain any DSP-specific information
which may be required to restart the job, if any.

The JVW will also contain indicators specifying whether the job completed validation
successfully or not (JVWFAIL in JVWSTA1 indicates that an ABEND occurred
processing the job).

Any information that is saved during the processing of the job (using the
IATXVMSG or IATXVSRE macros) will be retained and “snapped” with all the
information about the job if the operator responds “SNAP” to message IAT4174.

**Example of the Use of Some Fields in Validation**

[Figure 3 on page 14] shows a list of the fields that are most involved in job
validation. Following the list are some examples of code showing the use of the
fields.
JW fields to be used by a user-written validation/restart routine.

<table>
<thead>
<tr>
<th>Field</th>
<th>Size</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>JVWWORKD</td>
<td>8</td>
<td>Doubleword work area that can be used as a temporary save area</td>
</tr>
<tr>
<td></td>
<td></td>
<td>or as a conversion work area</td>
</tr>
<tr>
<td>JVWCONVD</td>
<td>8</td>
<td>Doubleword work area that can be used as a temporary save area</td>
</tr>
<tr>
<td></td>
<td></td>
<td>or as a conversion work area</td>
</tr>
<tr>
<td>JVWWORK</td>
<td>4</td>
<td>Fullword work area that can be used as a temporary save area</td>
</tr>
<tr>
<td></td>
<td></td>
<td>or as a conversion work area</td>
</tr>
<tr>
<td>JVWJOBNM</td>
<td>8</td>
<td>The job name</td>
</tr>
<tr>
<td>JVWJOBD</td>
<td>8</td>
<td>The job identifier</td>
</tr>
<tr>
<td>JVWICT</td>
<td>4</td>
<td>The address of the job's JCT</td>
</tr>
<tr>
<td>JVWSTA1-</td>
<td>1</td>
<td>Job status indicators</td>
</tr>
<tr>
<td>JWSTAB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>JVWWKFDB</td>
<td>12</td>
<td>A work FDB - to be used to read in a spool control block</td>
</tr>
<tr>
<td>JVWWCHFDB</td>
<td>12</td>
<td>A work FDB - to be used as a save area to the next FDB of a chain of FDBs</td>
</tr>
<tr>
<td>JVWWVFDB</td>
<td>16</td>
<td>The list form of the IATXVFDB macro that may be used with the execute form of the IATXVFDB or IATXVSRV macros</td>
</tr>
<tr>
<td>JVWWTO</td>
<td>124</td>
<td>The list form of the WTO macro that may be used with the IATXVSRE and IATXVMSG macros</td>
</tr>
</tbody>
</table>

Figure 3. Fields Used in Job Validation

THE FOLLOWING CODE SAMPLE SHOWS THE USES OF SOME FIELDS:

```
L R7,JVDJWAD          ADDRESS THE JOB'S JVW
USING JWSTART,R7

* THE FDB FOR MY CONTROL BLOCK (MYCB) IS IN THE JCT.
* L R3,JVWJCT          ADDRESS of THE JCT
USING JCTSTART,R3
MVC JVWWKFDB,JCTMYFDB COPY THE FDB
DROP R3              RELEASE REGISTER 3

* VALIDATE THE FDB
* IATXVFDB FDB=JVWWKFDB, VALIDATE THE FDB
  ID=MYCB,  - CONTROL BLOCK ID
  ROOT=JVWJCFDB,  - ROOT FDB IS THE JCT
  DESC=DESCRIP,  - CONTROL BLOCK DESCRIPTOR
  ERROR=OUT,    - ERROR RETURN
  NAVAIL=OUT,   - UNAVAIL RETURN
  MF=(E,JVWWFDB)  - EXECUTE FORM
```
* READ THE CONTROL BLOCK INTO STORAGE
*  
* JESREAD FDB=JVWWKFDB, READ THE CONTROL BLOCK
*     ID=MYCB, CONTROL BLOCK ID
*     ERROR=ERRMYCB ERROR RETURN
*     L R2,JVWWKFDB GET THE CONTROL BLOCK ADDRESS
*     USING MYCBSTRT,R2
*  
* VALIDATION OF PARTICULAR FIELDS WITH MYCB IS DONE HERE.
*  
* add code here to validate MYCB fields
*  
* IF MYCB IS UPDATED, THEN WRITE MYCB BACK TO SPOOL
*  
* label1 DS OH
*     AWRITE FDB=JVWWKFDB, WRITE THE CB TO SPOOL
*     ID=MYCB, CONTROL BLOCK ID
*     ERROR=ERRMYCB ERROR RETURN
*     B OUT LEAVE
*  
* RELEASE THE FDB
*  
* label2 DS OH
*     DROP R2 RELEASE REGISTER 2
*     ARELEASE FDB=JVWWKFDB RELEASE THE SPOOL BUFFER
*     B OUT LEAVE
*  
* IF ERRORS
*  
* ERRMYCB DS OH
*     TM JVWWKFDB+FDOSPFL1-FDBSTART,FDBONSP CB ON SPOOL?
*     BC ALLON,SKIP BIY
*     ARELEASE FDB=JVWWKFDB RELEASE THE SPOOL BUFFER
*  
* COPY THE MESSAGE INTO THE WTO WORK AREA AND ISSUE IATXVSRE
*  
* SKIP DS OH
*     XC JVWTO(JVWWTOL),JVWTO CLEAR THE WTO AREA
*     MVC JVWTO(DSP001L),DSP001 COPY MODEL MESSAGE
*     IATXVSRE JVWTO MARK THE JOB FOR DELETION
*     FDB=JVWWKFDB - FDB IN ERROR
*     B OUT LEAVE
*     DROP R7
*  
* RETURN TO THE CALLER
*  
* OUT DS OH ARETURN ,
*  
* CONSTANTS AND LIST FORM OF WTO'S
*  
* DESCRIP DC CL(SRVDESCL)'MY CONTROL BLOCK'
* DSP001 WTO 'MY CONTROL BLOCK (MYCB) FAILED A JESREAD',
*    MF=L
* DSP001L EQU *=DSP001
Installation Exit Routines

Installation exits enable an installation to tailor JES3 without having to modify JES3 code. By not modifying JES3 code, you reduce the amount of work necessary to install JES3 maintenance or new JES3 function. Thus, installation exit routines can be very useful in helping you develop an operating system tailored to your needs.

IBM supplies a module for each JES3 installation exit. (Some exits perform a default function, but most are dummy modules). You may, of course, write your own installation exit routine in place of any provided by IBM. For each exit, either the IBM-supplied routine or an installation exit routine must reside in the JES3 module library, or JES3 issues warning messages IAT3020 and IAT3102.

See [z/OS JES3 Initialization and Tuning Reference](https://www.ibm.com) for information on how the installation exit routines influence JES3 job management.

All of the IBM-supplied exit routines (except exit 26) specify AMODE 31 and RMODE=ANY. (IATUX26 specifies AMODE 31 but RMODE=24 meaning that it is loaded below 16 megabytes.) All exit routines are entered in 31-bit addressing mode. The exit modules are loaded above the 16-megabyte address.

As with DSPs, you must assemble your installation exit routine using the High Level Assembler with the z/OS macro libraries at the current release. Then you must link-edit your exit routine into a JES3 library replacing the exit routine provided by IBM. Any program that you write that has explicit AMODE or RMODE attributes must be link-edited with the z/OS Binder.

Sample Command Exit (IATSMX01)

Module IATSMX01 provides a sample command exit that converts a JES3 *SEND command into an MVS ROUTE command. Use this exit to assist your conversion from using the form of the JES3 *SEND command which is no longer supported.

If you install and activate this exit, it receives control and examines every command that executes on the system. If IATSMX01 detects a JES3 *SEND command with a target of a system name within the JES3 complex, it converts the command into an MVS ROUTE command. To install and activate this exit:

1. Assemble the exit with AMODE 31 and the RENT assembler option.
2. Linkedit the exit into an AFP-authorized library that is part of the LNKLST concatenation. Use the LNKLST parameters of RENT AMODE(31) RMODE(ANY)
3. Specify the exit name, IATSMX01, on the USEREXIT parameter of the .CMD statement in an MPFLSTxx member of SYS1.PARMLIB:
   ```
   MPFLSTxx
     ...
     .CMD USEREXIT(IATSMX01)
     ...
   ```
4. Activate the MPFLSTxx member either through IPL parameters or issue the MVS command SET MPF=xx
Sample MPF Exit (IATSMX02) — JES3 IAT6850 Message Processing

Module IATSMX02 provides a sample MPF exit that receives control for message IAT6850. You can use this exit to take action based on the type of job (TSO, STC or batch), the job’s jobname, or how many times the job exceeded the limit.

JES3 monitors WTO rates for address spaces that are known to JES3. When an address space’s WTO rate is exceeded, JES3 issues the IAT6850 message as illustrated below:

IAT6850 TSU VAIN (JOB16253) ON SY1 IN 001E ...REACHED WTO LIMIT
(  20000)  2 TIMES IN  22 SECS

When this sample exit gets control for the IAT6850 message, it does a table lookup for an entry that matches the address space type (STC, TSU, or JOB) and the jobname in the message. If a matching table entry is found, the address space’s WTO count is computed as the product of the message’s LIMIT and TIMES values. The computed WTO count is compared with the table entry’s WTO action threshold value and an action operator command is issued if the WTO count exceeds the table entry’s threshold.

The Excessive WTO Action Table is defined after the label ACTAB. The table has five columns:

1. Type - recognized values STC, TSU, JOB, or *. An '*' matches anything on IAT6850 messages type field.
2. Compare Length of Jobname - specifies how many characters of the jobname on the IAT6850 are used for matching this table entry.
3. Jobname - specifies the jobnames that match this table entry. An '*' matches anything.
4. WTO Limit - specifies the number of WTOs that the product of IAT6850 message’s LIMIT and TIMES values has to exceed in order for the action to take place. Note that table limit value can be higher than the JES3 excessive WTO limit. This allows the JES3 monitoring to run with low values such that advance warning messages are displayed before the action is taken. Note also that the product of IAT6850 message’s LIMIT and TIMES is always computed from the current message.
5. Action - specifies the operator command to be issued:
   - CAN - issues CANCEL (U=)jobname,DUMP,A=asid command.
   - RES - issues RESET jobname,A=asid,QUIESCE command. The RESET command assumes that the system is running in WLM goal mode.
   - MOD - issues *MODIFY J=jobnbr,NOLOG command to stop WTO logging into JESMSGLG data set.

Note: The table is searched from top to bottom and the first matching entry is used. An entry with type=’*’ and jobname=’*’ matches everything.

If you plan to use this sample MPF exit, you should customize the ACTAB to meet your installation requirements. The following steps should be followed to install and activate this exit:

1. Assemble with AMODE 31 and the RENT assembler option.
2. Linkedit the exit into an APF-authorized library that is part of the LNKLST concatenation. Use the following linkedit parms -
   - RENT AMODE(31) RMODE(ANY)
3. Specify the name of the exit on the USEREXIT parameter of the messages processing record (MSGID) statement in an MPFLSTxx member of SYS1.PARMLIB:

IAT6850,USEREXIT(IATSMX02),AUTO(YES),SUP(NO)

4. Activate the MPFLSTxx member either through IPL parameters or with the SET MPF=xx command.

Determining the JES3 Release Level

Other code, whether other IBM program product code, Solution Developer code, or installation-written code might need to determine what level of JES3 is installed. This can be important so that such code can determine what support is required within that code or what support JES3 provides for a particular release. The JES3-provided global assembler variable &J3VRSN in mapping macro IATYGLOB provides this indication; to access this variable you must add an assembler "COPY IATYGLOB" statement to your assembler code.

JES3 releases before OS/390® 1.1.0 set a &J3VRSN symbol of 'SP v.r.m'. OS/390 1.1.0 set &J3VRSN to 'SP 5.3.0'. This symbol and value is being maintained for compatibility in case products outside of JES3 were using this value. Consider this value stable and not to be changed or incremented in the future.

To accommodate future JES3 releases, use the following assembly-time variables:

**Variable in IATYGLOB**

**Description and Use**

**&J3LEVEL**

- **Value**: z/OS releases have a value in the form of 'z v.r.m' where:
  - v is the z/OS Version
  - r is the z/OS Release
  - m is the z/OS Modification Level
  For example, &J3LEVEL for z/OS Version 1, Release 2, Modification Level 0 would be ‘z 1.2.0’.
- **Description**: 8-byte string defined as is &J3VRSN;
- **Note**: The format of this field is an 8-byte EBCDIC string; however, do not rely upon the string data for release-to-release comparisons.

**&J3PLVL**

- **Value**: A numeric value that increases by 1 for each successive JES3 release.
- **Description**: A value that corresponds to a specific JES3 product release level as follows:

  **Table 2. JES3-Provided Global Assembler Variable (&J3PLVL) for Currently Supported JES3 Releases**

<table>
<thead>
<tr>
<th>JES3 Version and Release</th>
<th>J3PLVL Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS/390 2.6.0</td>
<td>&amp;HJS6606_PLEVEL (10)</td>
</tr>
<tr>
<td>OS/390 2.8.0</td>
<td>&amp;HJS6608_PLEVEL (11)</td>
</tr>
<tr>
<td>OS/390 2.9.0</td>
<td>&amp;HJS6609_PLEVEL (12)</td>
</tr>
<tr>
<td>OS/390 2.10.0</td>
<td>&amp;HJS7703_PLEVEL (13)</td>
</tr>
<tr>
<td>z/OS 1.2.0</td>
<td>&amp;HJS7705_PLEVEL (14)</td>
</tr>
</tbody>
</table>
Table 2. JES3-Provided Global Assembler Variable (&J3PLVL) for Currently Supported JES3 Releases  (continued)

<table>
<thead>
<tr>
<th>JES3 Version and Release</th>
<th>J3PLVL Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>z/OS 1.4.0</td>
<td>&amp;HJS7707_PLEVEL (15)</td>
</tr>
<tr>
<td>z/OS 1.5.0</td>
<td>&amp;HJS7708_PLEVEL (16)</td>
</tr>
<tr>
<td>z/OS 1.7.0</td>
<td>&amp;HJS7720_PLEVEL (17)</td>
</tr>
<tr>
<td>z/OS 1.8.0</td>
<td>&amp;HJS7730_PLEVEL (18)</td>
</tr>
</tbody>
</table>

- **Note:** The value itself has no inherent meaning.

**&J3SLVL**

- **Value:** A numeric value that increases by 1 for each successive JES3 APAR that provides significant service (for example, service that requires multisystem awareness of the presence or absence of this service within a JES3 complex).
- **Description:** A service level within the product level updated for significant JES3 updates.
- **Note:** This value itself has no special meaning.

**&J3REL**

- **Value:** v.r.m
- **Description:** Alternate form of &J3LEVEL; Where &J3LEVEL is in the form of zv.r.m, &J3REL is in the form of v.r.m
- **Note:** The value itself has no special meaning.

**&J3FMID**

- **Value:** HJSxxxx
- **Description:** The FMID of the current JES3 release. For example, z/OS Release 1.2.0 has the FMID value of HJS7705.
- **Note:** The value itself has no special meaning.

Programming Conventions for Writing DSPs and Installation Exits

This section describes some general programming conventions that you should follow when writing DSPs and installation exit routines. The conventions described are the same as those used by JES3 modules. Failing to observe some of the conventions can introduce failures in JES3 or in the installation exit code. Using JES3 macro instructions implicitly maintains most JES3 conventions.

Most JES3 installation exits and JES3 modules, data areas, and such., reside in virtual storage above 16 megabytes and execute in 31-bit addressing mode. This book describes coding techniques that assume 31-bit addressing mode to be the rule and not the exception. When certain segments of code need to refer to data below 16 megabytes, this book will clearly state so. For more information on writing code to run in 31-bit addressing mode see:

- [z/OS Migration](https://www.ibm.com/downloads/collationalibrary/OS390MIGRATION.pdf)

The list of conventions begins following Figure 4 on page 20.

The example in Figure 4 on page 20 shows the basic instructions, in their proper order, that a module should contain when used with JES3.
AMODE (Indicates if the module uses 24- or 31-bit addresses)
RMODE (Determines where the module will reside)
IATYASM (Tells MVS to use the OS/390 R1 version of MVS macros)

[Prolog for module goes here]

COPY IATYGLOB goes here (copies global symbols)

[Data DSECT macros go here]

CSECT (Entry point for module)

IATYMOD (Generates module identification information)

[Executable code begins here]

[APARNUM and PTFNUM are labels used to define the release level.]
APARNUM (7 bytes long)
PTFNUM (7 bytes long)

Figure 4. Order of Code Instructions in Typical JES3 Module

1. All modules must be reenterable.
2. Use an AMODE instruction to let the MVS system know the mode your module will run in. For DSPs and installation exits this should be AMODE 31 because they are called in 31-bit addressing mode.
3. Use an RMODE instruction to define where your module should reside. For JES3, this is normally RMODE ANY.
   You should specify RMODE ANY unless your module specifies AMODE 24 or your module has to switch to 24-bit mode during execution or your module contains data areas that must be below 16 megabytes, in which case you must specify RMODE 24.
   Note, if you use RMODE ANY, you must also specify AMODE 31.
4. If a module contains more than one CSECT, you must include an AMODE and an RMODE statement in the module for every CSECT in the module. This includes data CSECTs generated from macros.
5. All modules must issue the IATYASM macro before invoking any other JES3 or MVS macros. The IATYASM macro ensures that the module obtains the appropriate macro expansions. You should code the IATYASM macro immediately after the AMODE and RMODE statements in every JES3 module. By convention, these statements should be placed at the beginning of a module before the prologue. You should have a "COPY IATYGLOB" statement after the prolog to copy global symbols defined by JES3.
6. Because JES3 searches for a name in a module, you must include an IATYMOD macro in any installation exit module. When a module contains the IATYMOD macro, JES3 will insert the module name in dump output. This can be very helpful when browsing a dump during problem analysis. For more information on the IATYMOD macro, see its description.
7. All modules must contain a label called APARNUM and a constant called PTFNUM for reference by the IATYMOD macro expansion. APARNUM and
PTFNUM should each be defined as CL7'nnnnn'. These fields should identify the service level of a module and they will appear in dump output.

**Note:** If you want the PTF number to be the same as the JES3 release name, add a "COPY IATYGLOB" statement after the prolog and code the PTFNUM field as follows:

```
PTFNUM DC CL7'&J3REL'
```

8. MVS routines that either implicitly or explicitly use the MVS WAIT macro should be invoked only by using the generalized subtask facility. For more information on using WAIT macros, see section "Restrictions on MVS Services Available to DSPs and Installation Exit Routines" on page 27.

9. DSPs and installation exits must never cause the JES3 system to end abnormally.

10. If an error occurs, a user-written DSP or installation exit must:
    - Clean up all resources allocated in its behalf
    - Close all open files
    - Release all allocated JSAM buffers

    The DSP or installation exit should carry out these tasks by using JESTAE routines or by using the error return on any JES3 macros.

11. Whenever a service routine may be invoked using a macro, use the macro.

12. Be sure all MVS and JES3 services (such as macros) that your code uses are invoked in the right addressing mode and that all parameters and data areas passed have the right residence. (For example, the IATXLP macro can be issued only in 24-bit mode and its parameters and buffers must all be below the 16-megabyte address; EXCP can be issued in either mode, but the IOB, channel programs, and buffers must be below 16 megabytes in virtual storage).

    When you write an installation exit routine or a DSP which runs in 31-bit addressing mode, there may be instances when your routine will need to switch to 24-bit addressing mode. The most likely instance is that your routine uses a macro that calls a restricted MVS service which requires 24-bit addressing mode. Your routine must switch into 24-bit addressing mode before calling the service. Your routine must reinstate 31-bit addressing mode before returning to the module that called your routine.

    Try to make it a practice to switch addressing modes within your routine only to use restricted services that have to be called in 24-bit addressing mode.

    If a service requires one or more of its parameters to be in virtual storage below 16 megabytes, then you must insure that the data area is obtained below 16 megabytes. If the parameter is in your module (such as an inline parameter list generated by a macro or a DCB macro coded in your module), then you must specify RMODE 24 for the module. Note that this does not affect the addressing mode that can be used in the module. It only causes the module to be loaded below 16 megabytes.

    If the parameter is to be obtained dynamically using GETMAIN or AGETMAIN, the LOC= parameter may be used to specify where to obtain the storage. Note that an AMODE/RMODE 31 module can use the LOC=BELOW parameter.

    Unless indicated otherwise in "Introduction" on page 169 all services can be used in 31-bit addressing mode, and all parameters may reside above 16 megabytes and point to locations above 16 megabytes.

13. DSPs must perform all unit record and tape I/O using the JESOPEN, JESEXCP, and JESCLOSE macros. DSPs and installation exit routines operating in a C/I FSS address space cannot perform unit record and tape I/O.
14. DSPs, except DSPs running in a C/I FSS address space, must issue a LOGIN macro for operator communication and must always respond to console messages. DSPs running in a C/I FSS address space must issue a LOGIN macro to define the entry point to a console appendage routine but do not communicate directly with the operator. Instead, the CIDRVR FCT receives the operator commands and routes them to the C/I FSS address space using the MVS functional subsystem intercommunication (FSI). CI DSPs can issue WTO and WTOR macros through the C/I subtask.

15. All routines should reference JES3 control blocks using named fields in JES3 DSECTs. Routines should not make references using an absolute displacement because the displacements of fields are subject to change.

16. Because the transfer vector table (TVT) and the function control table (FCT) are the primary communication links between the DSPs and JES3, every module that runs in either:
   • The JES3 global or local address space under the JES3 primary or auxiliary task
   • A C/I FSS address space
must include the IATYTVT macro and the IATYFCT macro. The IATYTVT macro establishes the DSECT for the TVT and includes the USING assembler statement for the TVT. The IATYFCT macro maps the fields of the entry for your DSP in the FCT.

17. Dependencies exist between many JES3 mapping macros. Many of these dependencies are resolved by including the required macros in the dependent macro. This technique cannot be used to resolve dependencies in all cases. In cases where another macro is required and it cannot be included in the dependent macro, the prolog of the dependent macro will indicate which macros must be included by the programmer using the macro.

18. If the DSP performs a service on behalf of a specific job in the system, the IATXJQE macro can be used to access job-related control blocks; for example, use the IATXJQE macro to access the RESQUEUE (IATYRSQ). If the DSP obtains the JQE, the JQEFREEZ flag in JQEFLG1 should be tested. If the JQEFREEZ flag is on, the job is going through purge processing, and the DSP should bypass this job.

19. Do not use the BAKR, PC, or PR assembler instruction in any code that issues an AWAIT macro or calls a JES3 service that issues the AWAIT macro. Instead, use the ACALL and ARETURN macros. For more information on the ACALL and ARETURN macros, see “ASAVE Linkage and Save Area Use” on page 26.

Additional programming conventions, such as register use and linkage conventions, are explained below.

Further information on using exit routines see “Installation Exits” on page 58.

Register Usage

This discussion of register use is comprised of three sections: contents of registers on entry to a DSP, register conventions, and contents of registers on exit from a DSP.

Register usage for installation exits is defined for each exit. In general, registers 11, 12, 14, and 15 on entry are as defined below but other input registers are also provided. Also, most installation exits use the same register conventions.
Registers on Entry to a DSP
JES3 stores the following data in the registers before passing control to a DSP:

- Registers 0 through 10 are undefined.
- Register 11 contains the address of the function control table (FCT) for the DSP.
- Register 12 contains the address of the JES3 transfer vector table (TVT).
- Register 13 is the base register of the DSP’s data CSECT, if one is defined for the DSP.
- Register 14 is the address to which a called routine must return.
- Register 15 is the entry point address of the called program.

Register Conventions
JES3 DSPs use the following register conventions that all installation exit code for JES3 should also use. Exception: JES3 installation exits that run outside of the JES3 address space may follow other register conventions (for example, IATUX26 and IATUX57). In these cases, check the installation exit for register conventions.

- Registers 0, 1, and 2 should be reserved for passing parameters when invoking JES3 services.
- Registers 3 through 9 are used as work registers.
- Register 10 is the base register for all DSPs. Register 9 is the recommended second base register if one is required.
- Register 11 contains the address of the function control table (FCT) for the DSP. The multifunction monitor depends on the correct setting of this register at all times to maintain control of the system. A user-written DSP must never alter this register.
- Register 12 contains the address of the JES3 transfer vector table (TVT). A user-written DSP must never alter this register.
- Register 13 is the base register of the DSP’s work area or data CSECT.
- Register 14 is the address to which a called routine must return.
- Register 15 is the address of the called program. When control is returned to the calling program, register 15 contains the return code.

Contents of Registers on Exit from a DSP
On exit from a DSP, JES3 requires only that registers 11, 12, and 14 are the same as on entry to the DSP, and that register 15 contains a valid return code.

DSP Initialization and Termination Requirements
The code you write for your DSP must perform certain functions for the DSP to work properly within JES3. These functions are described here according to DSP initialization and DSP termination.

DSP Initialization
The following list represents the recommended standard sequence of required tasks for the initialization phase of a user-written DSP, which the job segment scheduler (JSS) will schedule. Use the list only as a general guide because, at times, you can omit steps or vary the sequence to reflect particular processing requirements.

As a minimum, perform step 1. The other steps depend on whether the DSP needs a job description accounting block (JDAB) and job data set (JDS) information.

1. Establish a base register for the DSP. By convention, DSPs use register 10 as the standard base register. Although another register can be used, using the standard base register convention will ease program analysis.
2. Issue the LOGIN macro to establish the means by which the operator can communicate with the DSP.
3. Establish the JESTAE environment.
4. Issue the IATXCNS macro specifying TYPE=GET. On return, register 1 contains the address of the parameter buffer for the job.
5. Extract the parameter information from the parameter buffer in the previous step.
6. Issue the IATXCNS macro specifying TYPE=RELEASE to free the parameter buffer.
7. Issue the GETUNIT macro to request any required JES3 support devices. If the required devices are unavailable, cancel or request specialized rescheduling for the unavailable devices. The system programmer must specify the type of devices required by the DSP in the device requirements table (DRT). The DRT is located within the DSP dictionary and is built using the REQ parameter on the IATYDSD macro. The system programmer must fill in the GETUNIT LIST (GLIST) before the DSP issues the GETUNIT macro.
8. Issue the JDSGET macro to get the JDS, and extract FDBs for the data sets to be processed as input, if any. The JDS contains an FDB and data set name for each SYSOUT data set or data set entered through the input stream (SYSIN) that is associated with the job.
9. Issue the JDSREL macro to release the JDS.
10. Issue a signon message. When the DSP has verified its parameters and has successfully obtained its devices, it can issue a signon message to notify the operator that the DSP is active.

**Note:** Any part of a DSP that executes as a subtask must not issue any JES3 ASAVE-protected macros (for example, JESREAD or MESSAGE).

**DSP Termination**
Before completing, the code in your DSP must do certain functions. The following list represents the recommended standard sequence of required tasks for the termination phase of a DSP. Use the list only as a general guide because, at times, you may omit steps or vary the sequence to reflect particular processing requirements.

1. Issue the ACLOSE macro for all open multirecord files.
2. Issue the JDSGET macro for the JDS. Issue the JESREAD macro for the JDAB.
3. Update the JDS, using the JDSADD macro, for all new JDS entries. If the DSP modifies existing JDS entries, use the JDSPUT macro.
4. Issue the JDSREL macro to release the JDS.
5. Update the JDAB. Required accounting information, such as lines printed or cards punched, should be posted in the corresponding JDAB scheduler element fields.
6. Issue the PUTUNIT macro to make any JES3 devices obtained using the GETUNIT macro available for allocation to other functions.
7. Issue the LOGOUT macro to end operator communication. If a DSP has started processing, always issue the LOGOUT macro to write the JDAB, end operator communication, or both.
8. Issue the AWRITE macro for the JDAB (if this was not done using LOGOUT). If the DSP did not update the JDAB, you can issue an ARELEASE macro instead.
9. Free any working storage obtained by your DSP, and clean up other resources used. Your DSP must release any resources it has obtained through MVS or JES3 services (such as ALOAD, AGETBUF, GETMAIN, JDSGET, and so on.).
Certain JES3 resources like the FCT, RQ, data CSECT, and the first DSP module that were provided on entry to the DSP are freed by JES3.

10. Delete the JESTAE environment.

11. Restore the contents of registers 11, 12, and 14 to what they were on entry to the DSP because JSS (job segment scheduler) depends on them. The JSS driver module, IATGRJR, then performs end-of-function processing.

12. Return control to JSS, specifying the appropriate return code in register 15. (See Table 3 for return codes.) (This means that you should put the return code in register 15 instead of using the RC= parameter on the ARETURN macro.)

Table 3. Return Codes Passed to JSS

<table>
<thead>
<tr>
<th>Code</th>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>FCTJSNRN</td>
<td>The function is complete. The scheduler element (SE) in the JCT for this job and function is set “complete,” and this job is returned to the queue for scheduling of further functions if required.</td>
</tr>
<tr>
<td>4</td>
<td>FCTJSRSK</td>
<td>The function is returned for later scheduling. The SE in the JCT for this job and function is set “not active” and “not complete,” and this job is returned to the queue in operator hold status. The operator must release the job for further processing to occur.</td>
</tr>
<tr>
<td>8</td>
<td>FCTJSSPR</td>
<td>The function is returned for specialized rescheduling. For all scheduler elements except C/I, the FCT GETUNIT list for the DSP must indicate the rescheduling requirements. JSS will reschedule the DSP when the device requirements in the list can be satisfied. The C/I scheduler element uses this return code when a dependent job control job requires the completion of another job before it can be scheduled, or when a processor is not available for a C/I catalog search (LOCATE processing). JSS reschedules the job when predecessor jobs complete, or when an eligible processor becomes available.</td>
</tr>
<tr>
<td>12</td>
<td>FCTJPURG</td>
<td>The function is returned for purging. All scheduler elements in the JCT for this job for all functions except PURGE and ENABLE are set “complete,” and the job is purged from the system without any further activity.</td>
</tr>
<tr>
<td>16</td>
<td>FCTJSCAN</td>
<td>The function is returned for canceling with print. The scheduler elements in the JCT for this job for all functions except ENABLE, OUTSERV, PURGE, and CBPRNT, if present, are set “complete.” This job is then returned to the queue.</td>
</tr>
<tr>
<td>20</td>
<td>FCTABDSP</td>
<td>The function is returned from FAILSOFT. This occurs when FAILSOFT ends a DSP. The FCT entry for this job is removed from the FCT chain and the function is not rescheduled.</td>
</tr>
<tr>
<td>24</td>
<td>FCTIieres</td>
<td>The function is returned from the converter/interpreter (C/I) for later scheduling. This may occur when an operator decreases the C/I maximum DSP use count. The scheduler element in the JCT for this job is set “not active, not complete,” and the job is returned to the queue. JSS reschedules the DSP when the C/I DSP becomes available. Only the converter/interpreter should use this return code.</td>
</tr>
</tbody>
</table>

1 Defined for byte FCTFLAG2 in the IATYFCT macro and for byte RQDSPRC in the IATYRSQ macro.

Module Loading

If your program consists of more than one module, you should organize it so that the first module is reenterable and all other parts are at least serially reusable. If you use multiple modules in your program, use the ALOAD and ADELETE macros for dynamic loading and unloading of the subprograms. In case of a failure in your DSP, be sure to set up a JESTAE environment that includes the ADELETE macro.
If the DSP dictionary includes the name of a data CSECT, the job segment scheduler driver (module IATGRJR) loads the appropriate CSECT and puts its address in register 13 before execution of the reenterable DSP. The only required statement to address this CSECT is:

```
USING csect-name,R13
```

Obtain additional storage, if required, by using the AGETMAIN macro.

IBM supplies a module for each JES3 installation exit. (Some installation exit modules are dummy modules). You can replace the IBM module with your own module in the JES3 library. If JES3 tries to load an installation exit and cannot find it, JES3 issues warning messages IAT3020 and IAT3102.

**ASAVE Linkage and Save Area Use**

JES3 provides a standardized linkage routine that has two purposes:

- It saves and restores registers over subroutine calls.
- It provides a trace of the call and the return that you may find helpful when diagnosing a problem.

The linkage routine, called ASAVE (in module IATGRSV), obtains a register save area and stores registers 2 through 13 in the area. When a return is made using the linkage routine, the registers are restored from the save area and the save area is released.

Most JES3 macros use ASAVE linkage when calling JES3 service routines. To use this capability in your own modules, you can use the following two macros to invoke the linkage routine:

- `ACALL` lets the caller enter a subroutine through the ASAVE linkage routine.
- `ARETURN` lets the called routine return using the linkage routine to the calling routine. Return is typically to the instruction following the ACALL but the RC= parameter can be used to return to a specified entry in a branch table following the ACALL.

The example shown in Figure 5 on page 27 illustrates the use of the ACALL and ARETURN macros. See Introduction for more information on these macros.

For tracing registers, the standard linkage routine does the following. On the ACALL, all registers received by the called routine are traced after the BALR is taken. For ARETURN, the following registers are traced:

- Register 10 – Traced as the user left it, but JES3 assumes it to be the base address of the module (or subroutine)
- Register 13 – Points to the location where the registers are saved
- Register 14 – Points to the actual address the subroutine will go back to
- Registers 15, 0, 1 – Traced as the subroutine left them when it issues the ARETURN macro
Table-Referencing Techniques

All JES3 tables should be referred to by using DSECTs generated by mapping macros. For ease of reference, place these macros at the beginning of a program after the prolog but before any executable instructions. Use of the mapping macros insulates the user-written DSPs from changes to system control blocks.

Do not make references to fields within tables where a specific order of the field data is assumed, as the order of the fields is subject to change.

Restrictions on MVS Services Available to DSPs and Installation Exit Routines

All MVS control program services, with the following exceptions, are available to user-written DSPs. User exits run under a DSP.

Do not use MVS macros that either implicitly or explicitly use the MVS WAIT function, except under the control of a JES3 subtask (see the IATXCSF macro). These macros include:

- ATTACH
- FIND
- PURGE
- BDL
- LINK
- WAIT
- CLOSE
- LOAD
- WTO
- DOM
- LOCATE
- WTOR
- DYNALLOC
- OPEN
- XCTL

DSPs should not use these macros. Use of these macros disrupts the flow of processing on the global main and can cause a degradation in system performance by possibly causing JES3 itself to wait. The queued access methods, QSAM and QISAM, use these macros and, therefore, should not be invoked.

If your program needs to use one of the above macros, you must first establish a JES3 subtask environment in which to use the macro. To do so, include the IATXCSF macro in your program. IATXCSF passes control to a JES3 subtask.
Do not use the MVS macros LOAD, DELETE, STIMER, and PCLINK because they interfere with JES3’s processing procedures. Use the ALOAD and ADELETE macros instead of LOAD and DELETE, and use the ATIME macro instead of STIMER. Use the MVS ESTAE and SETFRR macros only where absolutely necessary. Instead, use the JESTAE macro.

Under certain restrictions, your module may issue a DYNALLOC macro. First, be sure you issue the IATXCSF macro before the DYNALLOC macro to set up the JES3 subtask. Secondly, use the DYNALLOC macro only for jointly-managed or MVS-managed devices, not JES3-managed devices. Modules that run in a C/I FSS address space follow the same rules for using DYNALLOC as an authorized problem program.

If your DSP uses an MVS function that could cause an abnormal termination, be sure that you set up a JESTAE recovery environment. The purpose of the recovery environment is to insure that a failing DSP does not also bring down the JES3 primary task (IATNUC).

Your recovery routine should intercept the ABEND and pass control to a retry routine to retry the DSP. The retry routine should then decide what should happen to the DSP. Generally, there will be three choices of action:
- Let the DSP continue processing
- Reinitialize the DSP to handle the next request
- End the DSP

The following example illustrates the use of the IATXCSF macro, which is described in “Introduction” on page 169. The code to be executed under control of the subtask begins at label DRCLS BTK.

### IATXCSF Macro Example

<table>
<thead>
<tr>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TM</td>
<td>DCBOFLGS,DCBOFOPN Is DCB open?</td>
</tr>
<tr>
<td>BC</td>
<td>ALLOFF,DRTTB347 No, avoid closing it</td>
</tr>
<tr>
<td>IATXCSF</td>
<td>ENTER=DRCLS BTK Call subtask to close it</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* THE FOLLOWING CODE EXECUTES AS A SUBTASK OF JES3

<table>
<thead>
<tr>
<th>Label</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DRCLS BTK</td>
<td>DS OH Subtask to close DCB</td>
</tr>
<tr>
<td>LR</td>
<td>R4,R14 Save return address</td>
</tr>
<tr>
<td>CLOSE</td>
<td>(ICTPRDCB),MF=(E,ICTMFLOP) Issue MVS CLOSE macro</td>
</tr>
<tr>
<td>LR</td>
<td>R14,R4 Restore return address</td>
</tr>
<tr>
<td>SR</td>
<td>R15,R15 Clear return code</td>
</tr>
<tr>
<td>BR</td>
<td>R14 Return to caller</td>
</tr>
</tbody>
</table>

### JES3 Function Synchronization

The synchronization of parts of a function within JES3 is accomplished by issuing the AWAIT macro with either the TYPE=ON or TYPE=OFF parameter. This macro signifies that the DSP is waiting for an event to complete before proceeding.

The AWAIT macro refers to an event completion flag (ECF) to synchronize processing. The macro specifies the address of the ECF. The macro also supplies an ECF mask, which is compared to the ECF to determine when one or more of the events have occurred.
For the AWAIT TYPE=ON macro, completion is indicated when any corresponding ECF mask bit is on in the ECF; hence, if an ECF mask of X'51' is specified, completion occurs when the ECF is set to X'40', X'10', X'01', or any combination of these values.

For the AWAIT TYPE=OFF macro, completion is indicated when all bits specified in the ECF mask are off in the ECF; hence, if the ECF mask specified X'60', completion occurs when bits one and two of the ECF are off.

It is important that you understand the environment of JES3 at the time an AWAIT is issued. If non-serialized resources are being used when an AWAIT is processed, these resources may be changed or deleted before the completion of the AWAIT. For example, you need information from the job data accounting table (JDAB), and you use the JESREAD macro specifying the JDAB file description block (FDB) in a job's RESQUEUE. The JESREAD issues an AWAIT while waiting for the I/O to complete. If the RESQUEUE can be used concurrently by other DSPs, the RESQUEUE may be changed or deleted before the JDAB is completely read.

**Note:** The AWAIT macro (or functions using implied AWAITs, such as JESREAD) should not be issued in a DSP's JESTAE exit. The DSP should request retry and issue those macros in the retry routine.

Normal JES3 DSP processing consists of short execution intervals between waits. For most DSPs, these breaks occur naturally. However, if a DSP runs a long time without giving up control, it should issue the AWAIT TYPE=ON macro specifying a posted ECF at regular intervals.

The AWAIT TYPE=ON macro specifies that control is to be returned to the function issuing the macro only when at least one of the specified events have occurred. By providing dummy AWAIT macros in which the events being waited for are already posted as complete, the DSP gives the multifunction monitor (MFM) control to let higher priority DSPs execute.

**I/O for Tape, Unit Record, and BSC RJP Devices**

Initiation of input/output (I/O) operations is the same for both remote and local devices. That is, for both remote and local devices, your DSP must issue the following sequence of macros:

1. **JESOPEN**: to open the device
2. **JESEXCP**: to request I/O for the device
3. **JESCLOSE**: to close the device

Data chaining is only supported for print devices, and command chaining is supported for all output devices.

I/O for tape, unit record, and BSC RJP devices cannot be performed by DSPs and installation exit routines operating in a C/I FSS address space.

The following example illustrates the use of the JESOPEN, JESEXCP, and JESCLOSE macros. See **Introduction** for more information on these macros.
Storage Management

Four macros (AGETMAIN, APUTMAIN, IATXBGM, and IATXBFM) manage general JES3 storage. For other macros involved with storage management, see Table 16 on page 174. The AGETMAIN and APUTMAIN macros manage storage in the JES3 address space and in C/I FSS address spaces. The AGETMAIN macro gets an area of storage. If the amount of storage requested is not available, the calling routine can wait (using the BUSY= parameter) until the required amount becomes available. The APUTMAIN macro releases the storage when it is no longer needed.

The subsystem interface and data management use the IATXBGM and IATXBFM macros to manage storage in subpools 231, 241, and 245. The IATXBGM macro gets an area of storage from one of the subpools. The request can be either unconditional or conditional. If the request is unconditional and the amount of storage required is not available, JES3 abnormally ends the caller. If the request is conditional and the amount of storage required is not available, the macro returns a nonzero return code to the requesting routine. The IATXBFM macro releases storage in the subpool when it is no longer needed.

When obtaining storage, you can specify whether you want storage above or below 16 megabytes. The macros for obtaining storage contain a LOC= parameter that allows you to specify where the storage should come from.

See Introduction for more information on these macro instructions.

Making Entries in the JES3 Trace Table

Certain JES3 events are recorded in the JES3 event recording trace table by using the IATXTRC macro. The ID parameter uniquely identifies the event. IDs 0001 through 2999 are reserved for IBM and IDs 3000 through 4095 for the installation’s system programmer. All entries will contain a standard header including a time-of-day value. The system programmer may optionally supply data or registers to be traced. (See z/OS JES3 Diagnosis for information on which JES3 events are recorded.)

The following example illustrates the use of the IATXTRC macro, which is described in Introduction.
Enqueuing and Locks

Some JES3 resources (such as the FCT chain) must be accessed by only one DSP at a time. To protect against concurrent use of such resources, JES3 provides an enqueuing method that uses the ATEST, AENQ, and ADEQ macros. The IATYRSC macro defines the resources manipulated by these instructions.

These macro instructions are used to access JES3 resources as follows:
- AENQ obtains use of the named resource.
- ADEQ releases control of the named resources.
- ATEST tests the availability of the resource.

For an explanation of the use of locks in a z/OS system, see z/OS MVS Programming: Authorized Assembler Services Guide. For a description of the locks used by JES3, see z/OS JES3 Diagnosis.

The following example illustrates the use of the ATEST, AENQ, and ADEQ macros. See Introduction for more information on these macros.

```
ATEST NAME=JNCBCTL,TYPE=FCT,ERROR=RNTERROR
LTR R1,R1
BC NZERO,INUSE
AENQ NAME=JNCBCTL,BUSY=WAIT
.
.
ADEQ NAME=JNCBCTL
```

Accessing Spool-Resident Control Blocks

DSPs and installation exit routines can access spool-resident control blocks (such as the job description and accounting block (JDAB)) by using the JESREAD, AWRITE, WRTCHAIN, and ARELEASE macros. To create a spool-resident control block, JES3 issues the AGETBUF macro, which obtains a JSAM buffer. Then the calling routine builds the control block in the buffer.

User-written DSPs can create their own spool-resident control blocks in the same manner. Your program can use the IATYSRF macro to generate single record file headers for any spool control blocks that your program creates.

If your DSP runs under a subtask and the DSP needs to access the spool data set, you can use the IATXSDM macro in your subtask appendage to initialize access to the spool.

Writing a Spool-Resident Control Block

After the control block has been created, your program should issue the AWRITE macro to write the contents of the JSAM buffer to the spool. The AWRITE macro must specify the address of a file description block (FDB) defined by the calling

Figure 6. Use of the IATXTRC Macro
routine. The first four bytes of the specified FDB must contain the address of the
JSAM buffer containing the control block. The rest of the FDB should be initialized
to zeros.

When a spooled set of control blocks exceeds the size of a JSAM buffer, your
program is responsible for splitting the control blocks across buffers and chaining
them together. This results in the formation of a chained single record file (SRF).
The WRTCHAIN macro can be used to write a chained single record file [back] to
spool. You have the option of writing only modified records instead of all records in
the chain by specifying the MODONLY option on the WRTCHAIN macro.

Reading a Spool-Resident Control Block
A routine can bring a spool-resident control block back into a JSAM buffer using the
JESREAD macro. The JESREAD macro must specify the address of the same FDB
used by the AWRITE macro that initially wrote the control block to the spool. Then
the routine can use the control block in the buffer as needed.

The JESREAD macro must also specify the ID= parameter indicating the record to
be read. The ID= parameter is required on the macro. When your routine writes a
record, the routine must put a “record type identifier” in the ID field in the buffer.

When your routine reads a record, it must specify the ID= parameter on either the
JESREAD macro or the PURCHAIN macro. The ID value is compared to the ID in
the record being read, and if they do not match, the read request fails.

If the routine does not change the control block (or does not need to save the
changes), the routine frees the JSAM buffer occupied by the control block by
issuing the ARELEASE macro. The ARELEASE macro also changes the address of
the control block in the FDB from a buffer address to a spool address, so that later
JESREAD macro calls can read in the control block. If the routine changes the
control block and needs the changes saved, the routine issues the AWRITE macro
to write the contents of the buffer to the spool. Unless otherwise specified, the
JSAM buffer is freed after being written to the spool.

Serializing Access to FDBs (JESREAD Macro)
When a routine issues a JESREAD macro, the JESREAD macro service routine
serializes access to the FDB. Until the calling routine issues an AWRITE or
ARELEASE macro for that FDB, any later JESREAD macro requests for the same
FDB must wait. The JESREAD serialization prevents different DSPs from making
conflicting changes to a control block they are using concurrently.

If you code (or let default) AWAIT=YES on the JESREAD macro, control is not
returned to your routine until I/O is complete. Your routine then has exclusive
control of the FDB, and you do not have to test for I/O completion in the FDB. If
you code AWAIT=NO on the JESREAD macro, control is returned to your routine
immediately. You must then test for I/O completion before trying to use the FDB.
For more information, see the description of the JESREAD macro.

Shared Access to a Spool-Resident Control Block
If several DSPs need access to a spool-resident control block and can access it
without interfering with one another (for instance, if none of them update the control
block), the DSPs can share access to the control block. To avoid the JESREAD
serialization, the DSPs must serialize access to the control block themselves, as
needed. After the serialization logic (if any) but before accessing the control block,
The DSP must carry out several tests.
1. The DSP must test the control block’s FDB to determine if another DSP is reading the FDB into storage or writing the FDB to the spool data sets. If bit FDBCLOSE is off, I/O is in progress. The DSP must issue an AWAIT macro to wait until bit FDBCLOSE turns on. If bit FDBCLOSE is on, the FDB is accessible.

2. Next, the DSP must test the FDB to determine if the control block is already in a JSAM buffer. If bit FDBONSP is off, the first four bytes of the FDB contain the address of the JSAM buffer containing the control block. If bit FDBONSP is on, the DSP must issue the JESREAD macro to bring the control block from the spool into a JSAM buffer.

You can avoid doing these tests if your DSP and any other DSP that accesses the control block copies the FDB into a work area before reading it. If there is at least one DSP that reads the control blocks using the FDB, then this technique cannot be used.

The DSP and other DSPs can now access the control block directly in the buffer.

**Serializing Access to the JDS**

The JDS control block is provided with its own serialization techniques. When accessing individual JDS buffers, you should use the JDSHOLD macro to serialize the entire JDS data set for the job. When the JDS is held, IBM suggests using the IATXJDS macro rather than JESREAD, AWRITE, or ARELEASE to access individual JDS buffers.

Once serialized, use the JDSREL macro to release the JDS.

**Managing SWA (Scheduler Work Area) Control Blocks**

When JES3 reads the JCL for jobs to be executed, it builds control blocks containing data about the job. This data is used by the operating system when it executes the job. The information in these control blocks comes from the job’s JCL statements.

These control blocks are used by the scheduler function and they are called SWA control blocks (for the scheduler work area). A SWA control block is an internal representation of a job’s JCL. If a job has many JCL statements or uses many data sets and devices, there will be numerous SWA control blocks built for the job. These SWA control blocks could occupy a significant quantity of storage.

SWA blocks are actually placed into storage twice in a JES3 environment.

1. First, during JCL interpretation when the SWA blocks are created in the JES3 global or CI FSS address space, they are automatically moved into virtual storage above 16 megabytes by JES3. (All of the SWA blocks created by the converter/interpreter are placed above 16 megabytes.) This leaves more room in the JES3 address space below 16 megabytes for other JES3 control blocks. After JCL interpretation, the blocks are written to the spool data set (or spool for short).

2. When the job is selected later for execution, the SWA blocks are read from spool and relocated in storage in the job’s address space. In the job’s address space, the SWA blocks will not necessarily be moved above 16 megabytes. However, you can customize your JES3 system so that in certain situations a subset of SWA blocks will be moved above 16 megabytes in the job’s address space. For example, if your installation has some large jobs that will create many SWA control blocks, you can use certain JES3 installation exits to indicate that the blocks should be moved by keying off the job name.
Some of the steps you can take to ensure that SWA control blocks are moved above 16 megabytes are listed here:

- **Converter/Interpreter (CI) parameter list definition:**
  The parameters passed to the MVS Converter/Interpreter contain a flag indicating the job can have its SWA placed above 16 megabytes. You can set this flag to indicate “SWA above” by coding either a 4, 5, 6, or 7 as the first character in the IPARM initialization statement. Either of these values (in combination with other settings) will insure that the flag is turned on. (The flag is the third bit from the right (B'00000100') in the first byte of the CIPARM statement.)

- **During Input Services:**
  Installation exit 29 can be used to alter the CIPARM list for a job by changing the JDABPMID field in the JDAB.

- **During MVS CI Processing:**
  Installation exits IATUX03 and MVS exit IEFUJV can be used to change the setting of NELXA2 within NELPARM0. If NELXA2 is set on, SWA is allowed to be placed in virtual storage above 16 megabytes in the job’s address space.
  Installation exit 41 can be used to set IDDSWA in the IATYIDD which will allow the job’s SWA to be placed above 16 megabytes. Note that this exit is only invoked when the MAXJOBST value is exceeded for a job.

- **During Prescan Processing:**
  Installation exits 4, 5, and 6 can be used to change the setting of IDDSWA in the IATYIDD. If IDDSWA is set, SWA is allowed to be placed above 16 megabytes in the job’s address space.

- **During Job Initiation:**
  Installation exit 26 can be used to change the setting of IIWSWABV, in macro IATYIIW, but only on the first call for a job. If IIWSWABV is set, SWA is allowed to be placed above 16 megabytes in the job’s address space.

Note that you can only control whether some SWA is placed above 16 megabytes. You cannot control which SWA blocks are actually placed there.

**Accessing SWA Control Blocks**
All code that you write in DSPs or exit routines that acquires the address of a SWA block must use the SWA manager interface, the SWAREQ macro. By using SWAREQ, you will be able to access any SWA control block, regardless of its location.

Because all SWA blocks created in the JES3 address spaces are placed above 16 megabytes, all user code referencing SWA in JES3 address spaces must use 31-bit addressing mode. Be aware of this if you decide to use exits 3, 4, 5, or 6.

**Conventions for Communicating with the System Operators**
This section describes two JES3 facilities that allow you to write code that can communicate with JES3 operators. The first facility is a console appendage that accepts entries from the JES3 console services function. The second facility is the MESSAGE macro that can send a message to an operator’s console.

**Console Appendages**
All DSP-to-operator communication is accomplished by the console service routines. Each DSP that requires two-way communication with the operator must define a **console appendage routine** for accepting asynchronous entries from
console service. The DSP must use the LOGIN macro to define the console appendage routine’s entry point. For reentrant DSPs, therefore, appendages are typically located in the DSP data CSECT.

The console appendage is entered when a *START, *RESTART, or *CANCEL command is received for the DSP, and may be entered when a *FAIL command is received.

Because DSP console appendages run under the CONCMD FCT, which is the highest priority FCT, their processing time should be as short as possible. You should limit processing to deciding whether to accept the command, and, if necessary, saving the command and posting the command processing routine of the DSP.

On entry to the console appendage, register 1 contains the address of the console message buffer (which is mapped by the TYPE=INPUT expansion of the IATYCNS macro). The buffer includes the console destination block (CNDB), verb code, message length, and flags used by the JES3 command processing routines (IATXSCN1 and IATXSCN2). Register 0 contains the FCT address of the function that owns the console appendage.

The appendage should copy the CNDB contained within the IATYCNS parameter area using the IATXCNDB OPERATION=TRANSFER service. The DSP will subsequently use the MESSAGE service specifying this CNDB to communicate with the console issuing the command. The appendage should determine the state of the DSP and decide, based on the type of command, whether the DSP can process the command at that time. The results of that decision must be posted to the DSP driver. The appendage must return to console service using the ARETURN macro with one of the following return codes (shown in decimal form):

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>The DSP accepts the command for processing. In this case, the appendage should save a copy of the buffer, because the CONCMD DSP will release the buffer on return from the appendage.</td>
</tr>
<tr>
<td>04</td>
<td>The DSP requests that the command be queued for later processing. In this case, the buffer is queued from the DSP’s FCT. The DSP must dequeue the buffer (using the DEQMSG macro) when the buffer completes processing.</td>
</tr>
<tr>
<td>08</td>
<td>The DSP cannot process the command at this time, or the command is incorrect. In this case, the CONCMD DSP sends a rejection message (IAT7130) to the calling console.</td>
</tr>
<tr>
<td>12</td>
<td>The DSP cannot process the command at this time, or the command is incorrect. In this case, the CONCMD DSP sends a rejection message (IAT7130) to the calling console, and register 1 contains the address of a message length and text to be displayed in the IAT7130 message. The message text cannot exceed 21 characters.</td>
</tr>
</tbody>
</table>

The LOGOUT macro is used to end the console communication linkage established by the LOGIN macro.

**Working with Console Destination Blocks**

JES3 uses Console Destination Blocks (CNDBs) to save console identification and message routing information. JES3 uses the IATXCNDB macro service to work with
CNDBs. IATXCNDB includes services for creation, update, copy, and extraction. You should use the IATXCNDB service whenever you work with CNDBs.

When an operator issues a command, JES3 creates a CNDB containing information about the source of the command, such as the console identifier and console type. This CNDB is passed to the DSP’s console appendage along with the command text. When your DSP needs to communicate with the operator issuing the command, you may use the MESSAGE macro service, specifying the CNDB that was saved by your appendage. In addition, any commands that your DSP may internally issue on behalf of an operator may be issued using the INTERCOM service specifying a CNDB.

CNDBs may also be used for routing of unsolicited messages. For example, JES3 creates CNDBs for much of the message destination information specified in your initialization stream. When you need to issue a message to this destination, such as a message about a particular main processor, you use the MESSAGE service specifying the appropriate CNDB.

Sending Messages to the Operator
When a JES3 DSP needs to communicate with a system operator, it uses a MESSAGE macro to send the message to the operator’s console. (Based on certain conditions, JES3 uses other macros as well to issue messages. These conditions are explained in "Choosing the Appropriate Macro to Issue a Message" on page 42.) JES3 generally routes the message using the information contained within a console destination block.

Commands sent by an operator from a console are answered by JES3 using the MESSAGE macro, which permits a reply to be routed back to the operator console that issued the command.

When you write a DSP or installation exit that needs to send messages to the operator’s console, follow the guidelines used by JES3 that are explained in Choosing the Appropriate Macro to Issue a Message. In general, you will use the MESSAGE macro most of the time when issuing a message.

Use the ACTION and DESC parameters of the MESSAGE macro to define messages of differing types and importance. The ACTION parameter identifies the message as one of the following types:

- System or subsystem immediate action message
- System or subsystem informational message
- System or subsystem failure message
- Critical eventual action message

(A critical eventual message indicates that the system has detected a problem and action should be taken. However, the system can continue and will not wait for a response.)

The DESC parameter lets you specify any multiple console support (MCS) descriptor code(s). Based on the capabilities of the consoles, the ACTION and DESC parameter values (together) cause the system to display messages in different colors or intensities. If you want, you can use the MVS Message Processing Facility (MPF) to customize the display attributes of messages appearing on these consoles.

Table 4 on page 37 shows the default colors or intensity that each type of message produces. MPF exits changes them on a message basis. It also shows the macro
specifications that you may use to define a message as a particular type. For the MESSAGE macro description, see [Introduction].

Table 4. Color or Intensity of Messages by Message Type

<table>
<thead>
<tr>
<th>Type of Message</th>
<th>Corresponding MCS Descriptor Code</th>
<th>Color/Intensity</th>
<th>Macro Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>System or subsystem immediate action</td>
<td>(2,7)</td>
<td>White/intensified</td>
<td>MESSAGE ...,ACTION=YES WTO ...,DESC=(2,7)</td>
</tr>
<tr>
<td>Write to operator with reply</td>
<td>None</td>
<td>White/intensified</td>
<td>WTOR</td>
</tr>
<tr>
<td>System or subsystem failure</td>
<td>1</td>
<td>Red/intensified</td>
<td>MESSAGE ...,ACTION=FAIL WTO ...,DESC=1</td>
</tr>
<tr>
<td>Critical eventual action</td>
<td>(7,11)</td>
<td>Red/intensified</td>
<td>MESSAGE ...,ACTION=CRITICAL WTO ...,DESC=(7,11)</td>
</tr>
<tr>
<td>Informational</td>
<td>7</td>
<td>Green/normal intensity</td>
<td>MESSAGE ...,[ACTION=NO] WTO ...,DESC=7</td>
</tr>
</tbody>
</table>

**Message Processing**

Most messages your installation issues are either issued directly as WTOs or the macro used to issue the message expands to a WTO. The parameters you specify on the macros that issue messages dictate message processing and message display. Message processing refers to the:

- Suppression of a message
- Retention of a message
- Routing of a message
- Automation of a message

Message display refers to color, highlighting, and intensity attributes that the system uses when displaying messages at certain consoles.

You can use WTO/WTOR and JES3 installation exits to change the message processing and display specifications made when the message was issued. Understanding WTO/WTOR processing and how your installation exits affect the routing and display of a message will help you anticipate where the system will display the message. The system makes all decisions for the routing and display of the message on the main where it was issued. [Figure 7 on page 38](#) illustrates the processing performed for each message that is issued in your installation.
There are some special routing considerations for messages issued from functional subsystems (FSSs). See [z/OS JES3 Initialization and Tuning Guide](#) for more information about message routing exceptions.

1. Most messages issued in your installation are either issued directly as WTO/WTORs or are issued by another macro service that expands into a WTO.
2. You can use MPF to change the presentation and display of a message by implementing:
- A parmlib member of MPFLSTxx where you can:
  - suppress the message
  - retain the message
  - have an automation package process the message
  - specify an installation-supplied installation exit.
- An installation-supplied installation exit, such as IEAVMXIT.

You can use MPF installation exits to:
- Change or add to the routing and descriptor codes that were specified on the WTO/WTOR macro that issued the message.
- Alter the message text.
- Change whether the system can modify the routing for the message.

To process a message, MPF determines if you specified an installation-supplied installation exit in the MPFLSTxx member for the message. If an installation-supplied installation exit exists, MPF calls it to process the message. Otherwise, MPF uses the WTO/WTOR installation exit, IEAVMXIT, if it is active.

After the message has been processed by MPF exits, you may have added, deleted, or completely changed the message's original processing and display specifications. MCS sends the message to the subsystem interface (SSI) for processing where it can be processed by any subsystem that is waiting for WTO/WTOR message traffic.

3. Because JES3 is the primary subsystem, it is the first subsystem to receive notification of the WTO/WTOR. JES3 uses the routing decisions to perform functional message routing. Functional message routing may override the routing information specified by MPF:
- Combining the results of functional message routing with the routing codes MPF specified. These routing codes are used to display the message on MCS-managed consoles.
- Determining if the message should only be written to the log.
- Determining if the message should be displayed on a specific console.

Table 5 identifies how the SSI uses functional message routing to select the JES3 routing information. The numbers in the category column indicate the sequence of processing. If the message fits into more than one category, the last category overrides the routing decisions set by the previous category.

<table>
<thead>
<tr>
<th>Category</th>
<th>Message will be Routed to</th>
<th>Routing Code/Destination Class</th>
<th>The Next Step in Processing</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Deleted Message</td>
<td>JESMSG data set</td>
<td>JESMSG data set</td>
<td>No additional checking is required</td>
</tr>
<tr>
<td>2. Hardcopy only</td>
<td>Hardcopy log</td>
<td>Hardcopy log</td>
<td>Checks for a specified console, see step 10</td>
</tr>
<tr>
<td>3. Broadcast</td>
<td>Consoles receiving broadcast messages</td>
<td>ALL</td>
<td>Checks for a specified console, see step 10</td>
</tr>
<tr>
<td>4. Specified routing code</td>
<td>Dependent on the definition of the routing code on the MSGROUTE initialization statement</td>
<td>Selection is based on SUBSMOD= parameter. For additional information see MESSAGE (Standard Form: Issue a Message from a JES3 DSP) on page 548</td>
<td>Determines if the message was issued from an FSS, see step 8</td>
</tr>
</tbody>
</table>
### Table 5. Selection of Final Routing Code for WTO/WTOR Messages (continued)

<table>
<thead>
<tr>
<th>Category</th>
<th>Message will be Routed to</th>
<th>Routing Code/Destination Class</th>
<th>The Next Step in Processing</th>
</tr>
</thead>
<tbody>
<tr>
<td>5. Job status message</td>
<td>Consoles receiving the destination specified on the MAINPROC statement</td>
<td>Destination specified on MAINPROC statement</td>
<td>Determines if the message was issued from an FSS, see step 8</td>
</tr>
<tr>
<td>6. MSGTYPE parameter was the only parameter specified</td>
<td>Consoles receiving the destination specified on the MAINPROC statement</td>
<td>Destination specified on MAINPROC statement</td>
<td>Determines if the message was issued from an FSS, see step 8</td>
</tr>
<tr>
<td>7. No routing codes specified or added by MPF</td>
<td>Consoles receiving the destination specified on the MAINPROC statement</td>
<td>Destination specified on MAINPROC statement</td>
<td>Determines if the message was issued from an FSS, see step 8</td>
</tr>
<tr>
<td>8. Message from an FSS</td>
<td>The console where the *CALL WTR command was issued</td>
<td>Destination specified on the JUNIT parameter of the DEVICE statement for the FSS</td>
<td>Go to step 9</td>
</tr>
<tr>
<td>9. Special messages</td>
<td></td>
<td></td>
<td>Go to step 10</td>
</tr>
<tr>
<td>SETPRT-related messages (for local writers only)</td>
<td>No console</td>
<td>Hardcopy log</td>
<td></td>
</tr>
<tr>
<td>Device-related messages</td>
<td>Unchanged</td>
<td>JUNIT parameter of the DEVICE statement</td>
<td></td>
</tr>
<tr>
<td>RJP job-related messages</td>
<td>RJP workstation</td>
<td>Unchanged</td>
<td></td>
</tr>
<tr>
<td>Job failure-related messages</td>
<td>No console</td>
<td>Hardcopy log</td>
<td></td>
</tr>
<tr>
<td>System-related messages</td>
<td>Unchanged</td>
<td>Destination specified on the MAINPROC statement</td>
<td></td>
</tr>
<tr>
<td>10. Specific console was specified</td>
<td>Specified console</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

After JES3 has determined the message’s routing codes, the decisions made by functional message routing are:
- Used to issue the message on the appropriate MCS consoles in the sysplex.
- Sent in a staging area to the JES3 on the global, if required.
- Sent to other subsystems listening for WTO/WTOR message traffic. The other subsystems can change or add to the message routing JES3 used.

4. JES3 on the global receives the staging area and passes the message to IATUX70, if requested by IATUX69.

5. Before displaying the message on an MCS-managed console, if the JES3 GLOBMPF option is in effect, the message processing facility (MPF) processes the message. On the global, only the installation-supplied installation exits and IEAVMXIT (if it is installed) process the message. See [z/OS MVS Installation Exits](https://www.ibm.com/support/knowledgecenter/SSL面目d_10?topic=zosInstallationExits.htm) for information on implementing the MPF exits.

6. The SSI on the global:
- Allows any automation products that your installation has installed on the global to process the message.
- Uses the routing codes that were passed by the SSI on the local to display the message.

(Always use a LOGOUT macro if a LOGIN macro was used previously.)
Installation Exits Involved with Message Processing

MVS and JES3 both follow certain criteria to determine where to send a message. You define the criteria when you initialize the MVS system, and later when you initialize the JES3 system. This discussion does not explain everything you need to know about setting up message routing when you initialize MVS and JES3. For complete information about JES3 message processing, see z/OS JES3 Initialization and Tuning Reference.

This section describes how JES3 handles messages so that you can see how certain installation exits (specifically, exits 57, 69, and 70) can be used to further customize message handling in your system, and so that you can choose the correct macro to issue a message in a given situation.

MVS Facilities and Message Processing

In addition to JES3 exits 57, 69, and 70, the message processing facility (MPF) can help you tailor message processing on your system. IBM strongly recommends that you tailor your message processing using the MVS facilities instead of the JES3 facilities. Use the MVS facilities on all processors (global and locals) where messages can be issued because these facilities will receive control for every message.

For information on the MVS message facilities, see z/OS JES3 Initialization and Tuning Reference.

Selecting a Routing Code for JES3 MSGROUTE Processing (Installation Exit 57)

If a message is issued with only one routing code, JES3 uses that routing code to perform MSGROUTE processing. If a message is issued with more than one routing code, JES3 has to select a single routing code to perform MSGROUTE processing.

JES3 converts the single routing code that it chooses to an individual JES3 destination class and sends it to the appropriate consoles and logs based on that class. To determine which single routing code to use, JES3 uses the default algorithm shown in Table 6. If the JES3 algorithm is not suitable for your installation, use installation exit 57 to implement your own algorithm. Your algorithm then determines which single routing code to use during JES3 message route processing, instead of letting JES3 decide.

Table 6. Default Algorithm for Selecting a Single Routing Code to Present to JES3

<table>
<thead>
<tr>
<th>If your installation does not use exit 57 to select a single routing code when more than one routing code is present, JES3 selects one for you using the following algorithm:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Discard routing code 11.</td>
</tr>
<tr>
<td>2. Out of those that are 16 or less, choose the highest code and use it; and ignore all codes greater than 16.</td>
</tr>
<tr>
<td>3. If all of the routing codes are 17 or greater, choose the lowest code and use it.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
</tr>
<tr>
<td>• JES3 receives a WTO specifying routing codes 2, 11, 14, and 42.</td>
</tr>
<tr>
<td>• JES3 immediately discards code 11, and ignores code 42.</td>
</tr>
<tr>
<td>• JES3 then selects code 14 instead of code 2 because 14 is higher than 2. 14 is the winning routing code.</td>
</tr>
</tbody>
</table>

If a WTO macro specifies SUBSMOD=NO, JES3 does not perform MSGROUTE processing and does not call IATUX57.
Routing Messages to a JES3 Global Address Space Exit (Installation Exits 69 and 70)

You may infrequently need to perform some processing related to a message when running in the JES3 global processor’s address space rather than an MPF exit on the issuing processor. Installation exits 69 and 70 provide this function. Exit 69 can be used to examine every message on the issuing processor and to instruct JES3 to deliver selected messages to exit 70. JES3 will then transport the selected messages to the global processor and present them to exit 70, which runs in the JES3 global address space. IBM suggests that you perform message processing on the issuing system whenever possible to avoid the system overhead associated with use of these exits.

Choosing the Appropriate Macro to Issue a Message

Most of the time when your DSP or installation exit needs to issue a message, use the MESSAGE macro to issue the message. In addition to MESSAGE, your DSP or installation exit can use other macros to issue messages. The macro to use depends on where you want the message to go, where the message will be issued from (such as which address space), and when the message is to be issued (such as during JES3 initialization). Table 7 shows which macro you should use in a given situation. The table shows macro choices within the JES3 environment only (that is, under the JES3 NUCTASK or AUXTASK).

When using Table 7, remember the following points:

- Each box lists the macros in the order that IBM suggests using them.
- If your code is not part of JES3 (such as a writer FSS), use the WTO macro to issue the message. (Assume that WTOR is the same as WTO for this discussion.)
- The macros used for issuing a message from a C/I DSP should also be used under a C/I FSS.

Table 7. Macros Used for Issuing Messages to Different Destinations

<table>
<thead>
<tr>
<th>Desired Destination</th>
<th>CI DSP</th>
<th>Input Service</th>
<th>Initialization</th>
<th>Recovery/ Termination</th>
<th>SNA/RJP Application</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>JESMSGLG Data Set</td>
<td>IATXIW T</td>
<td>MESSAGE</td>
<td>IATXVMSG</td>
<td>MESSAGE</td>
<td>MESSAGE</td>
<td>MESSAGE</td>
</tr>
<tr>
<td>JESMSG</td>
<td>Notes® 2, 4</td>
<td></td>
<td>n/a</td>
<td>JESMSG</td>
<td>JESMSG</td>
<td>JESMSG</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note 2</td>
<td>Note 2</td>
<td>Note 2</td>
</tr>
<tr>
<td>Calling Console</td>
<td>MESSAGE</td>
<td>MESSAGE</td>
<td>MESSAGE</td>
<td>IATXSNM</td>
<td>MESSAGE</td>
<td>MESSAGE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>IATXVMSG</td>
<td>Note 1</td>
<td>Note 3</td>
<td></td>
</tr>
<tr>
<td>Console Receiving Specific Routing Codes or</td>
<td>MESSAGE</td>
<td>MESSAGE</td>
<td>MESSAGE</td>
<td>IATXSNM</td>
<td>MESSAGE</td>
<td>MESSAGE</td>
</tr>
<tr>
<td>Destination Classes</td>
<td>IATXIWT</td>
<td></td>
<td>MESSAGE</td>
<td>Note 1</td>
<td>Note 3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Note 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hardcopy Log</td>
<td>IATXIWT</td>
<td>MESSAGE</td>
<td>MESSAGE</td>
<td>IATXSNM</td>
<td>MESSAGE</td>
<td>MESSAGE</td>
</tr>
<tr>
<td></td>
<td>MESSAGE</td>
<td></td>
<td>IWASPOUT</td>
<td>Note 1</td>
<td>Note 3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Note 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Console other than Calling Console</td>
<td>MESSAGE</td>
<td>MESSAGE</td>
<td>MESSAGE</td>
<td>IATXSNM</td>
<td>MESSAGE</td>
<td>MESSAGE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Note 1</td>
<td>Note 3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 7. Macros Used for Issuing Messages to Different Destinations (continued)

<table>
<thead>
<tr>
<th>Desired Destination</th>
<th>CI DSP</th>
<th>Input Service</th>
<th>Initialization</th>
<th>Recovery/Termination</th>
<th>SNA/RJP Application</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>JES3OUT Data Set</td>
<td>n/a</td>
<td>n/a</td>
<td>IWASPOUT</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Note 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JES3SNAP Data Set</td>
<td>n/a</td>
<td>n/a</td>
<td>IATXVMSG</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>IATXVSRE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>IATXVSRV</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Note 1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Notes:
1. During JES3 initialization use the MESSAGE macro to send the message to an MCS console. Also, you can use the IWASPOUT macro to send the message to the JES3OUT data set. Note that all critical messages issued by IWASPOUT automatically go to the operator's console and the JES3OUT data set.
2. For JCL or JES3 control card errors, use the JESMSG macro to write a message to a job's JESMSGGL data set.
3. To send a message to a remote workstation (in a SNA/RJP application, for example), use the IATXSNM macro.
4. If you write a DSP that affects JES3 converter/interpreter processing, use the IATXIWT macro.

Guidelines for Issuing Multiline Messages

If a message is longer than the width of the screen where the message is to be displayed, MCS will split the line appropriately and continue the message on the next line. If the message contains a series of related lines, such as a table following a heading line, use the multiline MESSAGE service to issue the message.

Multiline messages do not have their text interrupted by intervening messages on MCS consoles. If the multiline message is an action message, it will be retrievable by the MVS D R command as a single entity.

Writing Installation Modifications for APPC/MVS

Overview
Advanced Program-to-Program Communication (APPC) is an implementation of System Network Architecture (SNA) LU 6.2 protocol that allows interconnected systems to communicate and share the processing of resources. APPC/MVS is a VTAM® application that extends APPC support to the z/OS MVS operating system. APPC/MVS in cooperation with APPC/VTAM provides full LU 6.2 capability to programs running on z/OS MVS. APPC/MVS programs can use certain JES3 functions such as JES3 output service processing.

Application programs that use APPC calls to carry out cross-system requests are called transaction programs. For example, a PS/2 application that needs MVS to print some data would invoke a transaction program on the workstation which communicates with a transaction program residing on the MVS host.

Transaction programs are associated with APPC initiators, which JES3 treats as long-running jobs that can sponsor many transaction programs. APPC initiators invoke JES3 functions on behalf of transaction programs. JES3 provides the following information to installation exits and DSPs:

- A bit (RQAPPC) to indicate that the resident job queue (RESQUEUE) belongs to an APPC initiator
- The RESQUEUE contains certain pointers to various control blocks that contains information for the APPC transaction program
• A data set output information block (DOI)
• A macro (IATXJMR) to obtain accounting information contained in the job management record (JMR).

Writing Installation Exits for APPC

JES3 treats all SYSOUT produced by APPC transaction programs as spin-off data sets. Any exit called during JES3 output service processing (see the table Exits Listed by JES3 Function for a list of exits that are called during output service) may perform modifications to information for transaction program SYSOUT data sets.

All output service exits must test the RQAPPC bit (in field RQFLG12 of IATYRSQ). If the bit is off (set to 0), the data set is not an APPC transaction SYSOUT data set. The exit routine can process the data set as it would any non-APPC SYSOUT request.

If the bit is on (set to 1), the data set is an APPC transaction SYSOUT data set. In this case, only the following control blocks contain information relating to the SYSOUT data set:
• JDS buffers chained from field RQJDSFDB may contain JDS entries for multiple users
• OSE buffers chained from field RQOSEFDB are OSEs for transaction-generated SYSOUT.

The OSE buffer may contain OSEs for more than one userid. Check the OSEUSRID field (in the OSE variable section) to find the owning userid for a particular data set section.

Note: All other RESQUEUE control blocks contain information relating to the APPC initiator itself.

The data set output information block (DOI) contains the owning userid, transaction id, and transaction name for the SYSOUT data set. Check the OSEDOIX flag in field OSEDFLG5. If the bit is on (set to 1), a DOI exists for the data set. For APPC support, the OSEOTSWB is the FDB address where the first logical record is the DOI.

JES3 uses the IATXJMR macro to obtain the JMR for the transaction data set. The JMR contains accounting information for the SYSOUT data set. See the "Introduction" on page 169 for a description of how to use IATXJMR to obtain accounting information.

Additional Considerations for APPC

If your modification directly accesses information through the JES3 resident job queue table (RESQUEUE), you should test the RQAPPC bit (in field RQFLG12 of IATYRSQ). If the bit is on (set to 1), the RESQUEUE is associated with an APPC initiator. Information in the following control blocks represents the APPC initiator itself, not the transaction program which the initiator is sponsoring.
• Job or data set track allocation table (JBT)
• Job description accounting block (JDAB)
• Job summary table (JST)

The JDS and OSEs associated with a transaction initiator may represent data for multiple users. Check the OSEUSRID field in the OSE variable section to find the owning userid of a specific data set section.
The modification may obtain certain information that is specific to the transaction program. For example, accounting information is contained in the job management record (JMR), which can be obtained using the IATXJMR macro.

### Testing a DSP or Installation Exit

To minimize the effect a failing program might have on JES3, use the following procedure before installing the program as a permanent entry in the JES3 library.

Place the program in a private library and use the LNK and SYSP options when the system IPL is performed. Be sure the private library is attached to the processor running the correct level of JES3. Then, if the program fails, perform a re-IPL of the system without the LNK and SYSP options, and the failing program will no longer be in the system.

**Note:** When module IATGRPT is assembled, a number is assigned to each entry in the DSP dictionary. If module IATGRPT is reassembled for migration purposes or to apply maintenance, you should be aware of this DSP numbering scheme. If you have your own DSPs at the end of the DSP dictionary, the numbers assigned to your DSPs will be overwritten by the new IBM DSP when module IATGRPT is reassembled.

### Executing a DSP Under the Auxiliary Task

Normally, DSPs only execute under the JES3 primary (IATNUC) task. However, the writer output multitasking function uses the JES3 auxiliary (IATAUX) task to offload part of the writer DSP processing from the primary task. When the JES3 global main runs on a multiprocessor, a writer operating under the auxiliary task can work in parallel with other JES3 functions. Although the auxiliary task and services available are designed for writer DSP processing, other DSPs can use the auxiliary task if they execute within the limits of the auxiliary task environment. To let a DSP run under the auxiliary task, you must be aware of the following programming considerations:

- You must code the IATXATDE macro in the DSP to create and have available an auxiliary task dispatching element (ATDE) to use the “set mode” (IATXSTMD) service. When the DSP finishes using the “set mode” service, code the IATXATDE macro again to free the ATDE.
- You must code the IATXSTMD macro at appropriate points in the DSP where you want to switch processing from the primary task to the auxiliary task and vice versa.
- The IATXTEST macro can be used at any time to determine the execution mode of the DSP.
- Because a DSP dispatched under the auxiliary task can execute concurrently with DSPs running under the primary task, you must serialize use of the resources shared between your DSP and other DSPs in the system.
- The auxiliary task must be enabled for work either by including the MT=ON parameter on the OPTIONS statement in the initialization stream or by entering the operator command *MODIFY MT=ON.
- The auxiliary task exists only in the JES3 global address space. On a JES3 local main, or in the FSS address space, the IATXATDE and IATXSTMD services immediately return control to the DSP without performing the requested function.
- Not all JES3 services are available under the auxiliary task. Table 8 on page 46 lists service routines that are supported under the auxiliary task.
Table 8. JES3 Services Available Under Auxiliary Task

The auxiliary task supports only a subset of the JES3 internal services. These services fall into the following categories: common services, output services, BSC RJP routines and SNA RJP services. The service routines within each of these groups which are supported under the auxiliary task are listed below. These services are identified by their macro interface if one exists. Otherwise, they are identified by their internal name and module.

<table>
<thead>
<tr>
<th>Common Services</th>
<th>Output Services</th>
<th>BSC RJP Routines</th>
<th>SNARJP Services</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACALL</td>
<td>IATXOSG</td>
<td>IATRJM2: ACMPRINT</td>
<td>IATXCDVE</td>
</tr>
<tr>
<td>AGETMAIN</td>
<td>IATXOSOI</td>
<td>Module IATRJM4: RTIMER</td>
<td>IATXFLCB</td>
</tr>
<tr>
<td>APUTMAIN</td>
<td>IATXOSP</td>
<td>RJPIO</td>
<td>IATXMBFE</td>
</tr>
<tr>
<td>ARETURN</td>
<td></td>
<td>RJPPUT</td>
<td>IATXFSC</td>
</tr>
<tr>
<td>IATXLPJ3</td>
<td></td>
<td></td>
<td>IATXRST</td>
</tr>
<tr>
<td>IATXSCN1</td>
<td></td>
<td></td>
<td>IATXRUPT</td>
</tr>
<tr>
<td>IATXSIO</td>
<td>IATXTRC</td>
<td></td>
<td>IATXSNGR</td>
</tr>
<tr>
<td>IATXSQE</td>
<td>JESTAE</td>
<td></td>
<td>IATXSNLKL</td>
</tr>
<tr>
<td>IATXSTMD</td>
<td></td>
<td></td>
<td>IATXSNM</td>
</tr>
<tr>
<td>IATXTEST</td>
<td></td>
<td></td>
<td>IATXSNST</td>
</tr>
<tr>
<td>IATXTRC</td>
<td></td>
<td></td>
<td>IATXSNTR</td>
</tr>
<tr>
<td>IATXSQE</td>
<td></td>
<td></td>
<td>IATXOSO!</td>
</tr>
</tbody>
</table>

Restrictions or considerations on the use of these services under the auxiliary task are as follows:

- **IATXSIO** – can only be used to do a multi-buffer read. Other types of I/O requests are not supported.
- **AWAIT** – can only be used under the auxiliary task to wait for events that will result in the auxiliary task being posted. Either a single event or list of events can be waited for. For a list of events, the auxiliary task does not support the ECF list management services IATXELA, IATXELD, and IATXELS.
- **ACALL** – the routine called must be capable of running in a multitasking environment. That is, the routine must properly serialize input into resources such as data areas. Also if the primary task and auxiliary task share the routine, it must either be reentrant, or it must serialize its processing so that it cannot be entered concurrently under both tasks.
- **FAILDSP** – can be used under the auxiliary task to end only the currently active FCT that is running under the auxiliary task. On the other hand, when this macro executes under the primary task, the FCT specified can be any FCT. That is, it makes no difference whether it is active or not, nor does it matter whether the FCT is under the primary task or auxiliary task.
- **JESTAE** – JESTAE environments can be created while under the auxiliary task and they will remain in effect when processing switches back to the primary task. However, if a failure occurs, JES3 always invokes the specified JESTAE exits under the primary task (with the FCT in NUCTASK mode) even though the DSP may be running under the auxiliary task at the time of the failure.
- **IATXOSOI** – can be used only to note an input record.
Executing a DSP or Installation Exit Routine in a C/I FSS Address Space

If your installation uses C/I functional subsystem (FSS) address spaces, any installation exit routine that executes during the converter, interpreter, and prescan phases of C/I processing must be able to execute in a C/I FSS address space environment. Any DSP you write that executes in a C/I FSS address space must also be compatible with the C/I FSS address space environment. The C/I FSS address space contains only a subset of the nucleus modules and services that are available in the JES3 address space.

When writing an installation exit routine or DSP to execute in a C/I FSS address space, apply the same rules and procedures as those explained in section “Programming Conventions for Writing DSPs and Installation Exits” on page 19. However, be sure that the routine or DSP uses only those services that are available in a C/I FSS address space.

The following installation exit routines can execute in a C/I FSS address space:

IATUX03 IATUX09
IATUX04 IATUX10
IATUX05 IATUX41
IATUX06

Table 9 shows which macros are available in a C/I address space.

Table 9. JES3 Executable Macros Available in a C/I FSS Address Space

<table>
<thead>
<tr>
<th>Dump Formatting Macro Instructions</th>
<th>ABNCODE</th>
<th>ABNCVDEC</th>
<th>ABNCVHEX</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ABNGET</td>
<td>ABNPUT</td>
<td>ABNVRFY</td>
</tr>
<tr>
<td></td>
<td>IATXMGIT</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Spool Data Management Macro Instructions</th>
<th>ABACKR</th>
<th>ABLOCK</th>
<th>ACLOSE</th>
<th>ADEBLOCK</th>
<th>AGETBUF</th>
<th>ALOCATE</th>
<th>ANOTE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AOPEN</td>
<td>AOPEND</td>
<td>APOINT</td>
<td>APUTBUF</td>
<td>ARELEASE</td>
<td>ATRACK</td>
<td>AWRITE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>IATXCPYF</td>
<td>IATXIXO</td>
<td>IATXRDCH</td>
<td>IATXREL</td>
<td>IATXSAS</td>
<td>IATXSO</td>
<td>IATXUBAL</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>JES3 Checkpoint Macro Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>IATXCKPT</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DSP Program Management Macro Instructions</th>
<th>ACALL</th>
<th>ADELETE</th>
<th>ALOAD</th>
<th>ARETURN</th>
<th>FAILDSP</th>
<th>IATXATF</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>IATXCSF</td>
<td>IATXFRQ</td>
<td>IATXGENF</td>
<td>IATXGFC</td>
<td>IATXGROQ</td>
<td>IATXICA</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>JESTAE</td>
<td>ROTAAAD</td>
<td>ROTADEL</td>
<td>ROTAPUT</td>
<td>VIOLATE</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Resource Control Macro Instructions</th>
<th>ADEQ</th>
<th>AENQ</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ATEST</td>
<td>AWAIT</td>
</tr>
</tbody>
</table>

Chapter 1. JES3 DSPs and Installation Exits 47
<table>
<thead>
<tr>
<th>Storage Management Macro Instructions</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AGETMAIN</td>
<td>IATXBGM</td>
<td>IATXGCL</td>
</tr>
<tr>
<td>APUTMAIN</td>
<td>IATXBPL</td>
<td>IATXRCL</td>
</tr>
<tr>
<td>IATXBFM</td>
<td>IATXDPL</td>
<td>IATXSQE</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Communication Macro Instructions</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CONCNVRT</td>
<td>IATXSCN1</td>
<td>LOGIN</td>
</tr>
<tr>
<td>CONREVRT</td>
<td>IATXSCN2</td>
<td>LOGOUT</td>
</tr>
<tr>
<td>IATXCN5</td>
<td>JESMSG</td>
<td>MESSAGE</td>
</tr>
<tr>
<td>IATXIWT</td>
<td>JSERV</td>
<td>SSISERV</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Job Management Macro Instructions</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>IATXELA</td>
<td>IATXRABP</td>
<td>JDSPOINT</td>
</tr>
<tr>
<td>IATXELD</td>
<td>JSDADD</td>
<td>JDSPUT</td>
</tr>
<tr>
<td>IATXELS</td>
<td>JDSGET</td>
<td>JDSREL</td>
</tr>
<tr>
<td>IATXRABC</td>
<td>JDSHOLD</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>JES3 Initialization Macro Instructions</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ICONVBIN</td>
<td>ISCAN1</td>
<td>ITREAD</td>
</tr>
<tr>
<td>ICONVHEX</td>
<td>ISCAN2</td>
<td>IWASPOUT</td>
</tr>
<tr>
<td>INITMWLE</td>
<td>ISORT</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SNA RJP Macro Instructions</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>IATXDEQ</td>
<td>IATXENQ</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>JES3 Inner Macro Instructions</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>IATXTREG</td>
<td>IATXMEXC</td>
<td>IATZCHK</td>
</tr>
<tr>
<td>IATZCALL</td>
<td>IATZMNOT</td>
<td>IATZTYPE</td>
</tr>
<tr>
<td>IATZLOAD</td>
<td>IATZMOVE</td>
<td>RQTAGEN</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Miscellaneous Macro Instructions</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ACVX</td>
<td>IATXDYT</td>
<td>IATXOD</td>
</tr>
<tr>
<td>ATIME</td>
<td>IATXLPJ3</td>
<td>IATXTRC</td>
</tr>
<tr>
<td>IATXCUE</td>
<td>IATXMPC</td>
<td>MOVEDATA</td>
</tr>
<tr>
<td>IATXCVB</td>
<td>IATXPHEX</td>
<td>MTLB</td>
</tr>
<tr>
<td>IATXCVD</td>
<td>IATXPR</td>
<td>ZEROCORE</td>
</tr>
<tr>
<td>IATXDYH</td>
<td>IATXPTCH</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** Macros issued by a routine that runs in a C/I FSS address space must not allocate track groups from the single track table (STT). Only routines operating in the JES3 global address space can access the STT.

The following other services are available in a C/I FSS address space:
- JES3 ESTAE and ESTAE retry routines
- The generalized subtask (macro IATXCSF)
- DIE and termination routines
- The initialization driver (module IATINIT)
- The JSAM DSP (resident FCT)
- The track allocation DSP (resident FCT)
- The FSS driver DSP (resident FCT)
- The general services DSP (resident FCT)
- The FAILSOFT DSP (resident FCT)
- The WAIT DSP (resident FCT)
- The ABEND formatting routines
- The CI DSP (converter/interpreter and prescan phases only)
- The IO DSP
- The FSS DSP dictionary (module IATGRPTF)
- The FSS TVT (module IATGRVTF)
- Any services accessible from a user address space.

Each C/I FSS address space has its own trace table and storage queue (not shared by the JES3 address space) in private area virtual storage.

For a complete list of all modules contained in the C/I FSS address space nucleus (IATNUCF), see z/OS JES3 Diagnosis Reference.

Installation exits and DSPs executing in C/I FSS address spaces may update a job’s spool files using JSAM or USAM. JSAM should be used when running under an FCT. USAM should be used when running under a subtask. To serialize access to the spool files, most of the job’s control blocks are “locked” in the JES3 global address space before the job is sent to the C/I FSS address space. (Control blocks are locked by turning off the FDBCLOSE bit in the appropriate FDB in the job’s RQ.) Routines in the C/I FSS address space can then update the job’s spool files while routines in the JES3 global address space cannot.

The only control block that is not “locked” is the job track allocation table (TAT). Job TAT updating for a DSP or installation exit routine executing in a C/I FSS address space must be done only from the JES3 global address space. To ensure proper spool space allocation, use the ATRACK macro, which sends the allocation request to the JES3 global address space. Failure to do so could cause duplicate track allocation and may force a cold start.

**Adding and Modifying the DEVICE Initialization Statement**

You can use two edit macros, IATDEVA and IATDEVC, to add or modify DEVICE initialization statements.

**IATDEVA (Adding a DEVICE Statement)**

Your system programmer can use the IATDEVA macro to simplify the task of adding DEVICE statements to the initialization stream. Multiple devices can be included by specifying a list of device numbers or a range of device numbers.

**Note:** It may be more convenient to define a range of devices by coding a single DEVICE statement with the NUMDEV parameter, instead of using the IATDEVA edit macro.

**Rules**

To use this macro, you must specify a ‘model’ DEVICE statement. The model statement is copied with the new device numbers substituted on the JNAME parameter. Type ‘MOD’ on the line of the DEVICE statement you want modeled. A slash (/) in any device number in the ‘model’ DEVICE statement is removed in the new DEVICE statement. All other values specified are copied directly.

Invoke this macro from a TSO edit session by typing:

```
IATDEVA dddd,dddd
```

Adding and Modifying the DEVICE Initialization Statement

You can use two edit macros, IATDEVA and IATDEVC, to add or modify DEVICE initialization statements.

**IATDEVA (Adding a DEVICE Statement)**

Your system programmer can use the IATDEVA macro to simplify the task of adding DEVICE statements to the initialization stream. Multiple devices can be included by specifying a list of device numbers or a range of device numbers.

**Note:** It may be more convenient to define a range of devices by coding a single DEVICE statement with the NUMDEV parameter, instead of using the IATDEVA edit macro.

**Rules**

To use this macro, you must specify a ‘model’ DEVICE statement. The model statement is copied with the new device numbers substituted on the JNAME parameter. Type ‘MOD’ on the line of the DEVICE statement you want modeled. A slash (/) in any device number in the ‘model’ DEVICE statement is removed in the new DEVICE statement. All other values specified are copied directly.

Invoke this macro from a TSO edit session by typing:
Specifies the device number(s) to be included on the additional DEVICE statements.

The IATDEVA macro can change device numbers defined with a slash (/), but you must specify these device numbers to the macro without the slash. For example, /002 or /3380 would be specified as 002 or 3380.

Example

This example shows how to specify the additional DEVICE statements for devices 571 through 572 by typing 'MOD' on the device statement that you want to model. You can specify A (after) or B (before) on the line command to determine where you want the DEVICE statements to go. If you don’t specify either A or B, then the DEVICE statement will be placed directly after the model DEVICE statement.

EDIT -------- USER.COMMON.DECKS(INITDECK) - 01.00 -------------------
COMMAND ====> iatdeva 571-572 SCROLL = 001575

MOD DEVICE,DTYPE=TA93400,JNAME=T570,JUNIT=(570,SY1,S1,ON,570,SY2,S2,ON, 001577 570,SY3,S3,OFF,570,SY4,S4,OFF,570,SY5,S5,OFF,570,SY6,S6,OFF,
001578 570,SY7,S7,OFF,570,SY8,S8,OFF),
001579 XTYPE=(TAPE1,TA),XUNIT=(570,SY1,S1,ON,570,SY2,S2,OFF,570,SY3,S3,OFF,
001580 570,SY4,S4,OFF,570,SY5,S5,OFF,570,SY6,S6,OFF,570,SY7,S7,OFF,
001581 570,SY8,S8,OFF)
A *

Results

The modeled DEVICE statement has been used to generate new DEVICE statements for device numbers 571 through 572:

EDIT -------- USER.COMMON.DECKS(INITDECK) - 01.00 -------------------
COMMAND ====> SCROLL = 001575

001576 DEVICE,DTYPE=TA93400,JNAME=T570,JUNIT=(570,SY1,S1,ON,570,SY2,S2,ON, 001577 570,SY3,S3,OFF,570,SY4,S4,OFF,570,SY5,S5,OFF,570,SY6,S6,OFF,
001578 570,SY7,S7,OFF,570,SY8,S8,OFF),
001579 XTYPE=(TAPE1,TA),XUNIT=(570,SY1,S1,ON,570,SY2,S2,OFF,570,SY3,S3,OFF,
001580 570,SY4,S4,OFF,570,SY5,S5,OFF,570,SY6,S6,OFF,570,SY7,S7,OFF,
001581 570,SY8,S8,OFF)

IATDEVC (Modifying a DEVICE Statement)

Your system programmer can use the IATDEVC macro to modify DEVICE statements in the initialization stream. Processor names are added or deleted from the DEVICE statement in order to create a DEVICE statement that reflects the current MAINPROC definitions.

Note: It may be more convenient to define a device or range of devices similarly on all processors by using *ALL as a processor name on the XUNIT and JUNIT parameters of the DEVICE statement instead of using the IATDEVC edit macro.
Rules
DEVICE statements for multiple devices can be modified by specifying a list of
device numbers or a range of device numbers. A block of DEVICE statements can
be modified by typing ‘CC’ on the first and last lines of the DEVICE statements to
be changed.

The DEVICE statements indicated are modified to reflect all the processors defined
by MAINPROC initialization statements. This means that processors can be both
added or deleted from the indicated DEVICE statements. If you change a DEVICE
statement that has a slash (/) in the device number, the slash remains in the
DEVICE statement. A warning message is issued if adding or deleting a processor
will affect jobs already in the system.

Invoke this macro from a TSO edit session by typing:

```
IATDEVC dddd, dddd-ddd, ?
```

**ddddd**

Specifies the device number(s) on DEVICE statements to be modified.

The IATDEVC macro can change device numbers defined with a slash (/), but
you must specify these device numbers to the macro without the slash. For
example, /002 or /3380 would be specified as 002 or 3380.

Example 1
In this example, devices 570 through 572 will reflect any addition or deletion of
MAINPROC statements in the initialization stream. Here are the MAINPROCS
before adding new processors:

```
EDIT -------- USER.COMMON.DECKS(INITDECK) - 01.00 ------------------- COLUMNS
001476 MAINPROC,NAME=SY1,SYSTEM=JES3...
001477 MAINPROC,NAME=SY2,SYSTEM=JES3...
001478 MAINPROC,NAME=SY3,SYSTEM=JES3...
001479 MAINPROC,NAME=SY4,SYSTEM=JES3...
001480 MAINPROC,NAME=SY5,SYSTEM=JES3...
001481 MAINPROC,NAME=SY6,SYSTEM=JES3...
001482 MAINPROC,NAME=SY7,SYSTEM=JES3...
001483 MAINPROC,NAME=SY8,SYSTEM=JES3...
```

Here are the MAINPROCS after adding new processors:

```
EDIT -------- USER.COMMON.DECKS(INITDECK) - 01.00 ------------------- COLUMNS
001476 MAINPROC,NAME=SY1,SYSTEM=JES3...
001477 MAINPROC,NAME=SY2,SYSTEM=JES3...
001478 MAINPROC,NAME=SY3,SYSTEM=JES3...
001479 MAINPROC,NAME=SY4,SYSTEM=JES3...
001480 MAINPROC,NAME=SY5,SYSTEM=JES3...
001481 MAINPROC,NAME=SY6,SYSTEM=JES3...
001482 MAINPROC,NAME=SY7,SYSTEM=JES3...
001483 MAINPROC,NAME=SY8,SYSTEM=JES3...
001484 MAINPROC,NAME=SY9,SYSTEM=JES3...
001485 MAINPROC,NAME=SY10,SYSTEM=JES3...
001486 MAINPROC,NAME=SY11,SYSTEM=JES3...
001487 MAINPROC,NAME=SY12,SYSTEM=JES3...
001488 MAINPROC,NAME=SY13,SYSTEM=JES3...
001489 MAINPROC,NAME=SY14,SYSTEM=JES3...
001490 MAINPROC,NAME=SY15,SYSTEM=JES3...
001491 MAINPROC,NAME=SY16,SYSTEM=JES3...
```
EDIT -------- USER.COMMON.DECKS(INITDECK) - 01.00 ------------------ COLUMNS
COMMAND ====> iatdevc 570-572 SCROLL =
001583 *---------------------------------------------------------------------*

001584 DEVICE,DTYPE=TA93400,JNAME=T570,JUNIT=(570,SY1,S1,ON,570,SY2,S2,ON,
001585 570,SY3,S3,OFF,570,SY4,S4,OFF,570,SY5,S5,OFF,570,SY6,S6,OFF,570,SY7,S7,OFF,
001586 570,SY8,S8,OFF),
001587 XTYPE=(TAPE1,TA),XUNIT=(570,SY1,S1,ON,570,SY2,S2,OFF,570,SY3,S3,OFF,
001588 570,SY4,S4,OFF,570,SY5,S5,OFF,570,SY6,S6,OFF,570,SY7,S7,OFF,
001589 570,SY8,S8,OFF)
001590 *
001591 DEVICE,DTYPE=TA93400,JNAME=T571,JUNIT=(571,SY1,S1,ON,571,SY2,S2,ON,
001592 571,SY3,S3,OFF,571,SY4,S4,OFF,571,SY5,S5,OFF,571,SY6,S6,OFF,571,SY7,S7,OFF,

Results

SY9 through SY16 are added to the DEVICE statements and the DESTCODEs for these processors are incremented:

EDIT -------- USER.COMMON.DECKS(INITDECK) - 01.00 ------------------ COLUMNS
COMMAND ====> SCROLL =
001575 *---------------------------------------------------------------------*

001576 DEVICE,DTYPE=TA93400,JNAME=T570,JUNIT=(570,SY1,S1,ON,570,SY2,S2,ON,
001577 570,SY3,SY3,OFF,570,SY4,S4,OFF,570,SY5,S5,OFF,570,SY6,S6,OFF,
001578 570,SY7,S7,OFF,570,SY8,S8,OFF,570,SY9,S9,OFF,570,SY10,S10,OFF,
001579 570,SY11,S11,OFF,570,SY12,S12,OFF,570,SY13,S13,OFF,570,SY14,S14,OFF,570,SY15,S15,OFF,
001580 570,SY16,S16,OFF),XTYPE=(TAPE1,TA),XUNIT=(570,SY1,S1,ON,
001581 570,SY2,S2,OFF,570,SY3,S3,OFF,570,SY4,S4,OFF,570,SY5,S5,OFF,
001582 570,SY6,S6,OFF,570,SY7,S7,OFF,570,SY8,S8,OFF,570,SY9,S9,OFF,
001583 570,SY10,S10,OFF,570,SY11,S11,OFF,570,SY12,S12,OFF,
001584 570,SY13,S13,OFF,570,SY14,S14,OFF,570,SY15,S15,OFF,
001585 570,SY16,S16,OFF)
001586 *
001587 DEVICE,DTYPE=TA93400,JNAME=T571,JUNIT=(571,SY1,S1,ON,571,SY2,S2,ON,
001588 571,SY3,SY3,OFF,571,SY4,S4,OFF,571,SY5,S5,OFF,571,SY6,S6,OFF,
001589 571,SY7,S7,OFF,571,SY8,S8,OFF,571,SY9,S9,OFF,571,SY10,S10,OFF,

Example 2

This example will reflect any change to the MAINPROC statements for device 570 when you type 'CC' on the first and last lines of the DEVICE statements you want to change. Here are the MAINPROCS before deleting any processors:

EDIT -------- USER.COMMON.DECKS(INITDECK) - 01.00 ------------------ COLUMNS
COMMAND ====> SCROLL =
004300 *---------------------------------------------------------------------*

004301 MAINPROC,NAME=SY1,SYSTEM=JES3...
004302 MAINPROC,NAME=SY2,SYSTEM=JES3...
004303 MAINPROC,NAME=SY3,SYSTEM=JES3...
004304 MAINPROC,NAME=SY4,SYSTEM=JES3...
004305 MAINPROC,NAME=SY5,SYSTEM=JES3...
004306 MAINPROC,NAME=SY6,SYSTEM=JES3...
004307 MAINPROC,NAME=SY7,SYSTEM=JES3...
004308 MAINPROC,NAME=SY8,SYSTEM=JES3...
004309 MAINPROC,NAME=SY9,SYSTEM=JES3...
004310 MAINPROC,NAME=SY10,SYSTEM=JES3...
004311 MAINPROC,NAME=SY11,SYSTEM=JES3...
004312 MAINPROC,NAME=SY12,SYSTEM=JES3...
004313 MAINPROC,NAME=SY13,SYSTEM=JES3...
004314 MAINPROC,NAME=SY14,SYSTEM=JES3...
004315 MAINPROC,NAME=SY15,SYSTEM=JES3...
004316 MAINPROC,NAME=SY16,SYSTEM=JES3...
Here are the MAINPROCS after deleting processors:

<table>
<thead>
<tr>
<th>Command</th>
<th>SCROLL</th>
</tr>
</thead>
<tbody>
<tr>
<td>USER.COMMON.DECKS(INITDECK) - 01.00 COLLUMNS 0</td>
<td>SCROLL ==</td>
</tr>
<tr>
<td>MAINPROC,NAME=SY1,SYSTEM=JES3...</td>
<td></td>
</tr>
<tr>
<td>MAINPROC,NAME=SY2,SYSTEM=JES3...</td>
<td></td>
</tr>
<tr>
<td>MAINPROC,NAME=SY3,SYSTEM=JES3...</td>
<td></td>
</tr>
<tr>
<td>MAINPROC,NAME=SY4,SYSTEM=JES3...</td>
<td></td>
</tr>
<tr>
<td>MAINPROC,NAME=SY5,SYSTEM=JES3...</td>
<td></td>
</tr>
<tr>
<td>MAINPROC,NAME=SY6,SYSTEM=JES3...</td>
<td></td>
</tr>
<tr>
<td>MAINPROC,NAME=SY7,SYSTEM=JES3...</td>
<td></td>
</tr>
<tr>
<td>MAINPROC,NAME=SY8,SYSTEM=JES3...</td>
<td></td>
</tr>
<tr>
<td>MAINPROC,NAME=SY9,SYSTEM=JES3...</td>
<td></td>
</tr>
<tr>
<td>MAINPROC,NAME=SY10,SYSTEM=JES3...</td>
<td></td>
</tr>
<tr>
<td>MAINPROC,NAME=SY11,SYSTEM=JES3...</td>
<td></td>
</tr>
</tbody>
</table>

**Results**

The DEVICE statement for device 570 has SY12 through SY16 deleted because of changes in the MAINPROC statements:

<table>
<thead>
<tr>
<th>Command</th>
<th>SCROLL</th>
</tr>
</thead>
<tbody>
<tr>
<td>USER.COMMON.DECKS(INITDECK) - 01.00 COLLUMNS 0</td>
<td>SCROLL ==</td>
</tr>
<tr>
<td>DEVICE, DTYPE=TA93400, JNAME=T570, JUNIT=(570,SY1,S1,ON,570,SY2,S2,ON,570,SY3,S3,OFF,570,SY4,S4,OFF,570,SY5,S5,OFF,570,SY6,S6,OFF,570,SY7,S7,OFF,570,SY8,S8,OFF,570,SY9,S9,OFF,570,SY10,S10,OFF,570,SY11,S11,OFF,570,SY12,S12,OFF,570,SY13,S13,OFF,570,SY14,S14,OFF,570,SY15,S15,OFF,570,SY16,S16,OFF)</td>
<td></td>
</tr>
</tbody>
</table>

Modifying Routines in an Output Writer FSS Address Space

Two functions occupy the output writer functional subsystem (FSS) address space:

- The JES3 routines that provide the communication link between the JES3 global address space and the output writer functional subsystem application(s) (FSAs)
- The output writer FSA(s)

The JES3 routines provide no installation exits and no service routines in support of DSPs. Therefore, you cannot execute a DSP in an output writer FSS address space.

The output writer FSA includes several installation exits, which are explained in Print Services Facility™ and Print Utility Program Reference. Use this publication for all information on modifying the output writer FSA.
SNA/NJE Networking

JES3 networking (NJE) allows JES3 users at one location to send jobs to another JES3 location for execution, to send SYSOUT data to another JES3 location for processing, and to send commands and messages to another JES3 location.

Each complex in the network is called a node and is identified by a unique node name. For JES3 complexes, this node name refers to the global processor and all of its local processors; the node name cannot refer to an individual system within the JES3 complex.

When defining a JES3 networking environment, your JES3 complex becomes the home node; the other locations in the network are called remote nodes.

Figure 8 shows an overview of JES3 networking where four JES3 complexes (or nodes) are connected by teleprocessing lines using both SNA features and BSC features.

In Figure 8, your home node is node B. It is directly-connected to nodes A and C, and is indirectly-connected to node D.

Figure 8. Overview of JES3 Networking

The JES3 systems network architecture/network job entry (SNA/NJE) is a JES3 networking facility that permits a JES3 installation to establish SNA/NJE networking sessions with other SNA/NJE nodes. SNA/NJE requires the MVS Bulk Data Transfer (MVS/BDT) facility Version 2 to transfer jobs and SYSOUT streams between nodes.

You can transmit two types of information using SNA/NJE: 1) A network stream containing a job, and 2) a network stream containing data to be printed (such as SYSOUT data).

A network stream containing a job is JCL you want executed at another node. JES3 users initiate JES3 networking jobs by including the /*ROUTE XEQ JECL statement to send MVS JCL or the // XMIT JCL statement to send either MVS or non-MVS JCL to another node for execution.
A network stream containing SYSOUT data can contain information such as memos, reports, or any other information you want to send to another node.

As just described, there are two kinds of network streams – job streams and SYSOUT streams.

A job stream includes the following segments:
1. job header
2. JCL required to process the job
3. SYSIN data
4. job trailer

A SYSOUT stream includes the following segments:
1. job header
2. data set header, where there is one or more data set headers per each SYSOUT data set in the stream.
3. the SYSOUT data set(s)
4. job trailer

Using Installation Exits to Modify Network Data

If you implement some of the JES3 installation exits to modify network data, you may want to verify that the programmers at the other nodes are using the same exits to make similar modifications.

Installation exits 39, 40, and 43 can be used to modify the information in the job headers and data set headers of the network streams sent between nodes. They can also be used to create user sections in the NJE headers constructed by JES3 for outbound NJE work. See "Format of NJE Header User Section" for an example of the format of these user sections.

- Exit 39 modifies the data set header for an outbound SYSOUT stream.
- Exit 40 modifies the job header for an outbound job stream.
- Exit 43 modifies the job header for an outbound SYSOUT stream.

Installation exits 36, 37, 38, and 68 can be used to access and use the information in the user sections of the NJE headers constructed by JES3 for outbound NJE work.

- Exit 36 extracts accounting information from the job header for the JMR.
- Exit 37 modifies the incoming data set header for local SYSOUT processing.
- Exit 38 modifies the job data set (JDS) entry created from the data set header for local SYSOUT processing.
- Exit 68 modifies the local NJE job trailers for a network stream containing SYSOUT.

Format of NJE Header User Section

The following figure shows the format of the user section in the NJE headers constructed by JES3 for outbound NJE work.
In the previous format of the user section:

- Field NxxULEN (2 bytes) is the length of the user section including these first four bytes.
- Field NxxUTYPE (1 byte) indicates the section type. The first two bits of this field must be on (B’11nnnnnn’).
- Field NxxUMOD (1 byte) indicates the section modifier. Any value is permitted.

For each of these fields ‘xx’ can be:

- JH for job header
- DH for data set header
- JT for job trailer

Following these first four bytes, the user section is mapped according to your installation’s requirements. The entire length of the header cannot exceed 4092 bytes. This number includes the 4-byte header prefix and the sections created by JES3. Macros IATYNJH, IATYNDH, and IATYNJT contain skeleton user sections, including equates to use for the TYPE and MOD fields. You can use these mapping macros to help you build your user section.

**Adding a NJE Header User Section**

When you add a NJE header user section, the installation exit must update the total length of the header (NxxLEN) to include the length of the section that you added.
To determine either where to add a user section, or to locate the user section that you received in a header, code your installation exit routine using the following sequence:

1. Obtain the total length of the header (NxxLEN) and subtract four from this value. You can use equates in IATYNJH, IATYNDH, and IATYNJT for the prefix length. This will account for the four-byte prefix.
2. Point to the general section of the header.
3. Map the general section with the user section mapping. This is possible because the first four bytes of all sections are identical.
4. Check the section length (NxxULEN) for a value of zero. If the value is zero, treat this as an error and bypass the rest of the loop.
5. Subtract the section length (NxxULEN) from the total header length.
6. If the result is zero or less, the end of the header has been reached.
7. Add the section length to the section pointer. This will advance you to the next section.
8. If you are looking for a user section, compare the NxxUETYPE and the NxxUMOD fields for the section you are looking for.
9. Go back to step 4 if you do not find the user section.

**Coding Considerations for Installation Exits in 31-Bit Addressing Mode**

JES3 SP™ makes use of 31-bit storage. All JES3 code, including all installation exit routines (except exit 26), but excluding short sections associated with restricted MVS services, runs in 31-bit mode.

This means that JES3 modules, control blocks, and major buffer pools reside above the 16-megabyte address (16Mb address) in the extended private area, and that JES3 common area requirements are in the extended common area of the JES3 address spaces. Also, most of the JES3 I/O drivers use central storage above the 16Mb address. Also, the JSAM buffers, which use the most space in the JES3 address space, reside in expanded storage.

**Adding an Installation Exit**

The IATXCUE macro can be used to generate reentrant installation exit linkage into mainline code. When the module entry point is to be located from the installation exit list mapped by the IATYUXL macro (that is, when EPLOC=UXL is specified on the IATXCUE macro), you must include your exit routine in the list mapped by IATYUXL. You must also reassemble and link-edit the DSP dictionary (modules IATGRUX, IATGRPT and IATGRPTF, if necessary). Also, you must reassemble and link-edit modules IATGRPT and IATGRPTF, which contain the installation exit list, and module IATGRUX, which loads the installation exits. Additionally, the IATYTVT and IATYREG mapping macros are required.

If you use the IATXCUE macro with BALR linkage, the called exit must return to the ERROR return with a simple branch return (+0 after the call), and return to the NORMAL return with a displacement of 4 (+4 after the call). See "Introduction" on page 169 for more information on the IATXCUE macro.
Installation Exits

Installation exits enable an installation to tailor JES3 without having to modify JES3 code. By not modifying JES3 code, you reduce the amount of work necessary to install JES3 maintenance or new JES3 function. Thus, installation exit routines can be very useful in helping you develop an operating system tailored to your needs.

This section discusses the JES3 installation exits available with your JES3 system. IBM supplies a default installation exit routine for each location in JES3 code where an installation exit is defined. You may, of course, write your own installation exit routine in place of any provided by IBM. The information provided with each exit includes:

- CSECT name – The name by which the system recognizes the exit routine.
- Function – The JES3 function from which the exit is taken, and the purpose of the exit.
- Linkage Conventions – Explains whether standard linkage conventions are used by the exit.
- Environment – The address space within which the exit routine runs and, if applicable, under which DSP or subtask the exit routine runs. The term “JES3 global address space” refers to the JES3 address space on the global. The term “JES3 local address space” refers to the JES3 address space on a local. This section also gives the addressing and residency mode for the exit routine.
- Information supplied on entry – The information that is passed to the exit routine upon entry.
- Programming considerations – Details on the information supplied on entry, entry and exit linkage, and special programming considerations to be followed when using the exit.
- Information to be supplied on exit – The information that must be passed back to the calling program when the installation exit finishes. If the description of the installation exit indicates that a return code must be supplied, then be sure that your code sets a return code.

IBM-supplied exit routines are provided with the system. For each exit, either the IBM-supplied routine or an installation exit routine must reside in the JES3 module library, or JES3 issues warning messages IAT3020 and IAT3102.

See [z/OS JES3 Initialization and Tuning Reference](#) for information on how the installation exit routines influence JES3 job management.

**Return Codes for JES3 Installation Exits**

Table 10 identifies the return codes that should be set by the installation exit before returning to JES3. Each installation exit must set the appropriate return code before returning to JES3 to obtain a strict B1 security classification.

If your installation has implemented installation exits, the following chart identifies the installation exits that belong to a specific JES3 function.

<table>
<thead>
<tr>
<th>Installation Exit</th>
<th>IBM-supplied Code in Installation Exit</th>
<th>Type of Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>IATUX03</td>
<td>No</td>
<td>RETURN or BR R14</td>
</tr>
<tr>
<td>IATUX04</td>
<td>No</td>
<td>ARETURN</td>
</tr>
</tbody>
</table>
Table 10. Return Codes for a Strict B1 JES3 Security Classification (continued)

<table>
<thead>
<tr>
<th>IATUX05</th>
<th>No</th>
<th>ARETURN</th>
</tr>
</thead>
<tbody>
<tr>
<td>IATUX06</td>
<td>No</td>
<td>ARETURN</td>
</tr>
<tr>
<td>IATUX07</td>
<td>No</td>
<td>ARETURN</td>
</tr>
<tr>
<td>IATUX08</td>
<td>No</td>
<td>ARETURN</td>
</tr>
<tr>
<td>IATUX09</td>
<td>No</td>
<td>ARETURN</td>
</tr>
<tr>
<td>IATUX10</td>
<td>No</td>
<td>ARETURN</td>
</tr>
<tr>
<td>IATUX11</td>
<td>No</td>
<td>ARETURN</td>
</tr>
<tr>
<td>IATUX14</td>
<td>No</td>
<td>ARETURN RC=04</td>
</tr>
<tr>
<td>IATUX15</td>
<td>No</td>
<td>BR R14</td>
</tr>
<tr>
<td>IATUX17</td>
<td>No</td>
<td>ARETURN</td>
</tr>
<tr>
<td>IATUX18</td>
<td>No</td>
<td>ARETURN</td>
</tr>
<tr>
<td>IATUX19</td>
<td>No</td>
<td>ARETURN RC=4</td>
</tr>
<tr>
<td>IATUX20</td>
<td>Yes</td>
<td>ARETURN RC=4</td>
</tr>
<tr>
<td>IATUX21</td>
<td>Yes</td>
<td>ARETURN RC=4</td>
</tr>
<tr>
<td>IATUX22</td>
<td>No</td>
<td>ARETURN RC=4</td>
</tr>
<tr>
<td>IATUX23</td>
<td>Yes</td>
<td>ARETURN RC=4</td>
</tr>
<tr>
<td>IATUX24</td>
<td>No</td>
<td>ARETURN</td>
</tr>
<tr>
<td>IATUX25</td>
<td>No</td>
<td>ARETURN</td>
</tr>
<tr>
<td>IATUX26</td>
<td>No</td>
<td>BR R14</td>
</tr>
<tr>
<td>IATUX27</td>
<td>No</td>
<td>ARETURN RC=4</td>
</tr>
<tr>
<td>IATUX28</td>
<td>No</td>
<td>ARETURN RC=12</td>
</tr>
<tr>
<td>IATUX29</td>
<td>No</td>
<td>ARETURN RC=8</td>
</tr>
<tr>
<td>IATUX30</td>
<td>Yes</td>
<td>ARETURN</td>
</tr>
<tr>
<td>IATUX32</td>
<td>No</td>
<td>BR R14</td>
</tr>
<tr>
<td>IATUX33</td>
<td>No</td>
<td>ARETURN</td>
</tr>
<tr>
<td>IATUX34</td>
<td>No</td>
<td>ARETURN</td>
</tr>
<tr>
<td>IATUX35</td>
<td>No</td>
<td>ARETURN</td>
</tr>
<tr>
<td>IATUX36</td>
<td>No</td>
<td>ARETURN</td>
</tr>
<tr>
<td>IATUX37</td>
<td>No</td>
<td>ARETURN</td>
</tr>
<tr>
<td>IATUX38</td>
<td>No</td>
<td>ARETURN</td>
</tr>
<tr>
<td>IATUX39</td>
<td>No</td>
<td>ARETURN RC=4</td>
</tr>
<tr>
<td>IATUX40</td>
<td>No</td>
<td>ARETURN RC=4</td>
</tr>
<tr>
<td>IATUX41</td>
<td>No</td>
<td>ARETURN</td>
</tr>
<tr>
<td>IATUX42</td>
<td>No</td>
<td>ARETURN</td>
</tr>
<tr>
<td>IATUX43</td>
<td>No</td>
<td>ARETURN RC=4</td>
</tr>
<tr>
<td>IATUX44</td>
<td>No</td>
<td>ARETURN</td>
</tr>
<tr>
<td>IATUX45</td>
<td>No</td>
<td>ARETURN RC=16</td>
</tr>
<tr>
<td>IATUX46</td>
<td>No</td>
<td>ARETURN</td>
</tr>
<tr>
<td>IATUX48</td>
<td>No</td>
<td>ARETURN</td>
</tr>
<tr>
<td>IATUX49</td>
<td>No</td>
<td>ARETURN</td>
</tr>
<tr>
<td>IATUX50</td>
<td>No</td>
<td>ARETURN</td>
</tr>
</tbody>
</table>
### Table 10. Return Codes for a Strict B1 JES3 Security Classification (continued)

<table>
<thead>
<tr>
<th>Module</th>
<th>Security Classification</th>
</tr>
</thead>
<tbody>
<tr>
<td>IATUX57</td>
<td>No BSM 0,R14</td>
</tr>
<tr>
<td>IATUX58</td>
<td>No BR R14</td>
</tr>
<tr>
<td>IATUX59</td>
<td>No PR</td>
</tr>
<tr>
<td>IATUX61</td>
<td>No ARETURN</td>
</tr>
<tr>
<td>IATUX62</td>
<td>No ARETURN</td>
</tr>
<tr>
<td>IATUX63</td>
<td>No ARETURN RC=0</td>
</tr>
<tr>
<td>IATUX66</td>
<td>No ARETURN RC=8</td>
</tr>
<tr>
<td>IATUX67</td>
<td>No ARETURN</td>
</tr>
<tr>
<td>IATUX68</td>
<td>No ARETURN</td>
</tr>
<tr>
<td>IATUX69</td>
<td>No BR R14</td>
</tr>
<tr>
<td>IATUX70</td>
<td>No BR R14</td>
</tr>
<tr>
<td>IATUX71</td>
<td>No ARETURN</td>
</tr>
<tr>
<td>IATUX72</td>
<td>No ARETURN</td>
</tr>
</tbody>
</table>

### Installation of JES3 Installation Exits

IBM suggests that any modifications to JES3 code or the installation of JES3 exits be performed utilizing the functions of System Modification Program Extended (SMP/E). This requires the preparation of SMP/E control statements and constructs suitable for SMP/E processing. Applying changes in an SMP/E controlled environment prevents down-leveling or the application of release incompatible maintenance.

If the installation uses IATJ3ASM, found in SYS1.SIATSAMP and the application of PTF maintenance changes any macros, the affected modules will automatically be reassembled by SMP/E.

For more information about SMP/E, see z/OS System Modification Program (SMP) System Programmer’s Guide.

### Installation Exits Listed by JES3 Function

Table 11 on page 61 shows a list of JES3 functions with the corresponding mnemonic names for each. The JES3 functions listed are described in the JES3 Logic Library. The mnemonic names for JES3 functions are identified in Volume 1 of the logic library.

The figure also shows which JES3 functions invoke which JES3 installation exits. JES3 installation exits are modules shipped with JES3. The exits are named IATUXnn, where nn is the 2-digit exit number, such as IATUX23.

Further information on the IATUXnn modules is contained in z/OS JES3 Diagnosis Reference. This publication includes brief module descriptions, addressing mode characteristics, and invocation information.

**JES3 Installation Exits Listed by JES3 Function**
### Table 11. JES3 Installation Exits Listed by JES3 Function (with Mnemonic Name)

<table>
<thead>
<tr>
<th>Function</th>
<th>Mnemonic</th>
<th>Installation Exits Called (by number):</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initialization</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dynamic System Interchange</td>
<td>IN/JV</td>
<td>14, 15, 63</td>
</tr>
<tr>
<td>Job Input and Management</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Input Service</td>
<td>IS</td>
<td>17, 24, 27, 28, 29, 33, 34, 40, 44</td>
</tr>
<tr>
<td>Converter/Interpreter Service</td>
<td>II</td>
<td>3, 4, 5, 6, 7, 8, 9, 10, 11, 26, 41, 46, 49</td>
</tr>
<tr>
<td>JQE/JCT Access Method</td>
<td>GRJX</td>
<td></td>
</tr>
<tr>
<td>Job Segment Scheduler</td>
<td>GRJS</td>
<td></td>
</tr>
<tr>
<td>Job Resource Management</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Main Device Scheduling</td>
<td>MD</td>
<td>61, 62, 71</td>
</tr>
<tr>
<td>Locate/Verify</td>
<td>LV</td>
<td>25</td>
</tr>
<tr>
<td>Dynamic Allocation Fast Path</td>
<td>DY</td>
<td></td>
</tr>
<tr>
<td>Job Scheduling Management</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Main Service</td>
<td>MS</td>
<td></td>
</tr>
<tr>
<td>Generalized Main Scheduling</td>
<td>MS</td>
<td></td>
</tr>
<tr>
<td>Workload Management</td>
<td>WL</td>
<td></td>
</tr>
<tr>
<td>Deadline Scheduling</td>
<td>DL</td>
<td></td>
</tr>
<tr>
<td>Dependent Job Control</td>
<td>DC</td>
<td></td>
</tr>
<tr>
<td>Job Output and Termination</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Output Service</td>
<td>OS</td>
<td>19, 20, 21, 22, 23, 39, 43, 45, 60, 72</td>
</tr>
<tr>
<td>Purge</td>
<td>PU</td>
<td></td>
</tr>
<tr>
<td>Complex Management</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Console Services</td>
<td>CN</td>
<td>18, 35, 70</td>
</tr>
<tr>
<td>Inquiry/Modify</td>
<td>IO/MO</td>
<td>48, 72</td>
</tr>
<tr>
<td>Utilities</td>
<td>UT</td>
<td></td>
</tr>
<tr>
<td>JMF</td>
<td>MF</td>
<td></td>
</tr>
<tr>
<td>Dump Job</td>
<td>DJ</td>
<td></td>
</tr>
<tr>
<td>General Routines</td>
<td>GR/GS</td>
<td></td>
</tr>
<tr>
<td>Abnormal Termination/Recovery</td>
<td>AB/FS</td>
<td>27, 30, 58, 59</td>
</tr>
<tr>
<td>Spool Data Management</td>
<td>DM</td>
<td></td>
</tr>
<tr>
<td>JES3 Communication</td>
<td>SI</td>
<td></td>
</tr>
<tr>
<td>Subsystem Interface</td>
<td>FC/FP</td>
<td>32, 57, 69</td>
</tr>
<tr>
<td>Functional Subsystem</td>
<td>SS</td>
<td></td>
</tr>
<tr>
<td>Subsystem Communication</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Remote Processing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>JES3 Networking</td>
<td>NT</td>
<td>35, 36, 37, 38, 39, 40, 42, 43, 66, 67, 68</td>
</tr>
<tr>
<td>BSC RJP</td>
<td>RJ</td>
<td></td>
</tr>
<tr>
<td>SNA RJP</td>
<td>SN</td>
<td></td>
</tr>
<tr>
<td>JES3/BDT Communication Interface</td>
<td>BD</td>
<td>50</td>
</tr>
</tbody>
</table>

**Note:** Some exits appear twice because they called from more than one place.

---

**CSECT Name:** IATUX03 (Examine/Modify Converter/Interpreter Text Created from JCL)

This exit, called from the JES3 accounting routine of the converter/interpreter (module IATIICX), provides you with the converter/interpreter text created from the JCL input. You have the opportunity to examine and modify the text. If the converter/interpreter text contains errors, you can add DSNAME messages using...
Installation Exit 03

the IEFCNMB mapping macro. See [z/OS MVS Installation Exits](#) for information about obtaining and initializing the CNMB. These messages will be printed by the converter. You must obtain storage for these messages; the converter will free the storage.

You can use this exit during C/I processing to allow SWA blocks to be moved into virtual storage above 16 megabytes in the job’s address space. The exit lets you change the setting of bit CNPRSWAA in field CNPRPARM of data area CNPRM (converter parameter list). Bit CNPRSWAA contains the current setting for where SWA is located. The mapping macro for the CNPRM is IEFCNPRM. The CNPRM is contained in the interpreter control table (ICTCNPRM).

On completion, you have the option of either letting the job continue normally or having the job fail.

**Note:** The MVS SMF exit from the converter (IEFUJV) can also be used to set the bit in the CNPRM.

**Environment**

This installation exit routine runs under the C/I subtask. It may be called while the C/I subtask is operating in either the JES3 global address space or a C/I FSS address space.

This exit is called in 31-bit addressing mode. The entry point in register 15 is a 31-bit address.

**Linkage Conventions**

This exit is invoked with a BALR instruction. You must save and restore the registers. To return to the calling module, use either a BR 14 instruction or a RETURN macro. Do not use the ARETURN macro.

**Information Supplied on Entry**

The contents of the registers on entry are:

- **R0** Irrelevant
- **R1** Address of the parameter list.
  - word 1 Address of the C/I text
  - word 2 Address of DSNAME messages (mapped by IEFCNMB)
- **R2-8** Irrelevant
- **R9** Address of the interpreter control table (ICT), which is mapped by IATYICT
- **R10-12** Irrelevant
- **R13** Pointer to CI DSP data area mapped by macro IATYIDD
- **R14** Return address
- **R15** Entry point address for IATUX03

**Programming Considerations**

Register 13 points to data CSECT mapped by IATYIDD, which contains a user save area (IDDUSSAVE) and a user work area (IDDUWORK). Registers 0 through 13 must be saved and restored because this installation exit does not use ASAVE linkage. JES3 macros that use the ASAVE linkage routine, including the IATXIWT macro instruction, must not be issued from this installation exit.

If bit ICTUX03 in flag ICTFLAG3 is turned on in IATUX03 the exit will be treated as a dummy exit.
With the storage management subsystem (SMS), SMS constructs such as STORCLAS, DATACLAS, and MGMTCLAS may or may not appear in the C/I text depending on whether they were specified in the JCL. IATUX03 can be used to add SMS constructs to the C/I text, if they were not specified in the JCL. IBM suggests that, if you want such constructs in the C/I text, you use the automatic class selection (ACS) exits, which are called by SMS processing during the MVS interpretation stage.

Refresh Status

This user exit is not refreshable through the command *MODIFY,X,M=IATxxxx,REFRESH if the user exit is running under a C/I FSS address space.

Information Supplied on Exit

The contents of the registers on exit must be:

- **R0-13**: Irrelevant
- **R14**: Return address
- **R15**: Return codes:
  - **0**: Normal return
  - **4**: JCL limit exceeded

  **Note**: This return code results in an 8FB abend.

  Other: Job fail return

CSECT Name: IATUX04 (Examine the Job Information)

This exit, from the prescan phase of the converter/interpreter (module IATIIPR), provides you with the job information passed to the prescan phase by the C/I compatibility module (IATIICM). The information provided consists of JOB statement data plus the contents of the SMF user field within the current job management record scheduler control block. You have the opportunity of examining this information before processing is begun.

On completion, you have the option of either continuing normally or having the job fail.

You can use this exit during C/I processing of the JOB JCL statement to allow SWA blocks to be moved above 16 megabytes in virtual storage in the job’s address space. The exit lets you change the setting of field IDDGTRC2 (which is ORed with field IDDSWA) in data area IATYIDD. Field IDDSWA contains the current setting for where SWA is located.

Environment

This installation exit routine runs under the CI DSP. It may be called while the CI DSP is operating in either the JES3 global address space or a C/I FSS address space.

This exit is called in 31-bit addressing mode. The entry point in register 15 is a 31-bit address.

Linkage Conventions

This exit routine is invoked with the ACALL macro using ASAVE linkage. To return from the exit, use the ARETURN macro without the RC= parameter.
Information Supplied on Entry

The contents of the registers on entry are:
- **R0-10**: Irrelevant
- **R11**: IATYFCT address
- **R12**: IATYTVT address
- **R13**: Pointer to CI DSP data area mapped by macro IATYIDD
- **R14**: Return address
- **R15**: Entry point address for IATUX04

Programming Considerations

The mapping macro IATYIDD PHASE=PRE,TYPE=DSECT provides addressability to the data area that contains the job information and pointers to the job information (for example, job accounting information and TSO userids).

User error messages can be written to the JESMSG data set using the IATXIWT macro.

Refresh Status

This user exit is not refreshable through the command *MODIFY,X,M=IATxxxx,REFRESH if the user exit is running under a C/I FSS address space.

Information Supplied on Exit

The contents of the registers on exit should be:
- **R0-10**: Irrelevant
- **R11-13**: Same as on entry
- **R14**: Return address
- **R15**: Return codes:
  - **0**: Normal return
  - **Other**: Job fail return

CSECT Name: IATUX05 (Examine the Step Information)

This exit, from the prescan phase of the converter/interpreter (module IATIIPR), provides you with the step information passed to the prescan phase by the C/I compatibility module (IATIICM). The information provided consists of EXEC statement data plus the contents of the SMF user field within the current job management record scheduler control block. You have the opportunity of examining the information before processing is begun.

On completion, you have the option of either continuing normally or having the job fail.

You can use this exit during C/I processing of the EXEC JCL statement to allow SWA blocks to be moved above 16 megabytes in virtual storage in the job’s address space. The exit lets you change the setting of field IDDGTRC2 (which is ORed with field IDDSWA) in data area IATYIDD. Field IDDSWA contains the current setting for where SWA is located.
Environment

This installation exit routine runs under the CI DSP. It may be called while the CI DSP is operating in either the JES3 global address space or a C/I FSS address space.

This exit is called in 31-bit addressing mode. The entry point in register 15 is a 31-bit address.

Linkage Conventions

This exit routine is invoked with the ACALL macro using ASAVE linkage. To return from the exit, use the ARETURN macro without the RC= parameter.

Information Supplied on Entry

The contents of the registers on entry are:

- R0-10: Irrelevant
- R11: IATYFCT address
- R12: IATYTVT address
- R13: Pointer to CI DSP data area mapped by macro IATYIDD
- R14: Return address
- R15: Entry point address for IATUX05

Programming Considerations

The mapping macro IATYIDD PHASE=PRE,TYPE=DSECT provides addressability to the data area that contains the step information and the pointers to the step information (for example, step name, procedure step name, step accounting information, step dependency codes, and region sizes).

User error messages can be written to the JESMSG data set using the IATXIWT macro.

Refresh Status

This user exit is not refreshable through the command *MODIFY,X,M=IATxxxx,REFRESH if the user exit is running under a C/I FSS address space.

Information Supplied on Exit

The contents of the registers on exit should be:

- R0-10: Irrelevant
- R11-13: Same as on entry
- R14: Return address
- R15: Return codes:
  - 0: Normal return
  - Other: Job fail return; this causes the job to be flushed from the system

CSECT Name: IATUX06 (Examine the DD Statement Information)

This exit, from the prescan phase of the converter/interpreter (module IATIIPR), provides you with the DD statement information passed to the prescan phase by the C/I compatibility module (IATIICM). The information provided consists of DD statement data plus the contents of the SMF user field within the current job.
management record scheduler control block. You have the opportunity to examine
the DD information before processing begins.

On completion, you have the option of either continuing normally or having the job
fail.

You can use this exit during C/I processing of the DD JCL statement to allow SWA
blocks to be moved above 16 megabytes in virtual storage in the job’s address
space. The exit lets you change the setting of field IDDGTRC2 (which is ORed with
field IDDSWA) in data area IATYIDD. Field IDDSWA contains the current setting for
where SWA is located.

Environment

This installation exit routine runs under the CI DSP. It may be called while the CI
DSP is operating in either the JES3 global address space or a C/I FSS address
space.

This exit is called in 31-bit addressing mode. The entry point in register 15 is a
31-bit address.

Linkage Conventions

This exit routine is invoked with the ACALL macro using ASAVE linkage. To return
from the exit, use the ARETURN macro without the RC= parameter.

Information Supplied on Entry

The contents of the registers on entry are:

<table>
<thead>
<tr>
<th>Register</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R0-R10</td>
<td>Irrelevant</td>
</tr>
<tr>
<td>R11</td>
<td>IATYFCT address</td>
</tr>
<tr>
<td>R12</td>
<td>IATYTVT address</td>
</tr>
<tr>
<td>R13</td>
<td>Pointer to CI DSP data area mapped by macro IATYIDD</td>
</tr>
<tr>
<td>R14</td>
<td>Return address</td>
</tr>
<tr>
<td>R15</td>
<td>Entry point address for IATUX06</td>
</tr>
</tbody>
</table>

Programming Considerations

The mapping macro IATYIDD PHASE=PRE,TYPE=DSECT provides addressability
to the data area that contains the DD information and the pointers to the DD
information (for example, DDNAME, DSNAME, and UNIT parameters, volume serial
numbers, and flags representing catalog references, volume backward references,
and label types).

User error messages can be written to the JESMSG data set using the IATXIWT
macro.

The DDL (interpreter DD level information) record (macro IATYDDL) contains the
information extracted from the MVS SWA control blocks associated with the DD
statement. For data sets managed by the storage management subsystem (SMS),
the DDL contains an indication that SMS manages the data set
(DDLFLG3=DDLSMS).

If a user has specified LIKE=datasetname on the DD statement, field DDLLIKEP
contains the address of the data set name.
Refresh Status

This user exit is not refreshable through the command
*MODIFY,X,M=IATxxxx,REFRESH if the user exit is running under a C/I FSS
address space.

Information Supplied on Exit

The contents of the registers on exit should be:

<table>
<thead>
<tr>
<th>Register</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R0-10</td>
<td>Irrelevant</td>
</tr>
<tr>
<td>R11-13</td>
<td>Same as on entry</td>
</tr>
<tr>
<td>R14</td>
<td>Return address</td>
</tr>
<tr>
<td>R15</td>
<td>Return codes:</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Other</td>
</tr>
</tbody>
</table>

CSECT Name: IATUX07 (Examine/Substitute Unit Type and Volume Serial Information)

This exit, from the postscan catalog resolution routine of the converter/interpreter
(module IATIIP0X), is taken whenever a cataloged data set could not be found in
the catalog. You have the opportunity of examining the unit type and volume serial
information available and, if necessary, substituting a different volume.

You have the option of either having the job fail or substituting the appropriate unit
and volume information.

Environment

This installation exit routine runs under the POSTSCAN DSP or the CI DSP in
either the JES3 global address space or a C/I FSS address space. If you implement
this exit to run in a C/I FSS address space, be aware that not all JES3 services and
tables are available in the C/I FSS address space. For information about services
available in a C/I FSS address space see Table 9 on page 47.

This exit is called in 31-bit addressing mode. The entry point in register 15 is a
31-bit address.

Linkage Conventions

This exit routine is invoked with the ACALL macro using ASAVE linkage. To return
from the exit, use the ARETURN macro without the RC= parameter.

Information Supplied on Entry

The contents of the registers on entry are:

<table>
<thead>
<tr>
<th>Register</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R0-2</td>
<td>Irrelevant</td>
</tr>
<tr>
<td>R3</td>
<td>Pointer to intermediate job summary table (IATYIJS) entry</td>
</tr>
</tbody>
</table>
| R4       | Pointer to the locate response area (IATYLRS), which contains the
          | volume information that you may change after moving it into your
          | own work area                                              |
| R5-10    | Irrelevant                       |
| R11      | IATYFCT address                  |
| R12      | IATYTVT address                  |
| R13      | Pointer to CI DSP data area mapped by macro IATYIDD  |
| R14      | Return address                   |
| R15      | Entry point address for IATUX07  |
Programming Considerations

If you want the job to continue, you must not modify the locate response area. Instead, you must pass back, in register 1, the address of a work area containing the unit and volume information (mapped by IATYUX07).

The mapping macro IATYIDD PHASE=PST,TYPE=DSECT provides addressability to the data area that contains the locate response pointers.

Because data sets managed by the storage management subsystem (SMS) must be cataloged, and JES3 does not use unit and volume serial information for SMS-managed data sets, IATUX07 cannot be used to provide information if a locate fails for an SMS-managed data set.

User error messages can be written to the JESMSG data set through the IATXIWТ macro instruction.

Refresh Status

This user exit is not refreshable through the command *MODIFY,X,M=IATxxxx,REFRESH if the user exit is running under a C/I FSS address space.

Information Supplied on Exit

The contents of the registers on exit should be:

R0 Irrelevant
R1 Pointer to a work area containing the unit and volume information, as mapped by macro IATYUX07 follows:
   • The control block identifier (4 bytes)
   • The length of the parameter list (2 bytes)
   • Number of unit or volser entries (2 bytes)
   • For each unit or volser entry:
     – Unit type (8 bytes)
     – Number of units required (2 bytes)
     – Number of volumes that follow (2 bytes)
     – Volume serial number(s) (6 bytes)
R2-R13 Same as on entry
R14 Return address
R15 Return codes:

0 The “Dataset Not Found” condition should be treated as an error. The job is failed.

Other The “Dataset Not Found” condition should not be treated as an error. Register 1 contains the address of an output area (mapped by IATYUX07) that contains the unit and volume information.

CSECT Name: IATUX08 (Examine Setup Information)

This exit, called from the postscan phase of the converter/interpreter (module IATIIPN), examines the number of units required by the job, and then decides whether job setup or high-watermark setup should be used for the job.
Storage management subsystem (SMS) data sets are not included in the unit count
summarizes because JES3 does not keep track of unit information for
SMS-managed volumes. Also, SMS-managed volumes and units are not affected by
high-watermark setup.

On completion, you have the option of either failing the job, continuing with the job
setup as specified, or sending the job through high-watermark setup to allocate the
minimum number of devices for the job.

Environment

This installation exit routine runs under the POSTSCAN DSP or the CI DSP in the
JES3 global address space.

This exit is called in 31-bit addressing mode. The entry point in register 15 is a
31-bit address.

Linkage Conventions

This exit routine is invoked with the ACALL macro using ASAVE linkage. To return
from the exit, use the ARETURN macro without the RC= parameter.

Information Supplied on Entry

The contents of the registers on entry are:

- R0-10: Irrelevant
- R11: IATYFCT address
- R12: IATYTVT address
- R13: Pointer to CI DSP data area mapped by macro IATYIDD
- R14: Return address
- R15: Entry point address for IATUX08

Programming Considerations

The mapping macro IATYIDD PHASE=PST,TYPE=DSECT provides addressability
to the data area that contains the setup information.

Refresh Status

This user exit is refreshable through the command
*MODIFY,X,M=IATxxxx,REFRESH.

Information Supplied on Exit

The contents of the registers on exit should be:

- R0-10: Irrelevant
- R11-13: Same as on entry
- R14: Return address
- R15: Return codes:
  - 0: Continue with job setup
  - 4: Send job through high-watermark setup
  - Other: Job fail return

If you also decide to send the job through high-watermark setup (return code of 4 in
register 15), you must specify the type of high-watermark setup required by setting
the IDDHWFLG flag in the CI DSP data area mapped by IATYIDD. Following are
the types of high-watermark setup (HWS) you can specify:
You must set the bit combination in the IDDHWFLG flag (using the appropriate label) that corresponds to the types of HWS required. For example, if you want HWS for tape devices, you should set the IDDHWFLG flag to IDDHWHTAP (X'40'). If you want HWS for all devices, you should set IDDHWFLG.

CSECT Name: IATUX09 (Examine Final Job Status, JST, and JVT)

This exit, called from the postscan phase of the converter/interpreter (module IATIIDR), before the converter/interpreter returns control to the job segment scheduler (JSS) driver (module IATGRJ), allows you to examine the final status of the job's setup requirements.

If a job fails before the postscan phase, this exit is called at the end of the prescan phase.

Information supplied to the user includes the final job status at interpreter completion, and the complete job summary table (JST) and job volume table (JVT), if they exist. You have the opportunity of examining the information before the interpreter relinquishes control.

On completion, you have the option of either continuing normally or failing the job.

Environment

This installation exit routine runs under the POSTSCAN DSP in the JES3 global address space.

This exit is called in 31-bit addressing mode. The entry point in register 15 is a 31-bit address.

Linkage Conventions

This exit routine is invoked with the ACALL macro using ASAVE linkage. To return from the exit, use the ARETURN macro without the RC= parameter.

Information Supplied on Entry

The contents of the registers on entry are:

<table>
<thead>
<tr>
<th>Register</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R0-10</td>
<td>Irrelevant</td>
</tr>
<tr>
<td>R11</td>
<td>IATYFCT address</td>
</tr>
<tr>
<td>R12</td>
<td>IATYTVT address</td>
</tr>
<tr>
<td>R13</td>
<td>Pointer to CI DSP data area mapped by IATYIDD.</td>
</tr>
<tr>
<td>R14</td>
<td>Return address</td>
</tr>
<tr>
<td>R15</td>
<td>Entry point address for IATUX09</td>
</tr>
</tbody>
</table>

Programming Considerations

The mapping macro IATYIDD PHASE=PST,TYPE=DSECT provides addressability to the data area that contains the pointer IDDCJST to the job summary table (JST) and the pointer IDDCJVT to the job volume table (JVT) buffers.
User error messages can be written to the JESMSG data set through the IATXIWT macro.

This exit allows you to modify the JST. The JST entries for SMS-managed data sets have a value different than the JST entries for non SMS-managed data sets. The JSTSMS bit in each JST DD entry specifies whether the entry represents a request that is managed by SMS. The fields for JST DD entries that are different for SMS-managed data sets are:

- **JSTTYPE** – This field may contain a unit type if the unit was specified in the JCL or the catalog.
- **JSTDFLG1** – This field contains the device type code “DASD” because SMS supports only permanently resident DASD.
- **JSTJVTX** – This field is not used for SMS-managed data sets.
- **JSTDFRF1 and JSTDBRF1** – The forward and backward unit chain pointers are always zero because JES3 does not keep track of unit information for SMS-managed data sets.
- **JSTDFRF2 and JSTDBRF2** – The forward and backward unit affinity chain pointers are always zero because JES3 ignores unit affinity for SMS-managed data sets.

Also, because JES3 is unaware of the SMS-managed volumes it requires, JES3 builds no JVT entries for SMS-managed data sets.

JST DD entries are also created for the non-SMS catalogs required by a job as determined by the SMS interpreter/dynamic allocation exit. These JST DD entries have a DD name of “JS3CATLG”.

Modifying JST DD entries for SMS-managed data sets does not have any effect on main device scheduler (MDS) processing because MDS uses these types of entries for data set integrity processing only.

**Refresh Status**

This user exit is refreshable through the command

```
*MODIFY,X,M=IATxxxx,REFRESH.
```

**Information Supplied on Exit**

The contents of the registers on exit should be:

- **R0-10** Irrelevant
- **R11-13** Same as on entry
- **R14** Return address
- **R15** Return codes:
  - 0 Normal return
  - Other Job fail return

**CSECT Name: IATUX10 (Generate a Message)**

This exit, from the converter/interpreter (module IATIIMS), provides a means of generating a message from installation exits IATUX04-IATUX09 and IATUX11 into the JESMSG data set. This exit is taken whenever a user message number request is passed to the message subroutine of the interpreter using the IATXIWT macro instruction, and no message exists in the pre-formatted message table.
Installation Exit 10

On completion, the user has the option of supplying a message pointer to the generated message or indicating that the message request is to be ignored.

Environment

This installation exit routine runs under the POSTSCAN DSP in the JES3 global address space or the CI DSP in the JES3 global address space or C/I FSS address space.

This exit is called in 31-bit addressing mode. The entry point in register 15 is a 31-bit address.

Linkage Conventions

This exit routine is invoked with the ACALL macro using ASAVE linkage. To return from the exit, use the ARETURN macro without the RC= parameter.

Information Supplied on Entry

The contents of the registers on entry should be:

- R0-10: Irrelevant
- R11: IATYFCT address
- R12: IATYTVT address
- R13: Pointer to CI DSP data area mapped by macro IATYIDD
- R14: Return address
- R15: Entry point address for IATUX10

Programming Considerations

Register 13 points to data CSECT mapped by IATYIDD, which contains register 0 (IDDMSAV0) as passed to the message processor, and register 1 (IDDMSAV1) as passed to the message processor. On return from this exit, if a message was supplied, the message subroutine issues the message to the JESMSG data set, the operator (depending on the parameters specified on the IATXIW macro or both). The message number is in the low-order three bytes of IDDMSAV1.

Refresh Status

This user exit is not refreshable through the command *MODIFY,X,M=IATxxx,REFRESH if the user exit is running under a C/I FSS address space.

Information Supplied on Exit

The contents of the registers on exit should be:

- R0: Irrelevant
- R1: Pointer to the generated message (length byte followed by text)
- R2-10: Irrelevant
- R11-13: Same as on entry
- R14: Return address
- R15: Return codes:

  - 0: Message request is to be ignored
  - Other: A message was generated
CSECT Name: IATUX11 (Inhibit Printing of the LOCATE Request/Response)

This exit, from the postscan catalog resolution routine of the converter/interpreter (module IATIIP0), is taken whenever the response to a LOCATE catalog search request has been received. You can use this exit to stop the printing of the LOCATE request message (IAT4401) or the LOCATE response message (IAT4402) on the JESMSG data set. See z/OS JES3 Messages for information on how message IAT4402 differs for SMS-managed data sets.

If an error occurs during the processing of the LOCATE request, module IATIIPOX issues message IAT4405, which identifies the error, instead of message IAT4402.

On completion, you have the option of either continuing normally or failing the job.

Environment

This installation exit routine runs under the POSTSCAN DSP or the CI DSP in either the JES3 global address space or a CI/FSS address space. If you implement this exit to run in a C/I FSS address space, be aware that not all JES3 services and tables are available in the C/I FSS address space.

This exit is called in 31-bit addressing mode. The entry point in register 15 is a 31-bit address.

Linkage Conventions

This exit routine is invoked with the ACALL macro using ASAVE linkage. To return from the exit, use the ARETURN macro without the RC= parameter.

Information Supplied on Entry

The contents of the registers on entry are:

R0-3    Irrelevant
R4      Address of LOCATE response fixed entry (IATYLRS).
R5-10   Irrelevant
R11     IATYFCT address
R12     IATYTVT address
R13     Pointer to CI DSP data area mapped by macro IATYIDD
R14     Return address
R15     Entry point address for IATUX11

Programming Considerations

The mapping macro IATYIDD PHASE=PST,TYPE=DSECT provides addressability to the data area that contains the LOCATE information.

Flag byte IDDLOCF1 in the data area mapped by IATYIDD and contains bits for the LOCATE request (IDDLOCW) and the LOCATE response message (IDDLOCRW). These bits are on when installation exit IATUX11 is entered; on exit, the user must turn the appropriate bit off if the messages are not to be printed.

Flags LRSFLG1 and LRSFLG3 in IATYLRS, pointed to by register 4, reveal if any error occurred. In such an event, message IAT4401 should not be suppressed, because it contains information about the original request.
Installation Exit 11

**Note:** Data returned from the catalog can span multiple non-contiguous locate response buffers.

**Refresh Status**

This user exit is not refreshable through the command *MODIFY,X,M=IATxxxx,REFRESH if the user exit is running under a C/I FSS address space.

**Information Supplied on Exit**

The contents of the registers on exit should be:

- **R0-10** Irrelevant
- **R11-13** Same as on entry
- **R14** Return address
- **R15** Return codes:

  - 0 Write the LOCATE response message as specified in IDDLOCF1 and continue processing
  - 4 Job fail return

**Example 1:** You do not want to print the locate request message IAT4401 and the locate response message IAT4402 for a generation data group (GDG) data set request for all generations, but you do want to print both messages for all other requests and responses.

Check to see if LRSGDG in LRSFLG1 is set.

- If LRSGDG is set, reset IDDLOCW and IDDLOCRW in IDDLOCF1.
- If LRSGDG is not set, return from the exit.

**Example 2:** You do not want to print the locate request message IAT4401 and the locate response message IAT4402 for an SMS data set, but you do want to print both messages for all other requests and responses.

Check to see if LRSSMSDS in LRSFLG3 is set.

- If LRSSMSDS is set, reset IDDLOCW and IDDLOCRW in IDDLOCF1.
- If LRSSMSDS is not set, return from the exit.

**CSECT Name: IATUX14 (Validate Fields in Spool Control Blocks During a JES3 Restart)**

This exit is called by the JES3 modules that verify the contents of spool control blocks during a JES3 restart. You can use this exit to validate fields in IBM-supplied control blocks that you normally would not have access to during a restart.

This exit is called from six JES3 job validation/restart modules:

- **IATINJV** – To determine if the exit should be enabled or disabled
- **IATIIJV** – To validate the PDB (macro IATYPDB)
- **IATDMJV** – To validate the JDS (macro IATYJDS)
- **IATJVDR** – To validate the JCT (IATYJCT), JDAB(IATYJDA), and JMR(IATYJMR)
- **IATMDJV** – To validate the JST (IATYJST), JVT (IATYJVT),
IATOSJV – To validate the FRP (IATYFRP) and OSE (IATYOSE) (The exit is not called if any OSE in the OSE buffer is scheduled for output or if the data set has been processed.)

This exit can be useful to your installation because it can help you restart jobs following a system failure instead of having those jobs failed by the system or canceled by the operator.

This exit can be used as a complement to a job validation/restart routine that you write to verify user spool control blocks from non-standard jobs. See the section "Installation-Written Job Validation and Job Restart Routines" on page 8 for information on writing your own validation/restart routine.

Installation exit 14 can also be used to validate user spool control blocks pointed to by the IBM spool control blocks.

Note: When IATUX14 is called by JES3, JES3 assumes that the control block is valid. It is up to your exit routine to verify that the control block is or is not valid. One method of checking is to verify a field that your installation has set up, such as a destination field.

Environment

This installation exit routine runs in the JES3 global address space.

This exit is called in 31-bit addressing mode. The entry point in register 15 is a 31-bit address.

Linkage Conventions

This exit is invoked with the ACALL macro using ASAVE linkage. To return from this exit, use the ARETURN macro with the RC= parameter.

Information Supplied on Entry

The contents of the registers on entry are:

R0
The identifier of the control block or zeros.

R1
Address of the control block's FDB (or zeros). Note: If either register 0 or 1 contains zeros, it indicates that this is the first time that the exit has been called from IATINJV. IATUX14 should set a return code to indicate if the exit should be enabled or disabled.

R2-10
Irrelevant

R11
Address of the FCT for INJOBVAL

R12
Address of the IATYTVT

R13
Address of the JVD (job validation data - IATYJVD csect) – (Field JVDJWAD in the JVD contains the address of the job's validation/restart work area (IATYJVW))

R14
Return address

R15
Entry point address for IATUX14

Programming Considerations

Use the following guidelines when writing this exit:

- IATUX14 must determine if the control block passed to it is in storage or on the spool data set by checking the "on-spool" bit in the FDB (FDBONSP in IATYFDB).
Installation Exit 14

- IATUX14 must also update the control block if any errors found are correctable. The method used to update a control block depends on which control block it is. Use these guidelines:
  - If the control block is the JCT, turn on bit JVWJCTUP in flag byte JVWSTA3. This will cause module IATINJR to update the JCT.
  - If the control block is the JDS, use the IATXJDS TYPE=WRITE macro to update the JDS.
  - All other control blocks should be updated with the AWRITE macro.
- When IATUX14 is finished with a control block, it must return the control block to JES3 in the form it was received. If the control block was on spool on entry, it must be on spool on exit. If the control block was in storage on entry, it must be in storage on exit.
- If IATUX14 updates a control block that should not be released to spool, then be sure to include the PUTBUF=NO keyword on the AWRITE or IATXJDS TYPE=WRITE macros.
- IATUX14 has addressability to the job’s JVW, but must not destroy the work FDBs used by JES3. These fields are JVWWWKJBT, JVWWKFDB, JVWCHFDB, JVWCHJBT, and JVWCUJBT. IATUX14 can use the work FDBs JVWWWKUSR and JVWCHUSR.
- When IATUX14 finishes running the first time it is called, it must set a return code indicating if it should be enabled or disabled. In register 15 on a first-time call, a value of 00 indicates to enable the exit, a value of 04 indicates to disable the exit (a dummy exit) and for JES3 not to call the exit again until JES3 is restarted. Use the ARETURN RC=4 macro for all normal returns after setting the return code in register 15.
- If IATUX14 finds an incorrect control block, use the ARETURN RC=0 macro to indicate a bad control block. JES3 will then mark the job for deletion.

Refresh Status

This user exit is not refreshable through the command
*MODIFY,X,M=IATxxxx,REFRESH.

Information Supplied on Exit

The contents of the registers on exit should be:
R0-1    Same as on entry
R2-13   Irrelevant
R14     Return address
R15     Return code (explained prior under ‘Programming Considerations’)

CSECT Name: IATUX15 (Scan an Initialization Statement)

This exit is entered from the following environments:
- Operator interface (IATINGS) during a cold, warm, or hot start with refresh.
- Initialization subroutines (IATINRN) during a cold, warm, or hot start with refresh or when the operator issues the *MODIFY,CONFIG command.

This exit permits you to scan an initialization statement immediately after the statement is read and before it is scanned by the system routines.

On completion, you have the opportunity of having the statement processed (as entered or as amended) or having the statement ignored.
**Note:** You can test this installation exit routine using the initialization stream checker (module IATUTIS). If you do, the exit routine must not issue any privileged instructions or a program check will result. For information on how to use that facility, see [z/OS JES3 Initialization and Tuning Reference](#).

**Environment**

When called by JES3, this installation exit routine runs in the JES3 global address space. When being tested by the initialization stream checker, this installation exit routine runs in the user address space.

This exit is called in 31-bit addressing mode. The entry point in register 15 is a 31-bit address.

**Linkage Conventions**

The user of this exit routine must save registers 0 to 13 because this exit does not use ASAVE linkage.

**Information Supplied on Entry**

The contents of the registers on entry are:

- **R0**: Pointer to an area containing: a 2-byte length field followed by the 1- to 8-character operator-entered parameters. **Note:** If register 0 contains zeros, no parameters were specified on the initialization statement.
- **R1**: Address of initialization statement with the high-order bit ‘on’ if entry is from IATINGS
- **R2-12**: Irrelevant
- **R13**: Pointer to 18-word save area
- **R14**: Return address
- **R15**: Entry point address for IATUX15

**Programming Considerations**

Register 0 points to a field containing a 2-byte length indicator, followed by 1 to 8 characters constituting the *string portion* of the ‘P=string’ value entered by the system operator in response to message IAT3012.

When entry is from module IATINGS, bit 0 of register 1 will be set to one. (This module reads from the beginning of the initialization stream and processes DYNALLOC initialization statements. It ceases reading when the first statement is detected that is not one of these two statements and is not a comment statement.)

When entry is from module IATINRN, bit 0 of register 1 is 0.

The initialization statement that triggered the exit from module IATINGS (the first non-comment statement following the last DYNALLOC initialization statement) will be presented to the exit twice. The first time it is presented, bit 0 of register 1 will be set to one.

The DYNALLOC initialization statement, if used, will be presented to the exit twice; the first time is the JES3 processing pass, the second time (R1 bit 0 is off) is for listing purposes on the JES3OUT data set.

**Note:** This exit is called only during a cold, warm, or hot start with refresh, or when the *MODIFY,CONFIG* command is used. IATUX15 is not called during a hot start.
Installation Exit 15

Refresh Status

This user exit is not refreshable through the command
*MODIFY,X,M=IATxxxx,REFRESH.

Information Supplied on Exit

The contents of the registers on exit should be:

<table>
<thead>
<tr>
<th>Register</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R0-1</td>
<td>Same as entry</td>
</tr>
<tr>
<td>R2-13</td>
<td>Irrelevant</td>
</tr>
<tr>
<td>R14</td>
<td>Return address</td>
</tr>
<tr>
<td>R15</td>
<td>Return codes:</td>
</tr>
<tr>
<td></td>
<td>0: Statement is to be processed.</td>
</tr>
<tr>
<td></td>
<td>4: Statement is ignored. Deleted statements are not printed on the JES3OUT data set.</td>
</tr>
</tbody>
</table>

CSECT Name: IATUX17 (Define Set of Scheduler Elements)

This exit, entered at the beginning of input service processing during job statement processing (module IATISJB), permits you to create the initial set of scheduler elements for each job. If this installation exit is not taken, the standard list of scheduler elements—converter/interpreter, main service, output service, and purge—is used.

On completion, you must have defined the required set of scheduler elements.

Environment

This installation exit routine runs in the JES3 global address space.

This exit is called in 31-bit addressing mode. The entry point in register 15 is a 31-bit address.

Linkage Conventions

This exit routine is invoked with the ACALL macro using ASAVE linkage. To return from the exit, use the ARETURN macro without the RC= parameter.

Information Supplied on Entry

The contents of the registers on entry are:

<table>
<thead>
<tr>
<th>Register</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R0-10</td>
<td>Irrelevant</td>
</tr>
<tr>
<td>R11</td>
<td>IATYFCT address</td>
</tr>
<tr>
<td>R12</td>
<td>IATYTVT address</td>
</tr>
<tr>
<td>R13</td>
<td>Pointer to data CSECT IATISDT - mapped by macro IATYISD</td>
</tr>
<tr>
<td>R14</td>
<td>Return address</td>
</tr>
<tr>
<td>R15</td>
<td>Entry point address for IATUX17</td>
</tr>
</tbody>
</table>

Programming Considerations

Register 13 points to data CSECT IATISDT, which contains the input service work area (ISUXWA). Field ISBUXWA contains an address of ISUXWA, which is a 160 byte area available for use by the exit.

The exit defines a list of 8-character DSP names to be used in constructing the standard set of scheduler elements for each job. The list is constructed in the input service work area at location ISUX17L. A maximum of 10 DSP names is permitted. The last DSP name must be followed by a doubleword of X'FF's.
IATUX17 is invoked for the first non-demand select job. It either builds a standard SE list or sets the first byte of the list area to X'FF' to specify that it is a dummy exit. IATUX17 is not entered again, until the next invocation of the input service driver.

**Refresh Status**

This user exit is refreshable through the command

*MODIFY,X,M=IATxxxx,REFRESH.

**Information Supplied on Exit**

The contents of the registers on exit should be:

- **R0-1** Irrelevant
- **R2-13** Same as on entry
- **R14** Return address
- **R15** Irrelevant

On exit, location ISUX17L must contain the list of DSP names to be used.

**CSECT Name: IATUX18 (Command Modification and Authority Validation)**

This exit routine, from JES3 input command processing (module IATCNIA in the CONCMD DSP), allows the user to modify a JES3 command and validate the console’s authority to enter the command. If the operator enters a JES3 command at a console that has been defined as not having a high enough authority level for that command, the command will be rejected. You can use this exit to allow a particular command to be issued from a console whose definition would reject the command.

You can also use this exit to change input command text or to completely process a command within the exit.

If you do not use exit 18, the console authority levels for RJP consoles are as specified on the LEVEL= parameter of the CONSOLE initialization statement, and for MCS consoles by direct conversion using MCS default values (for example, the master console has a JES3 console authority level 15, and a console with MCS system authority has JES3 console authority 5). [z/OS JES3 Initialization and Tuning Reference](https://www.ibm.com/support/knowledgecenter/SSEKH8/rdse734/gjesh_index.html) contains more information about defining console authority levels. [z/OS JES3 Commands](https://www.ibm.com/support/knowledgecenter/SSEKH8/rdse734/gjesh_index.html) explains the default authority levels for MCS consoles.

**Notes on Usage**

If you use installation exit 18, be aware of the following capabilities and restrictions:

- If you code the JES3 INTERCOM macro in your exit routine, be sure that the text of the INTERCOM is not identical to the input text being processed or you will create an infinite loop in your system.
- If you code the MESSAGE macro in your exit routine, you must specify the BUSY= parameter to prevent the CONCMD DSP from entering a possible wait state.
- The AWAIT macro is not allowed in this exit.
- This exit is not used for commands entered from NJE.
- Certain commands (for example, a “directed FREE”) cannot have their authority overridden by IATUX18. Other commands in this category are CALL, START, CANCEL, DSI, DUMP, RETURN.
Installation Exit 18

- If installation exit 18 indicates that the authority decision is to be made by JES3 (return codes 8 or 12), you must also code either installation exit 58 or 59 to prevent JES3 from using the security decision from RACF. See "CSECT Name: IATUX58 (Modify Security Information Before JES3 Security Processing)" on page 131 or "CSECT Name: IATUX59 (Modify Security Information After JES3 Security Processing)" on page 138 for more information.
- Only JES3 commands are presented to this exit.

Note: IBM suggests that you use a security product, such as RACF, for command authorization instead of exit code whenever possible. When exit code is required, MPF command exits should be considered first. Use IATUX18 only for exit code which must run in the JES3 global address space.

Environment
This exit is called in 31-bit addressing mode. The entry point in register 15 is a 31-bit address.

Linkage Conventions
This exit routine is invoked with the ACALL macro using ASAVE linkage. To return from this exit, your exit routine must place the return code in R15, and use the ARETURN macro without the RC= parameter.

Information Supplied on Entry
The contents of the registers on entry are:
- R0 Pointer to the beginning of the command buffer, which contains the authority level of the console, the command text, and the CNDB identifying the command issuer
- R1-9 Irrelevant
- R11 Pointer to the CONCMD FCT
- R12 Pointer to the IATYTVT
- R13 Pointer to the CONSOLES data CSECT (IATYCNC) - mapped by macro IATYCNC
- R14 Return address
- R15 Entry point address for IATUX18

Programming Considerations
When processing commands use information only from the command buffer which is mapped by the IATYCNS macro with the TYPE=INPUT parameter. Use the IATXCNDB service to obtain information from the console destination block which is provided in IATYCNS. The console destination block identifies the console issuing the command.

Refresh Status
This user exit is refreshable through the command *MODIFY,X,M=IATxxx,REFRESH if the exit is running in the global address space.

Information Supplied on Exit
The contents of the registers on exit must be:
- R0-13 Irrelevant
- R14 Return Address
Use the **ARETURN** macro without the **RC=** parameter to specify how JES3 is to process the command. The return code should be loaded into register 15 before returning to **IATCNIA**.

**RC=0** Command is to be processed.

**RC=4** Command is to be rejected; message IAT7130 will be issued.

**RC=8** JES3 performs authority checking for the command. This may include calling RACF if RACF command checking is active. Installation exits IATUX58 or IATUX59 may be used to alter the checking performed by RACF, or to instruct JES3 to use its own command authorization scheme.

**RC=12** JES3 performs authority checking for the command. This may include calling RACF if RACF command checking is active. Installation exits IATUX58 or IATUX59 may be used to alter the checking performed by RACF, or to instruct JES3 to use its own command authorization scheme. Following the processing for this command, the exit is made a dummy exit and is not called again.

**RC=16** The command has been completely processed by the exit. JES3 performs no further processing for the command.

---

**CSECT Name: IATUX19 (Examine/Modify Temporary OSE)**

This exit, from the output service resident driver (module IATOSDO), provides you with the contents of the temporary output service element (OSE) constructed for a SYSOUT data set. You have the opportunity of examining and, if required, changing the information before the OSE is spooled.

This exit is called when the OSE is initially created. It is also called when an OSE is rebuilt as the result of a *MODIFY,U command or a program request which moves output from the hold queue to the writer queue.

When sysout is modified so that it is moved from the hold queue to the writer or BDT queue, the OSE may or may not be rebuilt. IATUX19 will only receive control if the OSE is rebuilt. Use IATUX72 if you need to examine or modify the OSE for all sysout that is moved to the writer queue.

On completion, you have the option of either having or not having the SYSOUT data set processed.

**Note:** Do not use any JES3 executable macros documented in this book (such as JESMSG and IATXOSUP) with IATUX19 except ACALL, ARETURN, and JESTAE for subroutine calls.

---

**Environment**

This installation exit routine runs in the JES3 global address space under the OUTSERV DSP.

This exit is called in 31-bit addressing mode. The entry point in register 15 is a 31-bit address.

**Linkage Conventions**

This exit routine is invoked with the ACALL macro using ASAVE linkage. To return from the exit, use the ARETURN macro with the **RC=** parameter.
Information Supplied on Entry

The contents of the registers on entry are:

- **R0** Irrelevant
- **R1** Pointer to temporary OSE
- **R2-R7** Irrelevant
- **R8** Pointer to JDS
- **R9** Pointer to RESQUEUE - mapped by macro IATYRSQ
- **R10** Irrelevant
- **R11** Address of the OUTSERV FCT
- **R12** Address of the IATYTVT
- **R13** Address of the IATYOSA
- **R14** Return address
- **R15** Entry point address for IATUX19

**Note:** Pointers to the JDS and the resident job queue (RESQUEUE) have also been saved in fields OSAJDSA (JDS) and OSARQA (RESQUEUE) of IATYOSA.

Programming Considerations

Register 1 points to the temporary OSE information. (The OSE defines all the characteristics of the SYSOUT data set, for example, scheduling and setup information.) Because fields changed in the OSE may not be propagated into the related control blocks (such as the JDS), it is the user’s responsibility to determine the effects of changing any fields in the OSE, and to determine if the required results will be achieved.

**Notes:**
1. Do not perform I/O for the spool OSE chain for this job.
2. Do not use the IATXOSUP macro in this installation exit.

Refresh Status

This user exit is refreshable through the command *MODIFY,X,M=IATxxx,REFRESH.*

Information Supplied on Exit

The contents of the registers on exit should be:

- **R0-10** Irrelevant
- **R11-13** Same as on entry
- **R14** Return address
- **R15** Irrelevant

Use the RC= parameter on the ARETURN macro to specify how JES3 is to process the data set as follows:

- **RC=0** No further processing is to be performed for the data set
- **RC=4** The data set is to be placed on the output queue for normal processing.

CSECT Name: IATUX20 (Create and Write Job Headers for Job Output)

This exit, from the output service punch or print request routine (module IATOSPN, IATOSPR, or, for SNA devices, IATOSSN), lets you create and write job header pages or cards that separate the output of one job from another job. The OSE and CSECT IATODWD provided as input contain job and data set information that can be used in the job header.
If you use this exit to examine or modify data written on job header pages, use the 8-character job identifier instead of the 5-digit job number when referring to the job (for example, in messages to the operator).

The IBM-supplied default exit checks the SUPUNITS table to determine if job headers are required (defined on the DEVICE statement in the JES3 initialization stream). If not, the exit returns to the caller. If job headers are required, a page or card, depending on device type, is created and written to the device.

With TSO 2.3.1 or higher, you can specify output distribution information for individual data sets by using TSO/E OUTDES commands or OUTPUT JCL statements. Distribution information for data sets may also be added or deleted using the MVS OUTADD or OUTDEL macros. The default exit creates job headers for those data sets that reference an OUTPUT statement or output descriptor that contains any of the following output distribution keywords:

- NAME
- ROOM
- BUILDING
- DEPT
- ADDRESS
- TITLE

The job header contains the job name, job id, device name, device type, owning userid, print time, and print date. JES3 compresses lines for any value omitted by the user (except for print time, print date, and owning userid, which JES3 fills in).

Environment

This installation exit routine is called only for output writers operating in the JES3 global address space. (For an output writer operating in a functional subsystem address space, installation exit routine IATUX45 allows you to examine and modify job header information.)

This exit is called in 31-bit addressing mode. The entry point in register 15 is a 31-bit address.

Linkage Conventions

This exit routine is invoked with the ACALL macro using ASAVE linkage. To return from the exit, use the ARETURN macro with the RC= parameter.

Information Supplied on Entry

The contents of the registers on entry are:

- R0: WRTSWBP pointer list address, if routing keywords are specified.
- R1: Pointer to OSE
- R2-10: Irrelevant
- R11: IATYFCT address
- R12: IATYTVT address
- R13: Pointer to data CSECT IATODWD - mapped by macro IATYWTR
- R14: Return address
- R15: Entry point address for IATUX20

Programming Considerations

For jobs which represent callable dynamic support programs (DSPs), accounting information can exist that is either set by the ACCT= keyword on the *CALL command or by default on the account initialization statement. See [z/OS JES3]
Initialization and Tuning Reference for more information about the account initialization statement. In either case the installation should receive the callable DSP account information from the job description and accounting block (JDAB), field JDABJBCD.

Note: The JDABJBCD field represents a pseudo job card for the callable DSP and the account information must be parsed off.

IATUX20 uses SWBTUREQ, a scheduler JCL facility (SJF) service, to obtain the output distribution keyword values. The exit tests bit OSESEPT (of field OSEDFLG5) to indicate whether the data set references an OUTPUT statement containing any output distribution keywords. The SWB's for input to the SWBTUREQ service are:

- WTRSWBP - address of the SWBTU pointer list
- WTRSWBN - number of SWBTUs in the pointer list.

See z/OS MVS Programming: Authorized Assembler Services Reference SET-WTO for information on using SWBTUREQ.

Register 13 points to data CSECT IATODWD, which contains the job data (job name, job id, device name, and device type). Register 1 points to IATYOSE, which contains the data set attributes.

This installation exit should issue the following macro instructions: IATXOSOO LABEL=None to open the device, IATXOSP to write data, and IATXOSCO LABEL=None to close the device. If an open or write is rejected, the exit should issue IATXOSCO and return a rejection to the caller. A rejection that is returned from IATXOSP for an AFP printer may indicate that the current data set or job has just been changed. The installation exit should pass the rejection to the caller.

For an AFP printer, the address in the field FCTRQAD should not be explicitly or implicitly used. This RESQUEUE address may not be associated with the job currently active at the channel. Field WTRDRSQ in data CSECT IATODWD contains the address of the RESQUEUE for the job active on the writer. This address must be specified for all macros that require a RESQUEUE address (such as JDSGET) or that use the RESQUEUE contained in FCTRQAD to access other job-related control blocks. For example, the AWRITE macro with a TATPTR parameter specified uses the TAT FDB in the RESQUEUE contained in FCTRQAD.

Upon entry, the ROOSESUP for the RQ in WTRDRSQ is off (set to zero). Upon exit, ROOSESUP should be in the off state also.

The OSE is passed to IATUX20 is a copy of the disk OSE. You can access this information without using the IATXOSUP macro. Changes made to this OSE are not made to the job’s disk OSE chain or in any main storage representation of the disk OSE.

If you code the IATXOSUP macro in this installation exit, follow the step-by-step process in the description for the IATXOSUP macro.

Refresh Status

This user exit is refreshable through the command *MODIFY,X,M=IATxxxx,REFRESH.
Information Supplied on Exit

The contents of the registers on exit should be:

- **R0-10**: Irrelevant
- **R11-13**: Same as on entry
- **R14**: Return address
- **R15**: Irrelevant

Use the **RC=** parameter on the ARETURN macro to indicate ERROR or NORMAL completion as follows:

- **RC=0**: Error return. I/O will be ended by the caller under this condition. In the default exit, this return is used to indicate a reject return was received from an IATXOSP or IATXOSOO macro.
- **RC=4**: Normal completion (either job headers were not requested, or the output is complete)

CSECT Name: **IATUX21** (Create and Write Data Set Headers for Output Data Sets)

This exit, from the output service punch or print request routine (module IATOSPN, IATOSPR, or, for SNA devices, IATOSSN), lets you create and write data set header pages or cards that separate the output for one data set from another data set. The OSE and data CSECT IATODWD provided as input contains job and data set information that can be used in the data set header.

The IBM-supplied default exit checks the SUPUNITS table to determine if data set headers are required (defined on the DEVICE statement in the JES3 initialization stream). If not, the exit returns to the caller. If data set headers are required, a page or card, depending on device type, is created and written to the device.

The data set header contains the data set name, SYSOUT class, and priority. The IBM-supplied exit uses an over-sized block letter pattern when printing the data set header. The full data set name appears in single lines both above and below the block letters.

Environment

This installation exit routine is called only for output writers operating in the JES3 global address space. (For an output writer operating in a functional subsystem address space, installation exit routine IATUX45 allows you to examine and modify data set header information.)

This exit is called in 31-bit addressing mode. The entry point in register 15 is a 31-bit address.

Linkage Conventions

This exit routine is invoked with the ACALL macro using ASAVE linkage. To return from the exit, use the ARETURN macro with the **RC=** parameter.

Information Supplied on Entry

The contents of the registers on entry are:

- **R0**: Irrelevant
- **R1**: Pointer to OSE
- **R2-10**: Irrelevant
- **R11**: IATYFCT address
Installation Exit 21

R12 IATYTVT address
R13 Pointer to data CSECT IATODWD - mapped by macro IATYWTR
R14 Return address
R15 Entry point address for IATUX21

Programming Considerations

Register 13 points to data CSECT IATODWD, which contains the data set data (data set ddname, data set class, and data set priority). Register 1 points to an address mapped by IATYOSE, which contains the data set attributes.

This installation exit should issue the following macro instructions: IATXOS00 LABEL=None to open the device, IATXOSP to write data, and IATXOSCO LABEL=None to close the device. If an open or write is rejected, the exit should issue IATXOSCO and return a rejection to the caller. A rejection that is returned from IATXOSP for an AFP printer may indicate that the current data set or job has just been changed. The installation exit should pass the rejection to the caller.

For an AFP printer, the address in the field FCTRQAD should not be explicitly or implicitly used. This RESQUEUE address may not be associated with the job currently active at the channel. Field WTRDRSQ in data CSECT IATODWD contains the address of the RESQUEUE for the job active on the writer. This address must be specified for all macros that require a RESQUEUE address (such as JDGSET) or that use the RESQUEUE contained in FCTRQAD to access other job-related control blocks. For example, the AWRITE macro with a TATPTR parameter specified uses the TAT FDB in the RESQUEUE contained in FCTRQAD.

Upon entry, the RQOSESUP for the RQ in WTRDRSQ is off (set to zero). Upon exit, RQOSESUP should be in the off state also.

The OSE is passed to IATUX21 is a copy of the disk OSE. You can access this information without using the IATXOSUP macro. Changes made to this OSE are not made to the job's disk OSE chain or in any storage representation of the disk OSE.

If you code the IATXOSUP macro in this installation exit, follow the step-by-step process in the description for the IATXOSUP macro.

The SWBTUREQ service can be used to obtain access to the output distribution keywords.

- NAME
- ROOM
- BUILDING
- DEPT
- ADDRESS
- TITLE

The SWB's for input to the SWBTUREQ service are:

- WTRSWBP - address of the SWBTU pointer list
- WTRSWBN - number of SWBTUs in the pointer list.

See [33](#) for an example of how to use the SWBTUREQ service.

Refresh Status

This user exit is refreshable through the command

*MODIFY,X,M=IATxxxx,REFRESH.*
Information Supplied on Exit

The contents of the registers on exit must be:

- R0-10 Irrelevant
- R11-13 Same as on entry
- R14 Return address
- R15 Irrelevant

Use the RC= parameter on the ARETURN macro to indicate ERROR or NORMAL completion as follows:

- RC=0 Error return. I/O will be ended by the caller under this condition. In the default exit, this return is used to indicate a reject return was received from an IATXOSP or IATXOSCO macro.
- RC=4 Normal completion (either data set headers were not requested, or the output is complete)

CSECT Name: IATUX22 (Examine/Alter the Forms Alignment)

This exit, from the output service print/punch setup routine (module IATOSPS), provides you with the opportunity to alter the forms alignment.

Environment

This installation exit routine is called only for output writers operating in the JES3 global address space. When the printers are running under control of a functional subsystem, this exit is not called for.

This exit is called in 31-bit addressing mode. The entry point in register 15 is a 31-bit address.

Linkage Conventions

This exit routine is invoked with the ACALL macro using ASAVE linkage. To return from the exit, use the ARETURN macro with the RC= parameter.

Information Supplied on Entry

The contents of the registers upon entry are:

- R0 Irrelevant
- R1 Pointer to OSE
- R2-R10 Irrelevant
- R11 IATYFCT address
- R12 IATYTVT address
- R13 Pointer to data CSECT IATODWD - mapped by macro IATYWTR
- R14 Return address
- R15 Entry point address for IATUX22

Programming Considerations

Register 1 points to an address mapped by IATYOSE, which contains the form name and FCB ID, or the carriage tape name. Register 13 points to data CSECT IATODWD, field WTRDSUPO points to the IATYSUP entry for the device.

If you do not create any new forms alignment through this exit, processing continues with the current forms alignment.

To issue an action message to the SNA work station console from IATUX22, the following procedure is required:
This procedure is required regardless of the type of work station console (real or simulated). The SUSPEND call means specifically that the writer is about to issue an action message. Therefore, it should not be used for normal (non-action) messages.

The SUSPEND call is a special form of close where a prior open is not required, nor a subsequent open if a prior open has been issued. The macro will never return a rejection. In addition, the macro will loop if the writer does not have an SNA device.

Upon entry, the RQOSESUP for the RQ in WTRDRSQ is off (set to zero). Upon exit, RQOSESUP should be in the off state also.

The OSE is passed to IATUX22 is a copy of the disk OSE. You can access this information without using the IATXOSUP macro. Changes made to this OSE are not made to the job's disk OSE chain or in any storage representation of the disk OSE.

If you code the IATXOSUP macro in this installation exit, follow the step-by-step process in the description for the IATXOSUP macro.

**Refresh Status**

This user exit is refreshable through the command *

*MODIFY,X,M=IATxxxx,REFRESH.

**Information Supplied on Exit**

The contents of the registers on exit should be:

- **R0-10**: Irrelevant
- **R11-13**: Same as on entry
- **R14**: Return address
- **R15**: Irrelevant

Use the RC= parameter on the ARETURN macro to specify how JES3 is to process the data set as follows:

- **RC=0**: Indicates that a new forms alignment is required.
- **RC=4**: Normal completion (indicating that the current forms alignment should be continued)

**CSECT Name: IATUX23 (Create and Write Job Trailers for Job Output)**

This exit, from the output service punch or print request routine (module IATOSPN, IATOSPR, or, for SNA devices, IATOSSN), lets you create and write job trailer pages or cards that separate the output of one job from another job. The data CSECT IATODWD provided as input contains job and data set information that can be used in the job trailer.

IBM suggests that, if you use this exit to examine or modify data written on job trailer pages, you use the 8-character job id instead of the 5-digit job number when referring to the job (for example, when in messages to the operator).

The IBM-supplied default exit checks the SUPUNITS table to determine if job trailers are required (defined on the DEVICE statement in the JES3 initialization...
stream). If not, the exit returns to the caller. If job trailers are required, a page or card, depending on device type, is created and written to the device.

The job trailer contains the job name, job number, and the characters “JOB END”. The IBM-supplied exit uses an over-sized block letter pattern when printing the job trailer information.

Environment

This installation exit routine is called only for output writers operating in the JES3 global address space. (For an output writer operating in a functional subsystem address space, installation exit routine IATUX45 allows you to examine and modify data written to trailer pages.)

This exit is called in 31-bit addressing mode. The entry point in register 15 is a 31-bit address.

Linkage Conventions

This exit routine is invoked with the ACALL macro using ASAVE linkage. To return from the exit, use the ARETURN macro with the RC= parameter.

Information Supplied on Entry

The contents of the registers on entry are:

<table>
<thead>
<tr>
<th>Register</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R0-R10</td>
<td>Irrelevant</td>
</tr>
<tr>
<td>R11</td>
<td>IATYFCT address</td>
</tr>
<tr>
<td>R12</td>
<td>IATYTVT address</td>
</tr>
<tr>
<td>R13</td>
<td>Pointer to data CSECT IATODWD - mapped by macro IATYWTR</td>
</tr>
<tr>
<td>R14</td>
<td>Return address</td>
</tr>
<tr>
<td>R15</td>
<td>Entry point address for IATUX23</td>
</tr>
</tbody>
</table>

Programming Considerations

Register 13 points to data CSECT IATODWD, which contains the job data (WTRDJNAM-job name, WTRDJID-job number and WTRDRSQ-RQ address).

Module IATUX23 is organized into three sections of code: 1. the print trailer routine, 2. the punch trailer routine, and 3. the block-letter creation subroutine. On entry to one of these three sections of code, the base register (register 10) is loaded with the address of the section of code. Also, references to data not within a section of code are done with a LOAD ADCON instruction, for example, L Rx,=A(UU23IOS).

To issue an action message to the SNA work station console from IATUX23, the following procedure is required:

```
IATXOSCO LABEL=SUSPEND,REJECT=MSG01
MSG01 MESSAGE ACTION=YES,...
```

This procedure is required regardless of the type of work station console (real or simulated). The SUSPEND call means specifically that the writer is about to issue an action message. Therefore, it should not be used for normal (non-action) messages.

The SUSPEND call is a special form of close where a prior open is not required, nor a subsequent open if a prior open has been issued. The macro will never return a rejection. In addition, the macro will loop if the writer does not have a SNA device.
This installation exit should issue the following macro instructions: IATXOSOO LABEL=NONE to open the device, IATXOSP to write data, and IATXOSCO LABEL=NONE to close the device. If an open or write is rejected, the exit should issue IATXOSCO and return a rejection to the caller. A rejection that is returned from IATXOSP for any non FSS printer may indicate that the current data set or job has just been changed. The installation exit should pass the rejection to the caller.

For an AFP printer, the address in the field FCTRQAD should not be explicitly or implicitly used. This RESQUEUE address may not be associated with the job currently active at the channel. Field WTRDRSQ in data CSECT IATODWD contains the address of the RESQUEUE for the job active on the writer. This address must be specified for all macros that require a RESQUEUE address (such as JDSGET) or that use the RESQUEUE contained in FCTRQAD to access other job-related control blocks. For example, the AWRITE macro with a TATPTR parameter specified uses the TAT FDB in the RESQUEUE contained in FCTRQAD.

Upon entry, the RQOSESUP for the RQ in WTRDRSQ is off (set to zero). Upon exit, RQOSESUP should be in the off state also.

If you code the IATXOSUP macro in this installation exit, follow the step-by-step process in the description for the IATXOSUP macro.

**Refresh Status**

This user exit is refreshable through the command

 *MODIFY,X,M=IATxxxx,REFRESH.*

**Information Supplied on Exit**

The contents of the registers on exit should be:

- **R0-10** Irrelevant
- **R11-13** Same as on entry
- **R14** Return address
- **R15** Irrelevant

Use the **RC=** parameter on the ARETURN macro to indicate ERROR or NORMAL completion as follows:

- **RC=0** Error return. I/O will be ended by the caller under this condition. In the default exit, this return is used to indicate a reject return was received from an IATXOSP, IATXOSOO or IATXOSCO macro.
- **RC=4** Normal completion (either job trailers were not requested, or the output is complete)

**CSECT Name: IATUX24 (Examine the Net Id and the Devices Requested)**

This exit, from input service processing (module IATISNT), is provided at the time a dependent job network device dedication request is found on a /*NET control statement. You have the opportunity of examining the net id and the devices requested.

On completion, you have the option of either allowing or denying the request.

**Environment**

This installation exit routine runs in the JES3 global address space.
This exit is called in 31-bit addressing mode. The entry point in register 15 is a 31-bit address.

**Linkage Conventions**

This exit routine is invoked with the ACALL macro using ASAVE linkage. To return from the exit, use the ARETURN macro without the RC= parameter.

**Information Supplied on Entry**

The contents of the registers on entry are:

- **R0-4**: Irrelevant
- **R5**: Pointer to IATYDFC, the device fence control block
- **R6**: Address of DFCENTRY, which contains device type and number
- **R7**: Pointer to IATYJNT, the job net control block
- **R8**: Irrelevant
- **R9**: Pointer to IATYNET, the work area for IATISNT
- **R10**: Irrelevant
- **R11**: IATYFCT address
- **R12**: IATYTVT address
- **R13**: Pointer to data CSECT IATISDT - mapped by macro IATYSID
- **R14**: Return address
- **R15**: Entry point address for IATUX24

**Programming Considerations**

Register 13 points to data CSECT IATISDT, which contains the input service work area (ISUXWA).

**Refresh Status**

This user exit is refreshable through the command

```
*MODIFY,X,M=IATxxxx,REFRESH.
```

**Information Supplied on Exit**

The contents of the registers on exit should be:

- **R0-10**: Irrelevant
- **R11-13**: Same as on entry
- **R14**: Return address
- **R15**: Return codes:

  0: The request is allowed.

  4: The request is denied. The job will continue processing, but no fence devices will be allocated.

**CSECT Name: IATUX25 (Examine/Modify Volume Serial Number)**

This exit, called from module IATLVVR, allows you to verify non-standard label tape volumes. If the volume label examined is incorrect, you can specify the serial number of the volume that was actually mounted.

On completion, you have the option of specifying that the correct or wrong volume was mounted.
Environment

This installation exit routine can run in the JES3 global address space or in a JES3 local address space.

This exit is called in 31-bit addressing mode. The entry point in register 15 is a 31-bit address.

Linkage Conventions

This exit routine is invoked with the ACALL macro using ASAVE linkage. To return from the exit, use the ARETURN macro without the RC= parameter.

Information Supplied on Entry

The contents of the registers on entry are:

\begin{verbatim}
R0    Irrelevant
R1    Pointer to an input parameter area - mapped by NSLPARM DSECT in IATUX25
R2-10 Irrelevant
R11   IATYFCT address
R12   IATYTVT address
R13   Irrelevant
R14   Return address
R15   Entry point address for IATUX25
\end{verbatim}

Programming Considerations

Register 1 points to an input parameter area, which contains:

- `NSLRDA` – Pointer to data area
- `NSLVOL` – Pointer to volume to be verified
- `NSLFLAG` – Indicates that EOF was read off tape (instead of data)
- `NSLLAB` – Correct label type returned by this installation exit
  \begin{verbatim}
  N = NO LABEL
  S = STANDARD LABEL
  X = NON-STANDARD LABEL
  \end{verbatim}
- `NSLSER` – Correct volume serial returned by this installation exit

In `NSLSER`, you should indicate the actual volume that was incorrectly mounted. This volume serial will be used to construct a message to the operator indicating that the wrong volume was mounted.

Refresh Status

This user exit is refreshable through the command

\texttt{*MODIFY,X,M=IATxxxx,REFRESH} if the exit is running in the global address space.

Information Supplied on Exit

The contents of the registers on exit should be:

\begin{verbatim}
R0-1  Irrelevant
R2-13 Same as on entry
R14   Return address
R15   Return codes:
       0    Correct volume is mounted.
       4    Wrong volume is mounted.
\end{verbatim}
If a return code other than 0 or 4 is returned, this exit is assumed to be the dummy exit and is not called again until JES3 is restarted.

**CSECT Name: IATUX26 (Examine MVS Scheduler Control Blocks)**

This exit, from interpreter/initiator compatibility processing (that is, during job initiation) (module IATIII), permits you to scan all of the MVS scheduler control blocks before they are moved to the scheduler work area (SWA) in the user’s address space. (The scheduler control blocks were built during converter/interpreter service.) You have the opportunity to examine the data before processing continues.

On completion, you have the option of either continuing normally or having the job fail.

This exit can be used to allow SWA control blocks to reside in virtual storage above 16 megabytes. However, this process can only be done the first time the exit is called for a job. In other words, for the first SWA block for the job this exit can be used to modify the indicator for moving SWA above 16 megabytes. (Any attempt to modify the indicator following the first call to this exit will be ignored by JES3.)

The SWA-above-16-megabytes indicator is the IIWSWABV bit in the IIW data area. If the IIWSWABV bit is set on, SWA is allowed to be placed above 16 megabytes in the job’s address space.

**Environment**

This installation exit routine runs only in the user’s address space.

This exit is called in 31-bit addressing mode and runs in 31-bit mode. This exit has an RMODE of 24. This exit runs in the initiator’s address space, not in the JES3 address space.

To determine whether control blocks reside above or below 16 megabytes, see the list in [z/OS JES3 Diagnosis Reference](#).

**Linkage Conventions**

This exit routine does not use ASAVE linkage conventions. You must save and restore all registers except register 15.

Register 9 points to a data area mapped by IATYIIW, which contains a user save area (IIWUSAVE).

**Information Supplied on Entry**

The contents of the registers on entry are:

- **R0-4**: Irrelevant
- **R5**: Pointer to control block data
- **R6-8**: Irrelevant
- **R9**: Pointer to data area IATYIIW
- **R10-13**: Irrelevant
- **R14**: Return address
- **R15**: Entry point address for IATUX26

**Programming Considerations**

Register 5 points to the SWA header and the scheduler control block.
Installation Exit 26

The mapping macro IATYVSR can be used to map the SWA header and scheduler control block displacements.

Although subpool 127 is a user subpool, this subpool must not be used by installation exits before the problem program being attached. Device allocation requires the use of subpool 127 in key 1. If the subpool is used before the problem program with a key of other than 1, device allocation will incur a program check.

Refresh Status

This user exit is not refreshable through the command
*MODIFY,X,M=IATxxxx,REFRESH.

Information Supplied on Exit

The contents of the registers on exit should be:

<table>
<thead>
<tr>
<th>Register</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R0-13</td>
<td>Same as on entry</td>
</tr>
<tr>
<td>R14</td>
<td>Return address</td>
</tr>
<tr>
<td>R15</td>
<td>Return codes:</td>
</tr>
<tr>
<td></td>
<td>0               Normal return</td>
</tr>
<tr>
<td></td>
<td>Other           Job fail return</td>
</tr>
</tbody>
</table>

CSECT Name: IATUX27 (Examine/Alter the JDAB, JCT, and JMR)

This exit, from *CALL DSP command processing (module IATGRCD), allows you to access and alter the job description accounting block (JDAB), job control table (JCT), and job management record (JMR). For example, your installation may have accounting parameters in the JDAB that you need to modify for called DSPs. Or, you may want to access the JMR to modify an SMF 26 record.

IATUX27 can be used to control the writing of, or to modify, the dummy SMF 26 records built by IATGRCD for callable DSPs.

JES3 normally skips writing dummy SMF 26 records for DSPs that have been called. JES3 module IATGRCD sets the JDABNSMF flag which prevents the writing of the dummy SMF 26 records by purge processing module IATPURG.

If your installation needs the SMF 26 records built for callable DSPs, you can use IATUX27 to modify the records, and to cause the records to be written later during JES3 purge processing. To write the SMF 26 records, reset flag JDABNSMF in field JDABFLG2 of the JDAB from one to zero.

On completion, you have the option of having the *CALL command process normally or having the *CALL command be rejected.

Environment

This installation exit routine runs in the JES3 global address space.

This exit is called in 31-bit addressing mode. The entry point in register 15 is a 31-bit address.

Linkage Conventions

This exit routine is invoked with the ACALL macro using ASAVE linkage. To return from the exit, use the ARETURN macro with the RC= parameter.
Information Supplied on Entry

The contents of the registers on entry are:

<table>
<thead>
<tr>
<th>Register</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R0</td>
<td>Irrelevant</td>
</tr>
<tr>
<td>R1</td>
<td>Pointer to a parameter list</td>
</tr>
<tr>
<td>R2-10</td>
<td>Irrelevant</td>
</tr>
<tr>
<td>R11</td>
<td>IATYFCT address</td>
</tr>
<tr>
<td>R12</td>
<td>IATYTVT address</td>
</tr>
<tr>
<td>R13</td>
<td>Irrelevant</td>
</tr>
<tr>
<td>R14</td>
<td>Return address</td>
</tr>
<tr>
<td>R15</td>
<td>Entry point address for IATUX27</td>
</tr>
</tbody>
</table>

Programming Considerations

Register 1 points to three fullwords: the address of the JDAB, the address of the JCT, and the address of the JMR.

Refresh Status

This user exit is refreshable through the command *
MODIFY,X,M=IATxxxx,REFRESH.

Information Supplied on Exit

The contents of the registers on exit should be:

<table>
<thead>
<tr>
<th>Register</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R0-13</td>
<td>Irrelevant</td>
</tr>
<tr>
<td>R14</td>
<td>Return address</td>
</tr>
<tr>
<td>R15</td>
<td>Return codes:</td>
</tr>
<tr>
<td>0</td>
<td>*CALL is rejected</td>
</tr>
<tr>
<td>4</td>
<td>*CALL is to be processed normally</td>
</tr>
</tbody>
</table>

CSECT Name: IATUX28 (Examine the JOB JCL Statement)

This exit, entered from module IATISJB immediately after the JOB JCL statement is processed and before input service processing, provides access to the JOB statement data and the job management record.

On completion, you have the option of either continuing normally or having the job fail.

Environment

This installation exit routine runs in the JES3 global address space.

This exit is called in 31-bit addressing mode. The entry point in register 15 is a 31-bit address.

Linkage Conventions

This exit routine is invoked with the ACALL macro using ASAVE linkage. To return from the exit, use the ARETURN macro without the RC= parameter.

Information Supplied on Entry

The contents of the registers on entry are:

<table>
<thead>
<tr>
<th>Register</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R0-10</td>
<td>Irrelevant</td>
</tr>
<tr>
<td>R11</td>
<td>IATYFCT address</td>
</tr>
<tr>
<td>R12</td>
<td>IATYTVT address</td>
</tr>
</tbody>
</table>
Installation Exit 28

R13 Pointer to the input service area (data CSECT IATISDT) - mapped by macro IATYISD
R14 Return address
R15 Entry point address for IATUX28

Programming Considerations

Register 13 points to data CSECT IATISDT, which contains:

- LOCADDR – Address of JOB statement data
- JMRADDR – Address of job management record - mapped by macro IATYJMR
- ISFLAG1 – Contains X'F0' if JOB statement is a continuation
- ISFLAG11 – Contains X'04' (ISTSOSUB) if the origin of the job is a TSO submit.

Refresh Status

This user exit is refreshable through the command

*MODIFY,X,M=IATxxxx,REFRESH.

Information Supplied on Exit

The contents of the registers on exit should be:

- R0-1 Irrelevant
- R2-13 Same as on entry
- R14 Return address
- R15 Return codes:
  - 0 Accept the job
  - 4 Flush the job, but create an SMF record
  - 8 Flush the job, and do not create an SMF record
  - 12 Dummy the exit until the next invocation of the input service driver, dummy user exit causes job to be processed normally.

CSECT Name: IATUX29 (Examine the Accounting Information)

This exit, entered at the completion of input service processing (module IATISEN), provides access to the job control table (JCT), job description accounting block (JDAB), and job management record (JMR) for the job. This exit should be used to move the accounting information to the SMF record. You can examine and modify the job accounting information in the JDAB, the CIPARM list (field JDABPMID in the JDAB), JESMSGLG logging (JCTNOJLG flag), and output limits and output options.

'The setting of the JCTNOJLG flag depends on the JESMSG= keyword specification on the CLASS, STANDARDS, or both of the initialization statements. If this keyword is not used in these statements, the default setting is obtained from the STANDARDS initialization statement. Setting the JCTNOJLG flag overrides the default specification on the STANDARDS JESMSG parameter in the JES3 initialization stream. Setting the JCTNOJLG flag prevents logging of most messages in the JESMSGLG data set. JES3 logs some messages primarily for diagnostics purposes.

The CIPARM list is defined with the CIPARM initialization statement. The intent of this exit is to allow you to specify a CIPARM with an indicator of SWA above or below 16 megabytes. See restrictions below for further information.
For jobs that are started tasks, the exit needs to access the job’s JDAB. The job is considered to be a started task if it is a demand select job (the IDSEL bit in ISFALG9 is on) and is not an MVS/TSO logon (bit ISMVSTSO in ISFLAG10 is off). Both flags are referenced from the input service data area (data CSECT IATISDT). The job’s JDAB job output limit fields can be changed. They are:

JDABMAXC   Job cards
JDABMAXL   Job lines
JDABMAXB   Job bytes
JDABMAXP   Job pages

The job limit flag JDABLIMF, job cancel options, can also be changed. On entry, the job’s JDAB contains output limits and cancel options which are either JES3 defaults or values provided by the JES3 STANDARDS initialization statement. You need to consider that output limits are in hundreds for cards and thousands for lines and bytes when coding this exit.

Note: These values must be in the range defined by the JES3 STANDARDS initialization statement and the //*MAIN JES3 JCL statement. See z/OS JES3 Initialization and Tuning Reference and also z/OS MVS JCL Reference.

Dependencies

You can use this exit to set the bit, JCTSBOWN, in JCTFL7 of the job control table (JCT). This bit allows you to determine who is permitted to control the job’s output with STATUS, CANCEL, and OUTPUT commands. The bit can have the following settings:

<table>
<thead>
<tr>
<th>JCTSBOWN Value</th>
<th>Access By</th>
<th>Origin</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (default)</td>
<td>Job Owner (JCTOUSID)</td>
<td>“USER=“ parameter of the JOB card</td>
</tr>
<tr>
<td>1</td>
<td>Job Submitter (JCTTUSID)</td>
<td>The TSO user ID submitting the job</td>
</tr>
</tbody>
</table>

Before setting JCTSBOWN, you need to inspect the JCT fields JCTOUSID, JCTTUSID, or both for valid values.

JCTTUSID contains the user id (for instance, the TSO or FTP user id) of the user who submitted the job. For jobs submitted through the internal reader by batch jobs, the submitting user id of the batch job is propagated to the JCTTUSID field of the submitted job. Thus, if USER1 submits JOBA using TSO SUBMIT, and JOBA submits JOBB using SYSOUT=(A,INTRDR), JCTTUSID contains USER1 for both JOBA and JOBB.

Notes:

1. You can specify the submitting user id on the USER= parameter of the //*MAIN JES3 JCL statement. This value overrides the actual submitting user id.
2. You must code the exit, IATUX30, to check the setting of JCTSBOWN, if it is to have any effect on the processing of the STATUS, CANCEL, and OUTPUT commands.

On completion, you have the option of either continuing normally or having the job fail.
Installation Exit 29

Environment
This installation exit routine runs in the JES3 global address space.

This exit is called in 31-bit addressing mode. The entry point in register 15 is a 31-bit address.

Linkage Conventions
This exit routine is invoked with the ACALL macro using ASAVE linkage. To return from the exit, use the ARETURN macro without the RC= parameter.

Information Supplied on Entry
The contents of the registers on entry are:
- R0-10: Irrelevant
- R11: IATYFCT address
- R12: IATYTVT address
- R13: Pointer to the input service data CSECT (IATISDT) - mapped by macro IATYISD
- R14: Return address
- R15: Entry point address for IATUX29

Programming Considerations
Register 13 points to data CSECT IATISDT, which contains:
- JDABADDR – Address of the FDB for the JDAB
- JMRADDR – Address of the FDB for the JMR
- JCTBUILD – Address of the JCT build area
- ISFLAG11 – Contains X'04' (ISTSOSUB) if the origin of the job is a TSO submit.

The accounting information area is 144 bytes in length and must be specified in the following format:
- The first byte of the area contains the number (in binary) of accounting fields.
- The remaining bytes contain accounting fields. Each accounting field entry begins with a 1-byte length field (in binary) followed by the field.

Restrictions
You can use this exit to change any value in CIPARM (JDABPMID) except:
- The account number and programmer's name
- The job MSGLEVEL default
- The MSGCLASS

Refresh Status
This user exit is refreshable through the command
*MODIFY,X,M=IATxxxx,REFRESH.

Information Supplied on Exit
The contents of the registers on exit should be:
- R0-1: Irrelevant
- R2-13: Same as on entry
- R14: Return address
- R15: Return codes:
  - 0: Accept the job
  - 4: Flush the job
CSECT Name: IATUX30 (Examine Authority Level for TSO/E Terminal Commands)

This exit allows an installation that does not use the standard TSO/E jobname (a userid plus a one-character suffix) to authorize the use of STATUS, CANCEL, and OUTPUT commands. An installation using a client/server processing application (such as TCP/IP) can authorize the use of STATUS and CANCEL commands issued from the client.

This exit is entered each time a STATUS, CANCEL or OUTPUT request is selected for processing (from module IATGRWQ for a STATUS or CANCEL request, or module IATOSPD for an OUTPUT request). It allows you to authorize the use of these TSO/E terminal commands if authority checking is not being performed by TSO/E installation exit IKJEFF53. The exit may also be entered each time a job is selected to be used in response to the request.

On completion, you have the option of specifying that the request either be accepted or rejected.

Note: The installation decides who is to control the job and the output of the job with the STATUS, CANCEL, and OUTPUT commands. This decision can be made in the installation exit, IATUX29, by setting the JCTSBOWN bit of flag, JCTFL7, in the job control table (IATYJCT) that is associated with the job when field JCTTUSID (the job submitter) is to be used to authorize the command or JCTOUSID (the job owner userid) is to be used to authorize the command. See CSECT Name: IATUX29 (Examine the Accounting Information)” on page 96 for more information. Job access will be either by job submittor, job owner, or accessor environment element (ACEE) userid.

access by owner (JCTOUSID) if JCTFL7 (JCTSBOWN) = 0 (default)

access by submittor (JCTTUSID) if JCTFL7 (JCTSBOWN) = 1

Environment

This installation exit routine runs in the JES3 global address space.

This exit is called in 31-bit addressing mode. The entry point in register 15 is a 31-bit address.

Linkage Conventions

This exit routine is invoked with the ACALL macro using ASAVE linkage. To return from the exit, use the ARETURN macro without the RC= parameter.

Information Supplied on Entry

The contents of the registers on entry are:

R0 Irrelevant
R1 Pointer to input parameter area (IATYUX30)
R2-R10 Irrelevant
R11 IATYFCT address
R12 IATYTVT address
R13 Irrelevant
Installation Exit 30

R14 Return address
R15 Entry point address for IATUX30

Programming Considerations

Register 1 points to a parameter list (defined in macro IATYUX30) which contains the following:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UX30AID</td>
<td>Address of 8-byte field containing the userid from the requester’s accessor environment element (ACEE), left-justified and padded with blanks, or zeros (used for client/server processing applications such as TCP/IP). If this ACEE is not the ACEE of the original requester associated with the job (as could be the case in certain client/server environments), the installation must design this exit to process accordingly.</td>
</tr>
<tr>
<td>UX30FUNC</td>
<td>Address of function code identifying type of request being processed:</td>
</tr>
<tr>
<td></td>
<td>X'80' STATUS</td>
</tr>
<tr>
<td></td>
<td>X'40' CANCEL</td>
</tr>
<tr>
<td></td>
<td>X'10' OUTPUT</td>
</tr>
<tr>
<td>UX30SSOB</td>
<td>Address of subsystem options block (SSOB) extension mapped by macro IEFSSLS for STATUS/CANCEL request or by macro IEFSSSO for output request.</td>
</tr>
<tr>
<td>UX30JOB</td>
<td>One of the following:</td>
</tr>
<tr>
<td></td>
<td>• Job control table (JCT) address for CANCEL job selection entry</td>
</tr>
<tr>
<td></td>
<td>• Job queue element (JQE) address for STATUS job selection entry</td>
</tr>
<tr>
<td></td>
<td>• RESQUEUE address for OUTPUT job selection entry</td>
</tr>
<tr>
<td></td>
<td>• X'0' for request selection entry.</td>
</tr>
<tr>
<td>UX30ID</td>
<td>Address of 8-byte field containing the requesting TSO/E terminal userid (left-justified and padded with blanks), or the application name (client/server processing).</td>
</tr>
<tr>
<td>UX30FLG1</td>
<td>Flags defined in UX30FLG1 as follows:</td>
</tr>
<tr>
<td>Flag</td>
<td>Definition</td>
</tr>
<tr>
<td>UX30END</td>
<td>Indicates the end of the parameter list</td>
</tr>
<tr>
<td>UX30VAL</td>
<td>The valid TSO/E userid flag indicating that the address at offset X'18' (UX30ID) points to a valid TSO/E userid or application name.</td>
</tr>
<tr>
<td>UX30CVAL</td>
<td>The valid ACEE userid flag indicating that the address at offset X'8' (UX30AID) points to a valid client userid</td>
</tr>
</tbody>
</table>

Notes:
1. If UX30VAL is set, the UX30AID field is not used.
2. If UX30CVAL is set, the UX30ID field is not used.
3. The remaining bits in the byte are reserved.

UX30UFLG Flags defined in UX30UFLG as follows:

<table>
<thead>
<tr>
<th>Flag</th>
<th>Definition</th>
</tr>
</thead>
</table>

100 z/OS V1R11.0 JES3 Customization
Installation Exit 30

Bit UX308968 (X'80')  Issue only secondary message IAT8968
Bit UX308969 (X'40')  Issue only secondary message IAT8969
Bit UX30B896 (X'20')  Issues both messages IAT8968 and IAT8969
Bit UX30N896 (X'10')  Do not issue either message IAT8968 or IAT8969

Note: The remaining bits in field UX30UFLG are reserved. See Table 10 on page 58

Refresh Status

This user exit is refreshable through the command *MODIFY,X,M=IATxxxx,REFRESH.

Information Supplied on Exit

The contents of registers on exit should be:

R0  Dependent on the contents of register 15 as follows:
    • Not applicable if register 15 contains a return code of less than 16.
    • Must be one of the following return codes if register 15 contains a return code of 16 for request selection entry:
      UX30RN00  Use the userid in field UX30ID.
      UX30RN04  Use the userid in field SSCSJ0BN.
      UX30RN08  Use the userid in field UX30AID.

R1-13  Same as on entry
R14  Return address
R15  The contents of register 15 must be one of the following:

For request selection entry:

UX30JFOK  TSO/E installation exit IKJEFF53 is in use for authority checking; this request should be processed with no additional checking other than that implied in the STATUS/CANCEL/OUTPUT command descriptions.
UX30J3OK  IATUX30 is in use for authority checking; processing of this request should continue, and no additional authority is required.
UX30J3RT  IATUX30 is in use for authority checking; a subsequent entry is required each time a job is selected for this request.
UX30J3RJ  IATUX30 is in use for authority checking; this request should be rejected.
UX30J3JO  IATUX30 is in use for authority checking; use the JOB owner for all processing.

For job selection entry:

UX30JBOK  Process selected job.
UX30JBRJ  Reject selected job, but continue job scan.

Chapter 1. JES3 DSPs and Installation Exits 101
Installation Exit 30

UX30RQRJ  End this request.

CSECT Name: IATUX32 (Override the DYNALDSN Initialization Statement)

Installation exit IATUX32 receives control from the subsystem intercommunication dynamic allocation routine (module IATSICA) and allows you to override the DYNALDSN initialization statement for any dynamic allocation.

This exit also allows you to specify the actions to be taken for an APPC transaction program dynamic allocation. For example, you can specify whether integrity checking should be performed.

Environment

This installation exit routine runs only in the user’s address space.

This exit is called in 31-bit addressing mode. The entry point in register 15 is a 31-bit address.

This module must reside in the LPA.

Linkage Conventions

This exit does not use ASAVE linkage. Registers 4 through 14 must be maintained across the call to IATUX32. You can use AWAMACSV in the AWA (IATYAWA) to save registers.

Information Supplied on Entry

The contents of the registers on entry are:

- **R0-3** Irrelevant
- **R4** Address of the DYD entry (mapped by macro IATYDYD)
- **R5** Address of the JFCB containing the data set name and the volume serial number (mapped by macro IEFJFCBN)
- **R6** Irrelevant
- **R7** Address of the SSVT
- **R8** Address of the SSOB extension - mapped by macro IEFSSDY
- **R9** Address of MEMENTRY (mapped by macro IATYMEM)
- **R10-12** Irrelevant
- **R13** Address of the allocation work area (mapped by macro IATYAWA); AWARSAV1 contains the address of the current SIOT (mapped by macro IEFASIOT)
- **R14** Return address
- **R15** Entry point address for IATUX32

Programming Considerations

The following flags indicate that the dynamic allocation call is on behalf of a transaction program:

- MEMTPACT (in field MEMTFLG of IATYMEM) indicates that a transaction program is currently active
- AWABALLC (in field AWAFLAG1 of IATYAWA) indicates that this is a batch allocation call made on behalf of data sets defined in a TP profile.
Refresh Status

This user exit is not refreshable through the command *MODIFY,X,M=IATxxxx,REFRESH.

Information Supplied on Exit

The contents of the registers on exit should be:
- **R0-3**: Irrelevant
- **R4-13**: Same as on entry
- **R14**: Return address
- **R15**: Return codes:
  - **0**: Perform action specified by the dynamic allocation data set names table (DYD) entry indicator.
  - **4**: Bypass main device scheduler (MDS) data set integrity protection.
  - **8**: Protect (normal MDS processing)
  - **12**: Reject request. Do not allow dynamic allocation. Send rejection code.

CSECT Name: IATUX33 (Modify JCL EXEC Statement and JES3 Control Statement)

This installation exit receives control from the input statement scanner (module IATISLG) whenever JES3 input service encounters a record of a JCL EXEC statement, a JCL comment statement (except when it is placed between continuations of a JCL statement other than an EXEC statement), or a JES3 control statement (excluding the /*DATASET and /*ENDDATASET statements). It also receives control if JES3 input service encounters a JES2 control statement (that is, a */ followed by a non-blank character). This exit allows you to verify, modify, or add control statement parameters in the input stream.

Environment

This installation exit routine runs in the JES3 global address space.

This exit is called in 31-bit addressing mode. The entry point in register 15 is a 31-bit address.

Linkage Conventions

This exit routine is invoked with the ACALL macro using ASAVE linkage. To return from the exit, use the ARETURN macro, specifying RC=0.

Information Supplied on Entry

The contents of the registers on entry are:
- **R0**: Irrelevant
- **R1**: Address of the record to be processed
- **R2-10**: Irrelevant
- **R11**: IATYFCT address
- **R12**: IATYTVT address
- **R13**: Address of the input service data CSECT (IATISDT) - mapped by macro IATYISD
- **R14**: Return address
- **R15**: Entry point address for IATUX33
Programming Considerations

On entry, register 1 contains the address of the 80-byte record to be processed. However, if the previous exit from this routine specified a return code 8, then register 1 still points to the same record previously returned.

On return, register 15 must contain the current record status return code. If an incorrect return code is issued, the job is canceled, message IAT6139 is issued, and the installation exit is disabled until the next JES3 restart.

Standard JES3 control statement and JCL statement rules must be followed to ensure that a valid input stream is created.

Refresh Status

This user exit is refreshable through the command

*MODIFY,X,M=IATxxxx,REFRESH.

Information Supplied on Exit

The contents of the registers on exit should be:

<table>
<thead>
<tr>
<th>Register</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R0</td>
<td>Irrelevant</td>
</tr>
<tr>
<td>R1</td>
<td>The address of the record to be processed. (If a return code of 4 is issued, this address is ignored.)</td>
</tr>
<tr>
<td>R2-13</td>
<td>Same as on entry</td>
</tr>
<tr>
<td>R14</td>
<td>Return address</td>
</tr>
<tr>
<td>R15</td>
<td>Return code:</td>
</tr>
<tr>
<td>0</td>
<td>Process the record pointed to by the address in register 1 and get a new record.</td>
</tr>
<tr>
<td>4</td>
<td>Ignore the current record and get a new record to process.</td>
</tr>
<tr>
<td>8</td>
<td>Process the record pointed to by the address in register 1 and return control to the installation exit without getting a new record.</td>
</tr>
<tr>
<td>12</td>
<td>Dummy installation exit; the exit will not be called again until the system is restarted.</td>
</tr>
</tbody>
</table>

CSECT Name: IATUX34 (Modify JCL DD Statement)

This installation exit receives control from the input statement scanner (module IATISLG) whenever JES3 input service encounters a record of a JCL DD statement (excluding the DD * and DD DATA statements), or a JCL comment statement placed between continuations of such a statement. This exit enables you to verify, modify, or add JCL control statement parameters in the input stream.

Even though this exit can be used to add storage management system (SMS) related keywords, such as STORCLAS, DATACLAS, and MGMTCLAS, to the JCL, IBM suggests that, instead, you use the automatic class selection (ACS) exits called by SMS during the MVS interpretation stage.

Environment

This installation exit routine runs in the JES3 global address space.

This exit is called in 31-bit addressing mode. The entry point in register 15 is a 31-bit address.
Linkage Conventions

This exit routine is invoked with the ACALL macro using ASAVE linkage. To return from the exit, use the ARETURN macro without the RC= parameter.

Information Supplied on Entry

The contents of the registers on entry are:

- **R0**: Irrelevant
- **R1**: Pointer to the current record to be processed
- **R2-10**: Irrelevant
- **R11**: IATYFCT address
- **R12**: IATYTVT address
- **R13**: Address of the input service data CSECT (IATISDT) - mapped by macro IATYISD
- **R14**: Return address
- **R15**: Entry point address for IATUX34

Programming Considerations

On entry, register 1 contains the address of the 80-byte record to be processed. However, if the previous exit from this routine specified a return code 8, then register 1 still points to the same record previously returned.

On return, register 15 must contain the current record status return code. If an incorrect return code is issued, the job is canceled, message IAT6139 is issued, and the installation exit is disabled until the next JES3 restart.

Standard JCL statement syntax *must* be followed to ensure that the final result is a valid input stream.

Refresh Status

This user exit is refreshable through the command

*MODIFY,X,M=IATxxxx,REFRESH.

Information Supplied on Exit

The contents of the registers on exit should be:

- **R0**: Irrelevant
- **R1**: The address of the record to be processed. (If a return code of 4 is issued, this address is ignored.)
- **R2-10**: Irrelevant
- **R11-13**: Same as on entry
- **R14**: Return address
- **R15**: Return code:

  - 0: Process the record pointed to by the address in register 1 and get a new record.
  - 4: Ignore the current record and get a new record to process.
  - 8: Process the record pointed to by the address in register 1 and return control to the installation exit without getting a new record.
  - 12: Dummy installation exit; the exit will not be called again until the system is restarted.
CSECT Name: IATUX35 (Validity Check Network Commands)

This exit, from the JES3 networking console processor (module IATCNNJ), is entered before normal validation is performed for incoming network commands. It gives you the opportunity to assign special authority to commands received from specific nodes or userids.

Notes on Usage

If installation exit IATUX35 indicates that the authority decision is to be made by JES3 (return codes 4 or 8), you must also code either installation exit IATUX58 or IATUX59 to prevent JES3 from using the security decision from RACF. See “CSECT Name: IATUX58 (Modify Security Information Before JES3 Security Processing)” or “CSECT IATUX59 (Modify Security Information After JES3 Security Processing)” for more information.

Environment

This installation exit routine runs in the JES3 global address space.

This exit is called in 31-bit addressing mode. The entry point in register 15 is a 31-bit address.

Linkage Conventions

This exit routine is invoked with the ACALL macro using ASAVE linkage. To return from the exit, use the ARETURN macro without the RC= parameter.

Information Supplied on Entry

The contents of the registers on entry are:

- R0-2: Irrelevant
- R3: Pointer to NJE command/message transmission queue entry (IATYNBF)
- R4: Pointer to NJE command/message record (IATYNCN)
- R5: CMNDBFR+1. The command buffer (CMNDBFR) is a 132-byte area, located within module IATCNNJ, that contains the operator command text or message text and the text byte count (register 5 minus 1) before modification. JES3 uses this buffer to send the operator command or message to JES3 using INTERCOM. If you modify the text, you must adjust the count to reflect the change.
- R6: Irrelevant
- R7: Address of the NJE console queue - mapped by macro IATYNCO
- R8-10: Irrelevant
- R11: IATYFCT address
- R12: IATYTVT address
- R13: Irrelevant
- R14: Return address
- R15: Entry point address for IATUX35

Programming Considerations

For the INQUIRY command, commands submitted at a particular node will display only those jobs submitted at the same node. For the MODIFY command, commands submitted at a particular node and by a particular user will affect only those jobs submitted at the same node and by the same user.
Note: NJE consoles are designated as JES3-only consoles and only allow JES3 commands. Therefore, only JES3 commands will be processed on the receiving node.

Refresh Status

This user exit is refreshable through the command
*MODIFY,X,M=IATxxxx,REFRESH.

Information Supplied on Exit

The contents of the registers on exit should be:

- R0-2: Irrelevant
- R3-12: Same as entry
- R13: Irrelevant
- R14: Return address
- R15: Return code:
  - 0: Command is to be processed.
  - 4: The authority decision is to be made by IATCNNJ or IATCNIA using the IBM default for the command only if you code installation exit IATUX58 or IATUX59 to ignore the RACF security decision. If you do not code installation exit IATUX58 or IATUX59, the command authority check is done by RACF.
  - 8: The authority decision is to be made by IATCNNJ or IATCNIA using the IBM default for the command only if you code installation exit IATUX58 or IATUX59 to ignore the RACF security decision. If you do not code installation exit IATUX58 or IATUX59, the command authority check is done by RACF. The exit is made a dummy exit and is not called again.
  - 12: Command is to be rejected; no message is sent.
  - 16: Command is to be rejected; message IAT7143 will be issued.

CSECT Name: IATUX36 (Collect Accounting Information)

This exit is called by the JES3 networking job and SYSOUT receive processor (module IATNTJS). This exit allows the local installation to obtain information from the JES3 networking job header and the job control table (IATYJCT) to fill fields that are reserved for local accounting information in the:

- Job management record (IATYJMR)
- Job data accounting block (IATYJDA)

In addition to the skeleton JMR, the skeleton job control table (JCT) and job description and accounting block (JDAB) have been created. They can be accessed, if needed, through fields in the IATYNRD macro. Separate fields exist for job and SYSOUT streams.

Environment

This installation exit routine runs in the JES3 global address space.
This exit is called in 31-bit addressing mode. The entry point in register 15 is a 31-bit address.

Linkage Conventions

This exit routine is invoked with the ACALL macro using ASAVE linkage. To return from the exit, use the ARETURN macro without the RC= parameter.

Information Supplied on Entry

The contents of the registers on entry are:
- **R0-1**: Irrelevant
- **R2**: Total JMR length
- **R3**: Pointer to job header (NJHSTART) IATYNJH
- **R4**: Pointer to job management record (JMR) IATYJMR
- **R5-10**: Irrelevant
- **R11**: IATYFCT address
- **R12**: IATYTVT address
- **R13**: Pointer to NJE receiver data area (IATYNRD)
- **R14**: Return address
- **R15**: Entry point address for IATUX36

Programming Considerations

You can obtain access to the job control table (JCT) or the job data accounting table (JDAB) by:

1. Determining if the exit was called while JES3 was processing an NJE job stream or an NJE SYSOUT stream. If NRDJBRCV is on, JES2 was processing an NJE job stream. If NRDOPRCV is on, JES3 was processing an NJE SYSOUT stream.
2. Use Table 12 to determine from which fields you should obtain the addresses of the JCT and JDAB.

### Table 12. Obtaining Access to JCT or JDAB

<table>
<thead>
<tr>
<th></th>
<th>JCT</th>
<th>JDAB</th>
</tr>
</thead>
<tbody>
<tr>
<td>NJE JOB Stream</td>
<td>NRDJBJCT</td>
<td>NRDJBJDA</td>
</tr>
<tr>
<td>NJE SYSOUT Stream</td>
<td>NRDOPJCT</td>
<td>NRDOPJDA</td>
</tr>
</tbody>
</table>

Refresh Status

This user exit is refreshable through the command

```
*MODIFY,X,M=IATxxxx,REFRESH.
```

Information Supplied on Exit

The contents of the registers on exit should be:

- **R0-10**: Irrelevant
- **R11-13**: Same as on entry
- **R14**: Return address
- **R15**: Return code:
  - **0**: Local processing complete.
  - **4**: Dummy exit; exit will not be called again until the local system is restarted.
CSECT Name: IATUX37 (Modify the JES3 Networking Data Set Header for Local Execution)

This exit, from the JES3 networking job and SYSOUT receive processor (module IATNTJS for BSC and SNA, module IATDMJAM for TCP/IP), allows for special processing of SYSOUT classes for local SYSOUT data sets.

This exit is also used to modify data set headers received from another node.

Environment

This installation exit routine runs in the JES3 global address space.

This exit is called in 31-bit addressing mode. The entry point in register 15 is a 31-bit address.

Linkage Conventions

This exit routine is invoked with the ACALL macro using ASAVE linkage. To return from the exit, use the ARETURN macro without the RC= parameter.

Information Supplied on Entry

The contents of the registers on entry are:
- R0-2: Irrelevant
- R3: Pointer to data set header (NDHSTART) macro IATYNDH
- R4-10: Irrelevant
- R11: IATYFCT address
- R12: IATYTVT address
- R13: Pointer to NJE receiver data area (IATYNRD)
- R14: Return address
- R15: Entry point address for IATUX37

Programming Considerations

None.

Refresh Status

This user exit is refreshable through the command

*MODIFY,X,M=IATxxxx,REFRESH.

Information Supplied on Exit

The contents of the registers on exit should be:
- R0-2: Irrelevant
- R3-13: Same as on entry
- R14: Return address
- R15: Return code:

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Local processing complete.</td>
</tr>
<tr>
<td>4</td>
<td>Dummy exit; exit will not be called again until the system is restarted.</td>
</tr>
</tbody>
</table>
CSECT Name: IATUX38 (Change the SYSOUT Class and Destination for Networking Data Sets)

This installation exit, from the JES3 networking store-and-forward routine (module IATNTSF), allows you to change the SYSOUT class and destination of a SYSOUT data set that is to be processed locally. Module IATNTSF provides pointers to the job data set (JDS) entry and to the networking data set header. This exit can examine the JDS entry and make any modifications necessary for SYSOUT class and destination processing. This exit should NOT modify any JDS fields not related to SYSOUT class and destination processing as documented in this exit.

Environment

This installation exit routine runs in the JES3 global address space.

This exit is called in 31-bit addressing mode. The entry point in register 15 is a 31-bit address.

Linkage Conventions

This exit routine is invoked with the ACALL macro using ASAVE linkage. To return from the exit, use the ARETURN macro without the RC= parameter.

Information Supplied on Entry

The contents of the registers on entry are:

- R0-R1: Irrelevant
- R2: Size of JDS entry
- R3: Address of JDS entry (macro IATYJDS)
- R4: Pointer to the variable portion of the JDS entry.
- R5-6: Irrelevant
- R7: Pointer to data set header (NDHSTART) IATYNDH
- R8-10: Irrelevant
- R11: IATYFCT address
- R12: IATYTVT address
- R13: Pointer to store-and-forward data area (IATYNFD)
- R14: Return address
- R15: Entry point address for IATUX38

Programming Considerations

Any modification made to the JDS will be used, even if a return code of 4 is given to dummy exit.

Do NOT attempt to modify or add information to the variable portion of the JDS entry (beginning with field JDSTYPE), as this would cause unpredictable results when JES3 tried to use the entry.

The following fields are the only ones intended to be changed or set up by this exit. Because JES3’s SYSOUT class and destination processing is bypassed, it is the discretion of the installation to determine the proper setting of the documented fields. For example, if the installation determines that the networking output does not need a destination change, then the destination field does not have to be made valid; accordingly, JDSDEST and JDSDEST may be left unset. As another example, the installation must set JDSTYPE in order for output service to process this networking output; accordingly, JDSTYPE must be set to PRINT, PUNCH, HOLD, and so on.
**JDSCLASS**  
**SYSOUT class**

**JDSCTYPE**  
**SYSOUT type**

Description: Type of processing to be performed (for example: PRINT, PUNCH, HOLD, and so on.)

**JDSDEST**  
**Data set destination**

**JDSFLG3**  
**JDSDDEST Flag**

Description: Indicates that JDSDEST has been set.

Affect: If JDSDDEST is set then JDSDEST will be used to route the networking output. If JDSDDEST is NOT set the JDSDEST will not be used to route the networking output.

When to set: Set JDSDDEST whenever JDSDEST has been set.

**JDSFLG9**  
**JDSSYSD flag**

Description: Indicates that a secondary destination was NOT specified (for example, NDHGRMT is either all zeros or all blanks).

Affect: If JDSSYSD is set and no other destination is specified then the networking output will be routed to ANYLOCAL. If JDSSYSD is NOT set and no other destination is specified then the networking output will be routed to the origin (TSO userid if submitted from TSO). If a destination is specified then JDSSYSD has no affect.

When to set: Set JDSYSD when NO secondary was supplied in the data set header (for example, NDHGRMT is either all zeros or all blanks).

**JDSTXDS flag**

Description: Indicates that the networking output is TSO transmit output. This bit is set to indicate that the networking output primary destination is this node (for example, output will be handled on this node).

Affect: If JDSTXDS is set then the value in JDSDEST is used as a local destination without performing NJE node table search against this value. This is to support receiving TSO transmitted output targeted to a TSO userid equal to a networking node name. If JDSTXDS is NOT set, then the value in JDSDEST will search for the local destination in the NJE node table. If found, the networking output will be sent to the NJE node defined by JDSDEST.

When to set: Set JDSTXDS for TSO transmit output. TSO transmit output is identified by:
- the HOLD for TSO bit (NDHGF1HD is set), or
- the secondary destination is equal to the external writer name (for example, NDHGRMT is equal to NDHGXMTR).

### Refresh Status

This user exit is refreshable through the command

*MODIFY,X,M=IATxxxx,REFRESH.*

### Information Supplied on Exit

The contents of the registers on exit should be:

- **R0-1** Irrelevant
Installation Exit 38

R2  Size of the JDS entry (may not be modified)
R3-10  Irrelevant
R11-13  Same as on entry
R14  Return address
R15  Return code:

0  Modified characteristics of inbound SYSOUT.
4  Exit will not be called again until the system is restarted.

Note: The first time the exit is called, any changes made to the JDS will be used, even if the exit returns a code of 4 in register 15.

If a return is made with a return code other than 0 or 4, then module IATNTSF is routed through normal JES3 SYSOUT class and destination processing before continuing.

Setting Flags

Example — Setting Flags, JDSTXDS and JDSSYSD
If you want to set both flags JDSTXDS and JDSSYSD

| TM   | NDHGFLG1,NDHGFLHD | DSH indicates hold for TSO? |
| BC   | ALLON,UX380101    | BIY, go set TSO XMIT bit    |
| CLC  | NDHGRMT,TVTZERO   | Remote specified?           |
| BC   | EQ,UX38010T       | BIN, skip setting of no remote bit |
| CLC  | NDHGRMT,TVTBLANK  | Remote specified?           |
| BC   | NE,UX38010T       | BIY, skip setting of no remote bit |
| UX38010G | OH         |                               |
| UX38010T | DS,OH   | UX38010G,JD,JDSTXDS,JDSSYSD |
| UX380102 | DS,OH   | UX380102,Continue            |
| UX38010T | DS,OH   | UX38010T,NDHGRT,NDHGRTXTR   |
| UX380101 | DS,OH   | UX380101,NDHGRTXTR,NDHGRTXTR |
| UX380102 | DS,OH   | UX380102,Continue            |

CSECT Name: IATUX39 (Modify the Data Set Header for a SYSOUT Data Set)

This installation exit allows you to modify the data set header (mapped by IATYNDH) of a SYSOUT data set (in a network stream) before JES3 or MVS/BDT transmits the network stream containing the data set.

This installation exit can be used to process BSC/NJE jobs and SNA/NJE jobs. The call indicator supplied as input to the exit indicates whether BSC networking or SNA networking is in effect.

Environment

This installation exit routine runs in the JES3 global address space.

This exit is called in 31-bit addressing mode. The entry point in register 15 is a 31-bit address.
Linkage Conventions
This installation exit is called from module IATNTDH using ASAVE linkage with the ACALL and ARETURN macros. To return from the exit, use the ARETURN macro without the RC= parameter.

Information Supplied on Entry
The contents of the registers on entry are:

- **R0** A call indicator as follows:
  - If byte zero is X'00', then BSC mode is active. BSC mode refers to a data set header that belongs to data passing through BSC networking.
  - If byte zero is X'80', then SNA mode is active. SNA mode refers to a data set header that belongs to data passing through SNA networking. The remaining three bytes of register 0 are irrelevant.

- **R1-5** Irrelevant
- **R6** Pointer to OSE data set section
- **R7** Pointer to NJE data set header work area - mapped by IATYNDH (NDHSTART)
- **R8** Pointer to JES3 networking writer data CSECT - mapped by macro IATYNOD (module IATDNJ) if in BSC mode. If in SNA mode, register 8 contains zero.
- **R9** Pointer to OSE variable section
- **R10** Irrelevant
- **R11** Pointer to the IATYFCT
- **R12** Pointer to the IATYTFT
- **R13** Pointer to the IATYWTR data area if in BSC mode. Pointer to the IATYOSA data area if in SNA mode.
- **R14** Return address
- **R15** Entry point address for IATUX39

Programming Considerations:

**For Binary Synchronous Communication (BSC) Invocations**

Upon entry, the RQOSESUP for the RQ in WTRDRSQ is off (set to zero). Upon exit, RQOSESUP should be in the off state also.

The OSE is passed to IATUX39 is a copy of the disk OSE. You can access this information without using the IATXOSUP macro. Changes made to this OSE are not made to the job's disk OSE chain or in any storage representation of the disk OSE.

If you code the IATXOSUP macro in this installation exit, follow the step-by-step process in the description for the IATXOSUP macro.

**For Systems Network Architecture (SNA) Invocations:**

1. Do not perform I/O for the spool OSE chain for this job.
2. Do not use the IATXOSUP macro in this installation exit.

Upon entry, the RQOSESUP for the RQ in OSARQA is on (set to one). Do not reset RQOSESUP for this RQ at any time.

The OSE passed to IATUX39 can be accessed without additional serialization. Do not modify this OSE.
Installation Exit 39

Refresh Status

This user exit is refreshable through the command
*MODIFY,X,M=IATxxxx,REFRESH.

Information Supplied on Exit

The contents of the registers on exit should be:

<table>
<thead>
<tr>
<th>Register</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R0-10</td>
<td>Irrelevant</td>
</tr>
<tr>
<td>R11-13</td>
<td>Same as on entry</td>
</tr>
<tr>
<td>R14</td>
<td>Return address</td>
</tr>
<tr>
<td>R15</td>
<td>Return code:</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>4</td>
</tr>
</tbody>
</table>

CSECT Name: IATUX40 (Modify Job Header for a Network Stream Containing a Job)

This installation exit allows you to modify a job header (mapped by IATYNJH) for a network stream containing a job (JCL and SYSIN) before JES3 or MVS/BDT transmits the network stream. The header will be transmitted with the job to the execution node.

This installation exit can be used to process BSC/NJE jobs and SNA/NJE jobs. The call indicator supplied as input (in register 0) indicates whether the exit is being called by input service (IATISNJ) or by output service (IATNTHT).

Environment

This installation exit routine runs in the JES3 global address space.

This exit is called in 31-bit addressing mode. The entry point in register 15 is a 31-bit address.

Linkage Conventions

This exit is called from either module IATISNJ or module IATNTHT using ASAVE linkage with the ACALL and ARETURN macros. To return from the exit, use the ARETURN macro without the RC= parameter.

Programming Considerations

None.

Information Supplied on Entry

The contents of the registers on entry are:

<table>
<thead>
<tr>
<th>Register</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R0</td>
<td>A call indicator as follows:</td>
</tr>
<tr>
<td></td>
<td>– If byte zero is X'00', the caller is module IATISNJ.</td>
</tr>
<tr>
<td></td>
<td>– If byte zero is X'80', the caller is module IATNTHT.</td>
</tr>
<tr>
<td>R1-7</td>
<td>Irrelevant</td>
</tr>
<tr>
<td>R8</td>
<td>Pointer to the job header that you want to modify. The job header is mapped by IATYNJH.</td>
</tr>
<tr>
<td>R9-10</td>
<td>Irrelevant</td>
</tr>
<tr>
<td>R11</td>
<td>Pointer to the IATYFCT</td>
</tr>
<tr>
<td>R12</td>
<td>Pointer to the IATYTVT</td>
</tr>
</tbody>
</table>
R13 If the caller is IATISNJ, register 13 contains a pointer to the input service data area mapped by IATYISD. If the caller is IATNTHT, register 13 contains zero.

R14 Return address

R15 Entry point address for IATUX40

Refresh Status

This user exit is refreshable through the command
*MODIFY,X,M=IATxxxx,REFRESH.

Information Supplied on Exit

The contents of the registers on exit should be:

<table>
<thead>
<tr>
<th>Register</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R0-1</td>
<td>Irrelevant</td>
</tr>
<tr>
<td>R2-13</td>
<td>Same as on entry</td>
</tr>
<tr>
<td>R14</td>
<td>Return address</td>
</tr>
<tr>
<td>R15</td>
<td>Return code:</td>
</tr>
<tr>
<td>0</td>
<td>Job header was modified.</td>
</tr>
<tr>
<td>4</td>
<td>Dummy exit; exit will not be called again until the JES3 system is restarted.</td>
</tr>
<tr>
<td>8</td>
<td>The following occurs only in IATISNJ. Job header was not modified. This return code ensures that SNA/NJE job headers are not respooled (by IATOSBP) when no updates are made.</td>
</tr>
</tbody>
</table>

CSECT Name: IATUX41 (Determine the Disposition of a Job that Exceeds the Job JCL Limit)

This installation exit, from converter/interpreter subtask processing (module IATIIST), allows you to cancel a job that exceeds the job JCL limit or to override the job JCL limit. Module IATIIST calls this exit routine only when a job exceeds the job JCL statement limit.

The limit of JCL statements allowed for the job is determined by the value specified in the MAXJOBST parameter on the STANDARDS initialization statement. This installation exit can be used to set the IDDSWA bit in the IDD data area which will allow the job's SWA blocks to be moved into virtual storage above 16 megabytes. With SWA blocks moved above 16 megabytes, the job may be able to continue.

Environment

This installation exit routine runs under the C/I subtask. It may be called while the C/I subtask is operating in either the JES3 global address space or a C/I FSS address space.

This exit is called in 31-bit addressing mode.

Linkage Conventions

This exit is invoked with a BALR instruction. You must save and restore the registers. To return to the calling module, use either a BR 14 instruction or a RETURN macro. Do not use the ARETURN macro.
Installation Exit 41

Information Supplied on Entry

The contents of the registers on entry are:

- R0-6: Irrelevant
- R7: Pointer to the CI DSP data CSECT mapped by macro IATYIDD
- R8: Irrelevant
- R9: Pointer to the interpreter control table (ICT), which is mapped by IATYICT
- R10-11: Irrelevant
- R12: IATYTVT address
- R13: Save area address
- R14: Return address
- R15: Entry point address for IATUX41

Programming Considerations

Register 7 contains the address of the CI DSP data CSECT, which has the job’s JCL statement count, in field IDDJOBCT.

This exit is entered enabled and in key 1. The code must be reentrant. No JES3 services (such as AWAIT, JESREAD, and so on.) may be used because this installation exit routine runs under a JES3 subtask. The exit routine has read access to JES3 control blocks (such as the TVT) and job control blocks that are in main storage (such as the JMR).

Refresh Status

This user exit is refreshable through the command

*MODIFY,X,M=IATxxxx,REFRESH if the exit is running in the global address space.

Information Supplied on Exit

The contents of the registers on exit should be:

- R0-12: Same as on entry
- R13: Save area address
- R14: Return address
- R15: Return code:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Cancel the job.</td>
</tr>
<tr>
<td>4</td>
<td>Ignore the job limit for this job.</td>
</tr>
<tr>
<td>16</td>
<td>Disable CALL to installation exit; exit will not be called again until the system is restarted.</td>
</tr>
<tr>
<td>Other</td>
<td>Cancel the job.</td>
</tr>
</tbody>
</table>

CSECT Name: IATUX42 (TSO Interactive Data Transmission Facility Screening and Notification)

This exit, from the JES3 networking store-and-forward routine (module IATNTSF), allows you to accept or reject incoming files as they arrive at the receiver’s network node. If the installation exit routine accepts the file, the routine can request that JES3 notify the receiver that a transmitted file has arrived. If the installation exit routine rejects the file, JES3 notifies the sender that the file has been rejected and gives the reason why. The installation exit routine may provide the reason portion of the message. The default return from this exit routine accepts all files, and no notification messages are generated.
This exit is called for all incoming files which appear to be in NETDATA format. By default, if the destination is known to JES3, JES3 will place the incoming file on the writer queue for processing by a JES3 or FSS writer. If the destination is not known to JES3, it is treated as a TSO USERID and the incoming file is put on the hold queue for processing by TSO RECEIVE. IATUX42 will be called in both cases. The YUX42J3W bit can be used to determine whether JES3's default action would be to have the incoming file processed by a writer or by TSO RECEIVE.

**Note:** The DESTDEF initialization statement can be used to define destinations for inbound SYSOUT files to JES3.

**Environment**

This installation exit routine runs in the JES3 global address space.

This exit is called in 31-bit addressing mode. The entry point in register 15 is a 31-bit address.

**Linkage Conventions**

This exit routine is invoked with the ACALL macro using ASAVE linkage. To return from the exit, use the ARETURN macro without the RC= parameter.

**Information Supplied on Entry**

The contents of the registers on entry are:

- **R0**: Irrelevant
- **R1**: Pointer to the IATYUX42 parameter list (see below)
- **R2-10**: Irrelevant
- **R11**: IATYFCT address
- **R12**: IATYTVT address
- **R13**: Pointer to the networking store-and-forward data CSECT, IATNTFD
- **R14**: Return address
- **R15**: Entry point address for IATUX42

**Programming Considerations**

The parameter list passed to IATUX42 that is pointed to by register 1 is mapped by macro IATYUX42 and contains the following:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Offset in Hex</th>
<th>Length</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>YUX42OFA</td>
<td>+0</td>
<td>4</td>
<td>The address of output flag byte YUX42OF1. See the description of YUX42OF1 below for the valid bit settings.</td>
</tr>
<tr>
<td>YUX42JDS</td>
<td>+4</td>
<td>4</td>
<td>The address of the JDSENTRY block for the incoming file.</td>
</tr>
<tr>
<td>YUX42ORI</td>
<td>+8</td>
<td>4</td>
<td>The address of an 8-character field containing the originating node name.</td>
</tr>
<tr>
<td>YUX42USR</td>
<td>+C</td>
<td>4</td>
<td>The address of an 8-character field containing the originating userid or, if no userid was available, the job name.</td>
</tr>
<tr>
<td>YUX42SYS</td>
<td>+10</td>
<td>4</td>
<td>The address of an 8-character field containing the name of the processor on which JES3 will issue the TSO SEND command to the receiver. This name field will contain blanks (in EBCDIC) at entry to the installation exit routine if the receiver is not logged on. In that case, the installation exit routine may set the field for any TSO system in the JES3 complex. If the routine leaves the field containing blanks, JES3 will use the TSO SEND command to issue message IAT9154 on the global processor.</td>
</tr>
<tr>
<td>YUX42MSG</td>
<td>+14</td>
<td>4</td>
<td>The address of a 72-character field containing the reason portion of the message (IAT9155) to be sent to the originator if the file is rejected. If the installation exit routine does not modify this field, the reason portion of the message will read &quot;INVALID USERID.&quot; The first halfword of the field contains the length of the reason portion of the message. The high-order bit of the address is set to one to indicate the end of the parameter list.</td>
</tr>
</tbody>
</table>
Field Name | Offset in Hex | Length | Content
---|---|---|---
YUX42ID | +18 | 6 | Control block identifier “YUX42”.
YUX42VER | +1E | 2 | IATYUX42 version number.
YUX42OF1 | +20 | 1 | Output flag byte pointed to by YUX42OFA. The bit settings are defined as follows:

X'80' (YUX42DEL) —Delete the incoming file and send message IAT9155 to the originator.
X'40' (YUX42NOM) —Accept the incoming file and do not send a message to the receiver.
X'20' (YUX42WTQ) —Accept the incoming file and put it on the writer queue for processing by a JES3 or FSS writer.
X'10' (YUX42WTN) —Accept the incoming file and put it on the writer queue for processing by a JES3 or FSS writer. Change the destination to the value specified in YUX42ODS.
X'08' (YUX42HOQ) —Accept the incoming file and put it on the hold queue for processing by TSO RECEIVE.

YUX42IF1 | +21 | 1 | Input flag byte. These bits are set by JES3 before calling IATUX42 and should not be changed by the exit. The bit settings are defined as follows: X'80' (YUX42J3W) —JES3’s action, unless overridden by IATUX42, will be to put the incoming file on the writer queue. If this bit is off, then JES3 will put the incoming file on the hold queue for TSO RECEIVE.

YUX42RV1 | +22 | 2 | Reserved for IBM and must be zeros.
YUX42ODS | +24 | 4 | If YUX42WTN is set, the overriding destination.
YUX42RU1 | +2C | 14 | Available for installation use.
YUX42RV3 | +40 | 10 | Reserved for IBM and must be zeros.

This exit is not taken for files transmitted to the same node. To code user notification for files transmitted to the same node see TSO/E TRANSMIT exit INMXZ02.

Refresh Status

This user exit is refreshable through the command

*MODIFY,X,M=IATxxxx,REFRESH.

Information Supplied on Exit

The contents of the registers on exit should be:

R0-1 | Irrelevant
R2-13 | Same as entry
R14 | Return address
R15 | Return code:

0 | Local processing complete.
4 | Dummy exit; the exit will not be called again until the system is restarted.

CSECT Name: IATUX43 (Modify Job Header for a Network Stream Containing SYSOUT Data)

This installation exit allows you to modify a job header (mapped by IATYNJH) for a network stream containing SYSOUT data before JES3 or MVS/BDT transmits the network stream. The header will be transmitted with a job to the execution node.

The data typically associated with this exit is job data such as accounting and user information.
This installation exit can be used to process BSC/NJE jobs and SNA/NJE jobs. The call indicator supplied as input to the exit indicates whether BSC networking or SNA networking is in effect.

Environment

This installation exit routine runs in the JES3 global address space.

This exit is called in 31-bit addressing mode. The entry point in register 15 is a 31-bit address.

Linkage Conventions

This exit is called from module IATNTHT using ASAVE linkage with the ACALL and ARETURN macros without the RC= parameter.

Information Supplied on Entry

The contents of the registers on entry are:

- **R0**: Zero
- **R1**: Pointer to the NJE job header work area mapped by IATYNJH
- **R2**: A call indicator as follows:
  - If byte zero is X'00', then BSC mode is active. BSC mode refers to a job header that belongs to a job passing through BSC networking.
  - If byte zero is X'80', then SNA mode is active. SNA mode refers to a job header that belongs to a job passing through SNA networking.
  
  The remainder of register 2 is irrelevant.
- **R3-10**: Irrelevant
- **R11**: Pointer to the IATYFCT
- **R12**: Pointer to the IATYTVT
- **R13**: Pointer to the IATYNWR data area.
- **R14**: Return address
- **R15**: Entry point address for IATUX43

Programming Considerations

None.

Refresh Status

This user exit is refreshable through the command

*MODIFY,X,M=IATxxxx,REFRESH.

Information Supplied on Exit

The contents of the registers on exit should be:

- **R0**: Irrelevant
- **R2**: Same as on entry
- **R14**: Return address
- **R15**: Return code:

0: Job header was modified.
4: Dummy exit; the exit will not be called again until the JES3 system is restarted.
CSECT Name: IATUX44 (Modify JCL Statements)

This installation exit receives control from the input service logic control module (IATISLG) when JES3 input service encounters a record of a JCL statement other than a JOB, EXEC, or DD statement, or a JCL comment statement placed between continuations of such a statement. This exit allows you to verify, modify, or add JCL control statement parameters in the input stream. (You can change those statements by coding installation exit routines for IATUX28, IATUX33, or IATUX34, respectively.)

Environment

This installation exit routine runs in the JES3 global address space.

This exit is called in 31-bit addressing mode. The entry point in register 15 is a 31-bit address.

Linkage Conventions

This exit routine is invoked with the ACALL macro using ASAVE linkage. To return from the exit, use the ARETURN macro without the RC= parameter.

Information Supplied on Entry

The contents of the registers on entry are:

- R0        Irrelevant
- R1        Address of the current JCL statement
- R2-10     Irrelevant
- R11       IATYFCT address
- R12       IATYTVT address
- R13       Address of the input service data CSECT (IATISDT) - mapped by macro IATYISD
- R14       Return address
- R15       Entry point address for IATUX44

Programming Considerations

On entry, register 1 contains the address of the 80-byte record to be processed. However, if the previous exit from this routine specified a return code 8, then register 1 still points to the same record previously returned.

On return, register 15 must contain the current record status return code. If an incorrect return code is issued, the job is canceled, message IAT6139 is issued, and the installation exit is disabled until the next JES3 restart.

Refresh Status

This user exit is refreshable through the command

*MODIFY,X,M=IATxxxx,REFRESH.

Information Supplied on Exit

The contents of the registers on exit should be:

- R0        Irrelevant
- R1        Address of the current JCL statement
- R2-13     Same as on entry
- R14       Return address
- R15       Return code:
CSECT Name: IATUX45 (Change Job Information for Data Sets Processed by an Output Writer FSS)

This exit, from the functional subsystem (FSS) writer-get-data-set (GETDS) processor (module IATOSFG), allows you to change job information for data sets processed by output writer functional subsystems. The exit routine can change information that is to be written to the job header or data set header pages, or to the job trailer pages. IATUX45 can access NJE routing information such as programmer name and department. The exit routine can also change the data set characteristics, such as the copy count and character set. However, changes to any characteristics that are processed after JES3 output scheduling and device setup, such as forms characteristics, have no effect.

The exit routine makes these changes by modifying a service request list (SRL), located by field SRLSTART, sent from an output writer FSS that is requesting a data set. The output writer FSS sends the SRL, which is mapped by the IATYSRL macro, using the functional subsystem intercommunication (FSI). The section of the SRL containing information about the request to get a data set is the GETDS section. The primary control block is the job separator page area (JSPA).

You may expand the IAZJSPA area to add more information to be passed to the FSS address space. If you expand the IAZJSPA area, you must reassemble all JES3 modules that reference the IATYSRL macro.

Environment

This installation exit routine runs in the JES3 global address space.

This exit is called in 31-bit addressing mode. The entry point in register 15 is a 31-bit address.

Linkage Conventions

This exit routine is invoked with the ACALL macro using ASAVE linkage. To return from the exit, use the ARETURN macro with the RC= parameter.

Information Supplied on Entry

The contents of the registers on entry are:

- **R0**: Irrelevant
- **R1**: Pointer to IATYUX45 parameter list
- **R2-R10**: Irrelevant
- **R11**: IATYFCT address
- **R12**: IATYTVT address
- **R13**: Address of the writer data CSECT (IATODFD) - mapped by macro IATYWTR
- **R14**: Return address
Programming Considerations

Register 1 points to a parameter list, IATYUX45, that is mapped by the IATYUX45 mapping macro. The list contains the following parameters:

<table>
<thead>
<tr>
<th>Label</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UX45LVL</td>
<td>A two byte field containing the level number of the IATYUX45 control block. This field may be used to ease maintenance tasks by indicating to code in the installation exit the level of the IAYUX45 control block that is being processed. The current level number is defined in the equated symbol UX45LV01.</td>
</tr>
<tr>
<td>UX45LV01</td>
<td>An equated symbol indicating the current level number. IBM suggests that you maintain a history of the level numbers by adding, rather than replacing, additional values.</td>
</tr>
<tr>
<td>UX45JMRA</td>
<td>A read-only pointer to the address of the JMR control block. Use this field as input for the IATYJMR mapping macro to obtain the values of any data fields in the JMR. If this field contains zeros, the JMR is unavailable.</td>
</tr>
<tr>
<td>UX450SEV</td>
<td>The address of a read-only version of the OSE variable entry. Use this input field along with the OSESTART section of the IATYOSE mapping macro to obtain the values of data fields in the OSE variable entry. Note that UX450SEV points only to a copy of the fixed portion of the OSE variable entry and does not include any OSE data set entries. Use UX450SED to access the OSE data set entry.</td>
</tr>
<tr>
<td>UX450SED</td>
<td>The address of a read-only version of the OSE data set entry. Use this input field along with the OSEJDSPT section of the IATYOSE mapping macro to obtain the values of data fields in the OSE data set entry.</td>
</tr>
</tbody>
</table>

Register 13 contains the address of the writer data CSECT (IATODFD), which contains (in field WTRFSTAR) the address of the staging area containing the SRL to be modified.

IBM suggests that, if you use this exit to examine or modify data written on job header or trailer pages, you use the 8-character in field (JSPAJBID) job id instead of the 5-digit job number when referring to the job (for example, in messages to the operator).

Upon entry, the RQOSESUP for the RQ in WTRDRSQ is off (set to zero). Upon exit, RQOSESUP should be in the off state also.

The OSE passed to IATUX45 is a copy of the disk OSE. You can access this information without using the IATXOSUP macro. Changes made to this OSE are not made to the disk OSE chain or in any storage representation of the disk OSE.

If you code the IATXOSUP macro in this installation exit, follow the step-by-step process in the description for the IATXOSUP macro.

Refresh Status

This user exit is refreshable through the command *MODIFY,X,M=IATxxxx,REFRESH.
Information Supplied on Exit

The contents of the registers on exit should be:

- **R0-1**: Irrelevant
- **R2-13**: Same as entry
- **R14**: Return address
- **R15**: Irrelevant

Use the RC= parameter on the ARETURN macro to specify normal return or dummy exit as follows:

- **RC=0**: Normal return
- **RC=4**: Reserved
- **RC=8**: Reserved
- **RC=12**: Reserved
- **RC=16**: Dummy exit routine; this exit will not be called again until the system is restarted.

CSECT Name: **IATUX46** (Select Processors Eligible for C/I Processing)

This exit is called by the job segment scheduler (JSS) when JSS is ready to schedule a job for converter/interpreter (C/I) processing. When your installation has defined C/I FSS address spaces, this exit is called for each job to be scheduled for C/I processing. JES3 has already determined which mains are eligible to do the C/I processing. The methods JES3 uses to determine the mains are described in the topic “Defining and Managing C/I Service” in [*z/OS JES3 Initialization and Tuning*](#).

This exit, called from module IATIICS, allows you to override JES3’s decision and provide your own list of mains that are eligible to do the converter/interpreter, prescan, and locate phases of a job’s C/I processing.

If your installation does not define any C/I FSS address spaces, JES3 marks this exit routine as a dummy at initialization and never calls it, because without any C/I FSSs, all C/I processing will occur on the global main.

Environment

This exit routine runs enabled, in key 1, supervisor state, in the JES3 global address space under the JSS FCT.

This exit is called in 31-bit addressing mode. The entry point in register 15 is a 31-bit address.

Linkage Conventions

This exit routine is invoked with the ACALL macro using ASAVE linkage. To return from the exit, use the ARETURN macro without the RC= parameter.

Information Supplied on Entry

The contents of the registers on entry are:

- **R0-3**: Irrelevant
- **R4**: Address of the job’s JQE
- **R5-6**: Irrelevant
- **R7**: Address of the job’s JCT
Installation Exit 46

<table>
<thead>
<tr>
<th>Register</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R8-10</td>
<td>Irrelevant</td>
</tr>
<tr>
<td>R11</td>
<td>IATYFCT address</td>
</tr>
<tr>
<td>R12</td>
<td>IATYFTVT address</td>
</tr>
<tr>
<td>R13</td>
<td>Address of the interpreter data area (IATYIDA)</td>
</tr>
<tr>
<td>R14</td>
<td>Return address</td>
</tr>
<tr>
<td>R15</td>
<td>Entry point address for IATUX46</td>
</tr>
</tbody>
</table>

**Programming Considerations**

The exit routine must be reentrant. It runs under the JSS FCT and therefore should do as little processing as possible.

The installation exit main mask (field IDAUXMMK of data area IATYIDA, pointed to by register 13) shows which processors are IPLed; are online; have at least one active C/I FSS address space with available CI DSPs; and have not been eliminated from C/I eligibility because of the CIDEMAND/CIBATCH specification on the CLASS or STANDARDS initialization statement. The lowest-order bit in the first byte of the mask represents the main defined by the first MAINPROC initialization statement in the initialization stream; the second lowest-order bit represents the main defined by the second MAINPROC initialization statement; and so on. The installation exit routine may eliminate processors from this mask by changing the bit setting to ‘0’ but may not add any.

**Refresh Status**

This user exit is refreshable through the command *MODIFY,X,M=IATxxxx,REFRESH.

**Information Supplied on Exit**

The contents of the registers on exit should be:

<table>
<thead>
<tr>
<th>Register</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R0-1</td>
<td>Irrelevant</td>
</tr>
<tr>
<td>R2-13</td>
<td>Same as on entry</td>
</tr>
<tr>
<td>R14</td>
<td>Return address</td>
</tr>
<tr>
<td>R15</td>
<td>Return code:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Schedule the job for C/I processing in the JES3 global address space (assuming the global has not been made ineligible for scheduling because of the CIDEMAND/CIBATCH specification) or in any of the processors represented by the mask as available.</td>
</tr>
<tr>
<td>4</td>
<td>Do not schedule the job for C/I processing in the JES3 global address space. Use only the processors represented by the mask as available.</td>
</tr>
<tr>
<td>16</td>
<td>Schedule the job for C/I processing in the JES3 global address space (assuming the global has not been made ineligible for scheduling because of the CIDEMAND/CIBATCH specification) or in any of the processors represented by the mask as available. Mark this exit as a “dummy”; do not call it again.</td>
</tr>
<tr>
<td>Other</td>
<td>Same as a return code of 0.</td>
</tr>
</tbody>
</table>
CSECT Name: IATUX48 (Override Operator Modification of Output Data Sets)

This exit, called from IATMOOI, allows you to have the final say on whether a particular data set can be modified by the system operator using the MODIFY command. Before the data set is modified by the operator’s command (which might be to delete the data set), this exit allows you to override the operator’s action. This exit can be an effective tool for enhancing the security of your data sets.

IATUX48 receives control with pointers to the current OSE (in read-only mode) and to the work area (in read-write mode). The exit can use this information to prevent the modification request or to change the modification specifications on the command using the work area.

If you do not supply your own installation exit routine for IATUX48, the IBM-supplied installation exit routine sets the return code to 8 and allows the data set to be modified.

This exit is called separately for each data set associated with a job.

Environment

This exit runs in the JES3 global address space.

This exit is called in 31-bit addressing mode. The entry point in register 15 is a 31-bit address.

Linkage Conventions

This exit routine is invoked with the ACALL macro using ASAVE linkage. To return from the exit, use the ARETURN macro without the RC= parameter.

Information Supplied on Entry

The contents of the registers on entry are:

<table>
<thead>
<tr>
<th>Register</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R0</td>
<td>Irrelevant</td>
</tr>
<tr>
<td>R1</td>
<td>Address of a read-only copy of the current OSE.</td>
</tr>
<tr>
<td>R2-10</td>
<td>Irrelevant</td>
</tr>
<tr>
<td>R11</td>
<td>IATYFCT address</td>
</tr>
<tr>
<td>R12</td>
<td>IATYTVT address</td>
</tr>
<tr>
<td>R13</td>
<td>Pointer to the MODIFY command’s work area - mapped by macro IATYMOOS</td>
</tr>
<tr>
<td>R14</td>
<td>Return address</td>
</tr>
<tr>
<td>R15</td>
<td>Entry point address for IATUX48</td>
</tr>
</tbody>
</table>

Programming Considerations

This exit has the potential for being called twice for a job when the modification is allowed: once for the MOSE, and a second time for the actual OSE on the spool.

Refresh Status

This user exit is refreshable through the command *MODIFY,X,M=IATxxxx,REFRESH.

Information Supplied on Exit

The contents of the registers on exit should be:
Installation Exit 48

<table>
<thead>
<tr>
<th>Register</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R0-1</td>
<td>Irrelevant</td>
</tr>
<tr>
<td>R2-13</td>
<td>Same as on entry</td>
</tr>
<tr>
<td>R14</td>
<td>Return address</td>
</tr>
<tr>
<td>R15</td>
<td>Return code:</td>
</tr>
</tbody>
</table>

- **0**: Reject the request to modify the current OSE. Then go to the next OSE.
- **4**: Accept the modify request, and continue processing with the new values, if any.
- **8**: Accept the modify request, and continue processing with the new values, if any. Then mark the exit as a dummy so that it is not called again until JES3 is restarted.

**CSECT Name: IATUX49 (Override the Address Space Selected for C/I Processing)**

The converter/interpreter (C/I) scheduling module IATIICS calls this exit routine after JES3 selects an address space for a job's C/I processing. IATUX49 enables you to accept or reject the address space selected. JES3, or your installation if installation exit IATUX46 is used, has already determined which mains are eligible for C/I processing of the job.

For example, you may have certain large jobs that have a large number of DD statements and should run only in the FSS address space or you may have jobs that require fast turn-around and should run in the JES3 global address space. If JES3 selects an address space for the C/I processing for the job, you can use IATUX49 to reject JES3's choice. JES3 then goes through the selection process once more, and calls IATUX49 with a different address space. When JES3 finally selects the address space you want for the job, then your exit can accept it, and C/I processing will proceed.

From the information given on entry to the exit routine, you can determine which address space JES3 has selected to process a job's JCL. By using the return codes defined below, you indicate whether you accept or reject JES3’s selection of the address space.

If the installation does not define any C/I FSS address spaces, JES3 marks this exit routine as a dummy at initialization and never calls it, because with no C/I FSSs, C/I processing will occur in the JES3 address space on the global main.

**Environment**

This exit routine runs under the JSS FCT in the JES3 global address space. It runs enabled, in key 1, supervisor state.

This exit is called in 31-bit addressing mode. The entry point in register 15 is a 31-bit address.

**Linkage Conventions**

This exit routine is invoked with the ACALL macro using ASAVE linkage. To return from the exit, use the ARETURN macro without the RC= parameter.

**Information Supplied on Entry**

The contents of the registers on entry are:
Programming Considerations

The exit routine must be reentrant. Because it runs under the JSS FCT, it should do as little processing as possible.

Register 1 contains, on entry, the address of the C/I FSS table (IATYCFT) for the selected C/I FSS address space or zero for the JES3 global address space. The C/I FSS table contains two fields indicating the number of CI DSPs active in the C/I FSS address space:
- Field CFTBATUS, for batch CI DSPs
- Field CFTDSLUS, for demand select CI DSPs

Field CFTSSPT of the C/I FSS table points to its corresponding FSS table (IATYFSS), and the FSS table points back to the C/I FSS table. Field FSSASID of the FSS table contains the address space id of the C/I FSS address space.

If the return code from this exit routine indicates that the selected address space should be rejected, JSS selects the C/I FSS address space with the next greatest number of available CI DSPs. (JSS excludes the C/I FSS address space just rejected.) This process continues until the exit routine accepts a C/I FSS address space or no more C/I FSS address spaces remain to be chosen. If JSS cannot find another available address space, JSS does not schedule the job for C/I processing during this scheduling pass through the job queue.

Note: See *z/OS JES3 Initialization and Tuning Reference* for the method by which JES3 selects an address space for a job’s C/I processing.

Refresh Status

This user exit is refreshable through the command

*MODIFY,X,M=IATxxxx,REFRESH.

Information Supplied on Exit

The contents of the registers on exit should be:

<table>
<thead>
<tr>
<th>Register</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R0-10</td>
<td>Irrelevant</td>
</tr>
<tr>
<td>R11-13</td>
<td>Same as on entry</td>
</tr>
<tr>
<td>R14</td>
<td>Return address</td>
</tr>
<tr>
<td>R15</td>
<td>Return code:</td>
</tr>
<tr>
<td>0</td>
<td>Use the selected address space and continue processing the job.</td>
</tr>
<tr>
<td>4</td>
<td>Reject the selected address space and select another one.</td>
</tr>
</tbody>
</table>
**CSECT Name: IATUX50 (JES3 Unknown BSID Modifier Exit)**

This exit allows an installation to define codes for the BSIDMOD field in the BDT subsystem interface data area (BSID). The BSIDMOD field specifies the purpose and usage of each BSID (message, JES3 command, transaction, shuttle staging area, and others). This exit can recognize and process user-assigned codes specified for BSIDMOD fields.

This exit is called by IATBDCI.

See *MVS/Bulk Data Transfer Writing and Installing User Exits* for additional information.

**Environment**

This exit runs in the JES3 global address space.

This exit is called in 31-bit addressing mode. The entry point in register 15 is a 31-bit address.

**Linkage Conventions**

This exit routine is invoked with the ACALL macro using ASAVE linkage. To return from the exit, use the ARETURN macro without the RC= parameter.

**Information Supplied on Entry**

The contents of the registers on entry are:

- **R0**: Irrelevant
- **R1**: Address of a one-word parameter list
- **word 1**: Address of the BSID, mapped by macro IATYBSID, contains a BSIMOD code not recognized by MVS/BDT
- **R2-10**: Irrelevant
- **R11**: IATYFCT address
- **R12**: IATYTVT address
- **R13**: Irrelevant
- **R14**: Return address
- **R15**: Entry point address into IATUX50

**Programming Considerations**

None.

**Refresh Status**

This user exit is refreshable through the command `*MODIFY,X,M=IATxxxx,REFRESH.`
Information Supplied on Exit

The contents of the registers on exit should be:

<table>
<thead>
<tr>
<th>Register</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R0-1</td>
<td>Irrelevant</td>
</tr>
<tr>
<td>R2-13</td>
<td>Same as on entry</td>
</tr>
<tr>
<td>R14</td>
<td>Return address</td>
</tr>
<tr>
<td>R15</td>
<td>Return code set by IATUX50:</td>
</tr>
</tbody>
</table>

- 0: BSIDMOD code value is not recognized as being user-defined or is determined to be incorrect by IATUX50.
- 4: BSIDMOD code value is recognized as user-defined.
- 8: Set by the IBM-provided dummy exit; IATBDCI disables IATUX50 for further invocations.
- Other: Error message is written to operator. The exit will not be called again until the system is restarted.

CSECT Name: IATUX57 (Select a Single WTO Routing Code for JES3)

When JES3 receives a WTO macro (through the subsystem interface), JES3 changes the routing code on the WTO to its matching JES3 destination class. Where a WTO specifies more than one routing code, JES3 module IATSIWO uses a predefined algorithm (see [Table 6 on page 41]) to select a single routing code to convert to a destination class.

However, you can override the JES3 algorithm and use your own with installation exit 57. IATUX57 allows you to select a single routing code to be used by JES3. (However, if the WTO specifies SUBSMOD=NO, then installation exit 57 is not called by JES3.)

After a routing code has been selected, regardless of the method used, the WTO is processed by JES3 according to the routing information defined with JES3 message route processing. JES3 converts the single routing code to the corresponding destination class (as set up by the MSGROUTE initialization statement, if any), and then routes the message to the devices that are assigned to that class.

Environment

This installation exit routine runs in the address space of the issuer of the WTO macro. It can run on a local processor.

This exit is called in 31-bit addressing mode. The entry point in register 15 is a 31-bit address.

Linkage Conventions

This installation exit is called from module IATSIWO using the LINK macro.

Programming Considerations

This installation exit must reside in the LPA, and must be reentrant.

Your exit routine cannot issue a WTO macro.
Control Blocks Accessed by this Exit

IATYUX57

Information Supplied on Entry

On entry to exit 57, register 1 points to a parameter list (mapped by macro IATYUX57) that contains a number of fields. However, the only field that may be changed by this exit is the field YUX57RCD, containing the routing codes specified on the incoming WTO.

This field is 16 bytes long. Each of the 128 bits in the field corresponds to one MVS routing code. If a bit is on, it means that the routing code is specified on the WTO. When exit 57 finishes, only one of the 128 bits must be on. This single bit indicates the routing code to be used by JES3.

The contents of the registers on entry are:

- R0: Irrelevant
- R1: Pointer to IATYUX57 parameter list that contains the 16-byte field showing which routing codes are present.
- R2-10: Irrelevant
- R11: Address of the SSVT
- R12: Irrelevant
- R13: Save area address
- R14: Return address
- R15: Entry point address for IATUX57

Usage Notes

On return from the installation exit, IATSIWO will use only the routing code selected by the installation exit.

Registers 0 to 14 must be restored on exit.

If your exit routine selects more than one routing code, makes an incorrect selection, or does not supply a routing code when a return code indicates that one has been specified, then IATSIWO will select a routing code based on the original routing codes using the algorithm shown in Table 6 on page 41.

Other Considerations:

- Because MSGROUTE processing for a message is performed only on the processor issuing the message, IATUX57 is not called for a WTO on the global processor if the WTO originated on the local processor.
- When considering the use of installation exit 57, check the information on the MSGROUTE initialization statement, and also check if your installation is using MPF, or message exits in MVS code.

Refresh Status

This user exit is not refreshable through the command *MODIFY,X,M=IATxxxx,REFRESH.

Information Supplied on Exit

The contents of the registers on exit must be:

- R0-12: Irrelevant
- R13: Same as on entry
- R14: Return address
CSECT Name: IATUX58 (Modify Security Information Before JES3 Security Processing)

This exit is called from IATGRSC before JES3 calls the security authorization facility (SAF) to perform security processing using RACF. This exit gives you the opportunity to modify security checks or to make security decisions for JES3. The security decisions you can control include restricting access to SYSIN/SYSOUT data sets, controlling the use of operator commands, and controlling network jobs and data.

See Appendix A, “Information for JES3 Security Calls,” on page 607 for information about:

- Specific points in JES3 processing where this exit is called
- Information that is supplied to the exit
- Information that the exit can modify

Environment

This exit runs in either the user address space or the JES3 global address space.

This exit is called in 31-bit addressing mode. The entry point in register 15 is a 31-bit address.

Linkage Conventions

This exit is called from IATGRSC with a BALR instruction. You must save and restore the registers. To return to the calling module, use either a BR 14 instruction or a RETURN macro. Do not use the ARETURN macro.

Information Supplied on Entry

The contents of the registers on entry are:

- **R0**: Irrelevant
- **R1**: Address of the parameter list (IATYSSX mapping macro)
- **R2-12**: Irrelevant
- **R13**: Address of 18-word save area
- **R14**: Return address
- **R15**: Entry point address into IATUX58.

Programming Considerations

Register 1 points to a parameter list (defined by macro IATYSSX). Values at the following labels fall into two categories: those that can be modified and those that can only be read.

The following list shows labels whose values can be modified:
SSXATTR
Specifies the access authority of the user or group permitted access to the JES3 resource for which authorization checking is to be performed:

- SSXAALTR – User or group has total control over the resource.
- SSXACNTL – For VSAM data sets, the user or group has authority equivalent to the VSAM control password. For non-VSAM data sets and other resources, the user or group has UPDATE authority.
- SSXAPUPDT – User or group can open the resource to read or write.
- SSXAREAD – User or group can open the resource to read only.

See z/OS JES3 Initialization and Tuning Guide for information about how these authorization levels affect the specific JES3 resources (SSXCLASS).

SSXEXNOD
A nine-byte field where the first byte is the length of the execution node, and the remaining eight bytes contains the execution node name. The execution node is the node where the job executed.

SSXENTIT
Specifies the resource name for which RACF security processing is to be performed. For example, for an *INQUIRY,X command the resource name is JES3.DISPLAY.X

If a TSO user cancels a job, the resource name is CANCEL.node.userid.jobname

The resource name can be from zero to 53 bytes long.

SSXGROUP
A nine-byte field where the first byte is the length of the group name, and the remaining eight bytes contains the RACF group name. See your RACF administrator for valid RACF group names for your installation.

SSXJOBNM
Specifies the eight-character job name.

SSXLOG
Specifies how SAF or RACF should log the reason for the IATXSEC call (SSXLGSTR) in the SMF data sets.

SSXASIS
RACF should record the events in the manner specified in the profile that protects the resource. See your RACF administrator for more information about RACF profiles.

SSXNFAIL
If the authorization check fails, the attempt is not recorded. If the authorization check succeeds, the attempt is recorded in the manner specified in the RACF profile that protects the resource.

SSXNSTAT
The attempt is not recorded and no resource statistics are updated.

SSXNONE
The event is not recorded.

SSXLGSTR
Specifies a character string that RACF adds to the SMF record. This character string indicates why the IATXSEC macro call was made. The SMF log string can be from zero to 255 bytes long. The first byte must contain the length of the string.
SSXNPASS
A nine-byte field where the first byte is the length of the user’s new password, and the remaining eight bytes contains the new password.

SSXPASCK
Specifies whether the password should be checked by RACF.

SSXPCYES
RACF should check the password.

SSXPCNO
RACF should not check the password.

SSXPASWD
A nine-byte field where the first byte is the length of the password, and the remaining eight bytes contain the password.

SSXPOE
Specifies an eight-byte field containing the name of the input device from which the job was submitted. For example, the input device for a job submitted through a card reader is the SUPUNIT DD name of the card reader device.

SSXRECVR
Specifies an eight-byte field containing the userid of the user who has the authority to access the resource when there are no profiles for the resource. For example, in normal cases, if user A sends a data set to user B, user B would have to be authorized to access user A’s data sets. However, by specifying RECVR=useridB, user B can receive the data set without being authorized.

SSXRTOKN
Specifies a resource security token. The resource security token is the user token the creator of the resource had at the time the resource was created. For example, when a TSO user cancels a job, the resource security token is the user token of the person who submitted the job in the first place.

SSXSECLB
Specifies the eight-byte field containing the security label. The security label represents the association between a particular security level (for example, FOR YOUR EYES ONLY) and a set of security categories (for example, CONTROL and KAOS). For information about the security labels for your installation, contact your RACF administrator.

SSXSGRP
A nine-byte field where the first byte is the length of the submittor’s RACF group name, and the remaining eight bytes contain the submittor’s RACF group name. See your RACF administrator for valid RACF group names.

SSXSNODE
A nine-byte field where the first byte is the length of the submittor’s node name, and the remaining eight bytes contain the submittor’s node name.

SSXSSTOKN
Specifies the submittor’s security token.

SSXSUSRIR
A nine-byte field where the first byte is the length of the submittor’s userid and the remaining eight bytes contain the submittor’s userid.

SSXSSSION
Specifies the type of session as one of the following:
• SSXSSEXEB – External batch session
Installation Exit 58

- SSXSSINB – Internal batch session
- SSXSSNJB – NJE batch session
- SSXSSRJB – RJE batch session
- SSXSSNJO – NJE operator session
- SSXSSRJO – RJE operator session
- SSXSSSTR – Started task session
- SSXSSTSO – TSO session
- SSXSSNJS – NJE SYSOUT session
- SSXSSTKU – NJE unknown user session

SSXTOKIN
Specifies the input security token.

SSXTOKOT
Specifies the output security token.

SSXTRUST
Specifies whether the user is a member of the trusted computer base.

SSXTRYES
The user is a member of the trusted computer base.

SSXTRNO
The user is not a member of the trusted computer base.

SSXUSERI
A nine-byte field where the first byte is the length of the userid and the remaining eight bytes contain the userid.

SSXUTOKN
Specifies the user security token. The user token represents the person that requires authorization to the resource. For example, when a TSO user cancels a job, the user security token represents the TSO user performing the cancel.

SSXWORKA
Specifies the address of the security authorization facility (SAF) work area.

SSXPSSCS
Address of the Cancel SSOB extension used when IATXSEC is used to authorize a TSO CANCEL command.

SSXPTMID
Address of the TSO terminal identifier used when IATXSEC is used to authorize a TSO CANCEL command.

The following list shows labels whose values can only be read:

SSXCLASS
Specifies the name of the RACF class as one of the following:

Table 13. RACF Classes Used to Protect JES3 Resources

<table>
<thead>
<tr>
<th>Resource:</th>
<th>RACF Class:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Job data sets (SYSIN/SYSOUT)</td>
<td>JESSPOOL</td>
</tr>
<tr>
<td>JES3 writers</td>
<td>WRITER</td>
</tr>
<tr>
<td>TSO SUBMIT/CANCEL</td>
<td>JESJOBS</td>
</tr>
<tr>
<td>Commands (other than RJP/NJE)</td>
<td>OPERCMDS</td>
</tr>
<tr>
<td>USERIDS</td>
<td>USER</td>
</tr>
</tbody>
</table>
SSXMCNTL
Specifies how JES3 should handle messages from SAF, installation exit IATUX58, or installation exit IATUX59.

SSXMCWTO
Specifies that SAF or the installation exit should write the message to the operator.

SSXMCRTN
Specifies that messages should be returned to the caller.

SSXMCJES
Specifies that messages should be written to the job’s JESMSGLG data set. This is done by module IATGRSC.

SSXMODE
Specifies the phase of JES3 processing the caller of IATXSEC is in as one of the following:

SSXNUCMD
The caller is running under the JES3 NUC task.

SSXINIMD
IATXSEC is being issued during JES3 initialization.

SSXSTKMD
The caller is running under a JES3 subtask.

SSXUSRMD
The caller is running in a user’s address space.

SSXFRMOT
Specifies the format for the output token (SSXTOKOT) as one of the following:

SSXFOINT
The security product should convert the token from the external to the internal (encrypted) format.

SSXFOEXT
The security product should convert the token from the internal (encrypted) to the external format.

SSXENCERY
For a REQUEST=EXTRACT, specifies the address of the data to be encrypted. The first byte of the address is the length of the data.

SSXENCRT
For a REQUEST=VERIFYX, specifies whether the password needs to be encrypted.

SSXENCYS
Indicates that the password needs to be encrypted.

SSXENCNO
Indicates that the password does not need to be encrypted.

SSXNJEJH
Specifies the address of the NJE job header.

SSXNJEDH
Specifies the address of the NJE data set header.

SSXWPSLC
Specifies the number of different types of selection characteristics that are specified (length of SSXWPSLM).
SSXWJNAM
   Specifies the 8-character job name.

SSXWJBID
   Specifies the 8-character job identifier.

SSXDDSN
   Specifies the 24-character writer DDNAME.

SSXWPSLM
   Specifies the 16-byte selection mask.

SSXOPRTY
   Specifies the priority of the job’s output.

SSXODEST
   Specifies the 8-byte destination of the job's output.

SSXOMDID
   Specifies the output’s 4-byte copy mod identifier.

SSXOSTCK
   Specifies whether a stacker is required.

SSXOTYPE
   Specifies the 8-byte device type required (for example, PRT3211).

SSXOFRMS
   Specifies the 8-byte type of form required.

SSXOFLSH
   Specifies the 4-byte flash required.

SSXOUCS
   Specifies the 4-byte required UCS identifier.

SSXOCLSS
   Specifies the SYSOUT class.

SSXOMODE
   Specifies the 8-byte process mode.

SSXOFLAG
   Specifies the following (These values are the same as the values of OSEFLAG in IATYOSE):
   OSECMLPT   All output elements are complete.
   OSESCHD    The element is scheduled for output.
   OSEPEND    The output is pending.
   OSEWHOLD   The output is held.
   OSERMTD    The destination is a remote node.
   OSEOPEND   The data set has been processed but is not yet complete.
   OSESYS     The data set is held.
   OSETSO     The TSO data set is held.

SSXOWTRN
   Specifies the external writer name.

SSXOSSSO
   For process SYSOUT (PSO), specifies the address of IEFSSSO.
SSXJRFL1
Specifies where the JESNEWS data set is to be printed.

SSXPRG
Specifies that the JESNEWS data set is to be purged after the current user is finished using it.

SSXLCL
Specifies that the JESNEWS output is to be sent to local printers.

SSXTSO
Specifies that the JESNEWS output is to be sent to TSO users.

SSXRJP
Specifies that the JESNEWS output is to be sent to remote printers (RJP).

SSXDSN
Specifies that the JESNEWS output is to be sent to local, TSO, and RJP printers.

SSXNEWFL
Specifies the type of JESNEWS function requested.

SSXJNEW
Indicates a request to add to the JESNEWS data set.

SSXJREP
Indicates a request to replace the JESNEWS data set.

SSXJDEL
Indicates a request to delete from the JESNEWS data set.

SSXJTYP
Indicates that nothing is to be done with the JESNEWS data set.

SSXPRCS
Indicates that JESNEWS was started with the //*PROCESS statement.

SSXPWD
Indicates that the JESNEWS password was entered correctly.

Refresh Status
This user exit is not refreshable through the command *MODIFY,X,M=IATxxxx,REFRESH.

Information Supplied on Exit
The contents of the registers on exit must be:

- **R0**: Irrelevant
- **R1**: Pointer to a chain of messages that are mapped by IATYSEMS and must reside in subpool SAFMSGSP (defined in IATYEQU).
- **R2-12**: Irrelevant
- **R13**: Same as on entry
- **R14**: Return address
- **R15**: Return code set by IATUX58:

**Note**: A valid user token must be provided before IATUX58 may return with a return code of 0, 4, or 8. For information on user tokens, see either the z/OS Security Server RACF General User's Guide or the z/OS Security Server RACF Security Administrator's Guide.
The exit satisfied the request (by using installation-defined processing within the exit itself). JES3 does not call the security authorization facility (SAF).

4 The exit allowed processing for the request to occur using facilities outside the exit that did not involve SAF or RACF. JES3 does not call the security authorization facility (SAF).

8 The exit did not satisfy the security request. JES3 does not call the security authorization facility (SAF).

12 Allow JES3 security processing (through RACF) to make the security decision. After JES3 security processing, call installation exit IATUX59.

16 Allow JES3 security processing to make the security decision (through RACF). After JES3 security processing, do not call installation exit IATUX59.

20 Dummy installation exit; the exit will not be called again until the system is restarted.

CSECT Name: IATUX59 (Modify Security Information After JES3 Security Processing)

This exit is called from IATGRSC after JES3 calls the security authorization facility (SAF) to perform security processing using RACF. This exit gives you the opportunity to modify security checks or to make security decisions for JES3.

The security decisions you can control include restricting access to SYSIN/SYSOUT data sets, controlling the use of operator commands, and controlling network jobs and data.

See Appendix A, “Information for JES3 Security Calls,” on page 607 for information about:

- Specific points in JES3 processing where this exit is called
- Information that is supplied to the exit
- Information that the exit can modify

Environment

This exit runs in either the user address space or the JES3 global address space.

This exit is called in 31-bit addressing mode. The entry point in register 15 is a 31-bit address.

Linkage Conventions

This exit is called from IATGRSC with a BAKR instruction. To return to the calling module, use a PR instruction. Do not use the ARETU RN macro.

Information Supplied on Entry

The contents of the registers on entry are:

- R0 Irrelevant
- R1 Address of the parameter list (IATYSSX)
Programming Considerations

Register 1 points to a parameter list (defined by macro IATYSSX). Values at the following labels fall into two categories: those that can be modified and those that can only be read.

The following list shows labels whose values can be modified:

**SSXATTR**
- Specifies the access authority of the user or group permitted access to the JES3 resource for which authorization checking is to be performed:
  - SSXAALTR – User or group has total control over the resource.
  - SSXACNTL – For VSAM data sets, the user or group has authority equivalent to the VSAM control password. For non-VSAM data sets and other resources, the user or group has UPDATE authority
  - SSXAUPDT – User or group can open the resource to read or write.
  - SSXAREAD – User or group can open the resource to read only.

See [z/OS JES3 Initialization and Tuning Guide](https://www.ibm.com/support/knowledgecenter/SSEKSS_2.1.1/com.ibm.zos.r21.jes3is.doc/ae/pfinteg.html) for information about how these authorization levels affect the specific JES3 resources (SSXCLASS).

**SSXEXNOD**
- A nine-byte field where the first byte is the length of the execution node, and the remaining eight bytes contains the execution node name. The execution node is the node where the job executed.

**SSXENTIT**
- Specifies the resource name for which RACF security processing is to performed. For example, for an *INQUIRY,X command the resource name is JES3.DISPLAY.X

If a TSO user cancels a job, the resource name is CANCEL.node.userid.jobname

The resource name can be from zero to 53 bytes long.

**SSXGROUP**
- A nine-byte field where the first byte is the length of the group name, and the remaining eight bytes contains the RACF group name. See your RACF administrator for valid RACF group names for your installation.

**SSXJOBNM**
- Specifies the eight-character job name.

**SSXLOG**
- Specifies how SAF or RACF should log the reason for the IATXSEC call (SSXLGSTR) in the SMF data sets.

**SSXASIS**
- RACF should record the events in the manner specified in the profile that protects the resource. See your RACF administrator for more information about RACF profiles.

**SSXNFAIL**
- If the authorization check fails, the attempt is not recorded. If the
authorization check succeeds, the attempt is recorded in the manner specified in the RACF profile that protects the resource.

SSXNSTAT
The attempt is not recorded and no resource statistics are updated.

SSXNONE
The event is not recorded.

SSXLGSTR
Specifies the address of a character string that RACF adds to the SMF record. This character string indicates the IATXSEC macro call was made. The SMF log string can be from zero to 255 bytes long. The first byte must contain the length of the string.

SSXNPASS
A nine-byte field where the first byte is the length of the user's new password, and the remaining eight bytes contains the new password.

SSXPASCK
Specifies whether the password should be checked by RACF.

SSXPCYES
RACF should check the password.

SSXPCNO
RACF should not check the password.

SSXPASWD
A nine-byte field where the first byte is the length of the password, and the remaining eight bytes contain the password.

SSXPOE
Specifies an eight-byte field containing the name of the input device from which the job was submitted. For example, the input device for a job submitted through a card reader is the SUPUNIT DD name of the card reader device.

SSXRECVR
Specifies an eight-byte field containing the userid of the user who has the authority to access the resource when there are no profiles for the resource. For example, in normal cases, if user A sends a data set to user B, user B would have to be authorized to access user A's data sets. However, by specifying RECVR=useridB, user B can receive the data set without being authorized.

SSXRTOKN
Specifies a resource security token. The resource security token is the user token the creator of the resource has at the time the resource was created. For example, when a TSO user cancels a job, the resource security token is the user token of the person who submitted the job in the first place.

SSXSECLB
Specifies the eight-byte field containing the security label. The security label represents the association between a particular security level (for example, FOR YOUR EYES ONLY) and a set of security categories (for example, CONTROL and KAOS). For information about the security labels for your installation, contact your RACF administrator.

SSXSGRP
A nine-byte field where the first byte is the length of the submittor's group name, and the remaining eight bytes contain the submittor's RACF group name. See your RACF administrator for valid RACF group names.
SSXSNODE
A nine-byte field where the first byte is the length of the submittor’s node name, and the remaining eight bytes contain the submittor’s node name.

SSXSTOKN
Specifies the submittor’s security token.

SSXSUSRI
A nine-byte field where the first byte is the length of the submittor’s userid and the remaining eight bytes contain the submittor’s userid.

SSXSSION
Specifies the type of session as one of the following:
- SSXSSEXB – External batch session
- SSXSSINB – Internal batch session
- SSXSNJB – NJE batch session
- SSXSSRJF – RJE batch session
- SSXSSNJO – NJE operator session
- SSXSSRJO – RJE operator session
- SSXSSSTR – Started task session
- SSXSTTSO – TSO session
- SSXSSNJE – NJE SYSOUT session
- SSXSTKU – NJE unknown user session

SSXTOKIN
Specifies the input security token.

SSXTOKOT
Specifies the output security token.

SSXTRUST
Specifies whether the user is a member of the trusted computer base.

SSXTRYES
The user is a member of the trusted computer base.

SSXTRNO
The user is not a member of the trusted computer base.

SSXUSERI
A nine-byte field where the first byte is the length of the userid and the remaining eight bytes contain the userid.

SSXUTOKN
Specifies the user security token. The user security token represents the person that requires authorization to the resource. For example, when a TSO user cancels a job, the user security token represents the TSO user performing the cancel.

SSXWORKA
Specifies the address of the security authorization facility (SAF) work area.

SSXPSSCS
Address of the Cancel SSOB extension used when IATXSEC is used to authorize a TSO CANCEL command.

SSXPTMID
Address of the TSO terminal identifier used when IATXSEC is used to authorize a TSO CANCEL command.

SSXJRFL1
Specifies where the JESNEWS data set is to be printed.
SSXPRG
Specifies that the JESNEWS data set is to be purged after the current user is finished using it.

SSXLCL
Specifies that the JESNEWS output is to be sent to local printers.

SSXTSO
Specifies that the JESNEWS output is to be sent to TSO users.

SSXRJP
Specifies that the JESNEWS output is to be sent to remote printers (RJP).

SSXDSN
Specifies that the JESNEWS output is to be sent to local, TSO, and RJP printers.

SSXNEWFL
Specifies the type of JESNEWS function requested.

SSXJNEW
Indicates a request to add to the JESNEWS data set.

SSXJREP
Indicates a request to replace the JESNEWS data set.

SSXJDEL
Indicates a request to delete from the JESNEWS data set.

SSXJTYP
Indicates that nothing is to be done with the JESNEWS data set.

SSXPRCS
Indicates that JESNEWS was started with the /*PROCESS statement.

SSXPWD
Indicates that the JESNEWS password was entered correctly.

The following list shows labels whose values can only be read:

SSXCLASS
Specifies the name of the RACF class as one of the following:

<table>
<thead>
<tr>
<th>Resource:</th>
<th>RACF Class:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Job data sets (SYSIN/SYSOUT)</td>
<td>JESSPOOL</td>
</tr>
<tr>
<td>JES3 writers</td>
<td>WRITER</td>
</tr>
<tr>
<td>TSO SUBMIT/CANCEL</td>
<td>JESJOBS</td>
</tr>
<tr>
<td>Commands (other than RJP/NJE)</td>
<td>OPERCMDS</td>
</tr>
<tr>
<td>USERIDS</td>
<td>USER</td>
</tr>
</tbody>
</table>

SSXMCNTL
Specifies how JES3 should handle messages from SAF, installation exit IATUX58, or installation exit IATUX59.

SSXMCWTO
Specifies that SAF or the installation exits should write messages to the operator.

SSXMCRTN
Specifies that messages should be returned to the caller.
SSXMCJES
Specifies that messages should be written to the job’s JESMSGGLG data set. This is done by module IATGRSC.

SSXMODE
Specifies the phase of JES3 processing the caller of IATXSEC is in as one of the following:

SSXNUCMD
The caller is running under the JES3 NUC task.

SSXINIMD
IATXSEC is being issued during JES3 initialization.

SSXSTKMD
The caller is running under a JES3 subtask.

SSXUSRMD
The caller is running in a user’s address space.

SSXFRMOT
Specifies the format for the output token (SSXTOKOT) as one of the following:

SSXFOINT
The security product should convert the token from the external to the internal (encrypted) format.

SSXFOEXT
The security product should convert the token from the internal (encrypted) to the external format.

SSXENCRT
For a REQUEST=EXTRACT, specifies the address of the data to be encrypted. The first byte of the address is the length of the data.

SSXENCRT
For a REQUEST=VERIFYX, specifies whether the password needs to be encrypted.

SSXENCYS
Indicates that the password needs to be encrypted.

SSXENCNO
Indicates that the password does not need to be encrypted.

SSXNJJEJH
Specifies the address of the NJE job header.

SSXNJEDH
Specifies the address of the NJE data set header.

SSXWPSLC
Specifies the number of different selection characteristics that are being used (length of SSXWPSLM).

SSXWJNAM
Specifies the 8-character job name.

SSXWJBD
Specifies the 8-character job identifier.

SSXDDSN
Specifies the 24-character writer DDNAME.
SSXWPSLM
Specifies the 16-byte selection mask.

SSXOPRTY
Specifies the priority of the job’s output.

SSXODEST
Specifies the 8-byte destination of the job’s output.

SSXOMDID
Specifies the output’s 4-byte copy mod identifier.

SSXOSTCK
Specifies whether a stacker is required.

SSXOTYPE
Specifies the 8-byte device type required (for example, PRT3211).

SSXOFRMS
Specifies the 8-byte type of form required.

SSXOFLSH
Specifies the 4-byte flash required.

SSXOUCS
Specifies the 4-byte required UCS identifier.

SSXOCLSS
Specifies the SYSOUT class.

SSXOMODE
Specifies the 8-byte process mode.

SSXOFLAG
Specifies the following (These values are the same as the values of OSEFLAG in IATYOSE):

OSECMLPT  All output elements are complete.
OSESCHD  The element is scheduled for output.
OSEPEND  The output is pending.
OSEWHOLD  The output is held.
OSERMTD  The destination is a remote node.
OSEOPEND  The data set has been processed but is not yet complete.
OSESYS  The data set is held.
OSETSO  The TSO data set is held.

SSXOWTRN
Specifies the external writer name.

SSXOSSSO
For process SYSOUT (PSO), specifies the address of IEFSSSO.

Refresh Status
This user exit is not refreshable through the command
*MODIFY,X,M=IATxxxx,REFRESH.

Information Supplied on Exit
The contents of the registers on exit must be:
CSECT Name: IATUX60 (Determine Action to Take When a TSO User Is Unable to Receive a Data Set)

When JES3 determines that for security reasons a TSO user cannot receive a data set because of the contents of the current profile of the userid, module IATOSPC invokes installation exit IATUX60 to allow the installation to decide what action should be taken.

Environment

This exit runs in the JES3 global address space.

This exit is called in 31-bit addressing mode. The entry point in register 15 is a 31-bit address.

Linkage Conventions

This exit routine is invoked with the ACALL macro using ASAVE linkage. To return from the exit, use the ARETURN macro with the RC= parameter.

Information Supplied on Entry

The contents of the registers on entry are:

- R0: Irrelevant
- R1: Address of parameter list mapped by UX60STRRT in IATUX60
- R2-R10: Irrelevant
- R11: Address of IATYFCT
- R12: Address of IATYTVT
- R13: Irrelevant
- R14: Return address
- R15: Entry point address for IATUX60
Installation Exit 60

Programming Considerations

Register 1 points to a parameter list (defined in IATUX60 at label UX60STRT) that contains at offset:

0   Pointer to the 44-byte data set name
4   Pointer to the user id of the TSO user who is unable to receive the data set.
8   Pointer to the user security token (UTOKEN) of the TSO user who is unable to receive the data set.
12  Pointer to the resource security token (RTOKEN) of the data set.

Refresh Status

This user exit is refreshable through the command

*MODIFY,X,M=IATxxx,REFRESH.

Information Supplied on Exit

The contents of the registers on exit are:

R0-10  Irrelevant
R11-12 Same as on entry
R13  Irrelevant
R14  Return address
R15  Return code

0  Keep the data set on the output service queue.
4  Delete the data set from the output service queue.
8, 12, or 16  Dummy installation exit; this exit will not be invoked again, and all data sets that fail the security call will be deleted.

CSECT Name: IATUX61 (Cancel Jobs Going on the MDS Error Queue)

This installation exit lets you selectively cancel jobs that the main device scheduler (MDS) is going to put on the MDS error queue. Jobs that are put on the MDS error queue must be canceled or restarted by the operator. By canceling jobs before they get to the error queue, this exit reduces the operator actions needed.

This exit is called by IATMDSB.

Environment

This exit runs in the JES3 global address space.

This exit is called in 31-bit addressing mode. The entry point in register 15 is a 31-bit address.

Linkage Conventions

This exit routine is invoked with the ACALL macro using ASAVE linkage. To return from the exit, use the ARETURN macro without the RC= parameter.

Information Supplied on Entry

The input to IATUX61 is a code in register 0 indicating the reason JES3 is putting the job on the MDS error queue, and the RESQUEUE (IATYRSQ) address in register 1.
The register 0 values are (the labels are equates for the values as defined in macro IATYMDS):

<table>
<thead>
<tr>
<th>Code</th>
<th>Label</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>RSNFETCH</td>
<td>The job had an error during fetch processing</td>
</tr>
<tr>
<td>4</td>
<td>RSNALOCQ</td>
<td>The job had an allocation error</td>
</tr>
<tr>
<td>8</td>
<td>RSNVERF</td>
<td>The job had a verification error</td>
</tr>
<tr>
<td>12</td>
<td>RSNREST</td>
<td>The job had an error during MDS restart</td>
</tr>
<tr>
<td>16</td>
<td>RSNFAILS</td>
<td>The error is due to MDS failsoft processing</td>
</tr>
<tr>
<td>20</td>
<td>RSNPRMIO</td>
<td>The job had a permanent JSAM I/O error</td>
</tr>
<tr>
<td>24</td>
<td>RSNNOBUF</td>
<td>No JSAM buffers are available for JESREAD</td>
</tr>
<tr>
<td>28</td>
<td>RSNNOJST</td>
<td>The job had a zero JST FDB</td>
</tr>
<tr>
<td>32</td>
<td>RSNSYSSL</td>
<td>The error occurred during MDSSRS system select processing</td>
</tr>
<tr>
<td>36</td>
<td>RSNSYVER</td>
<td>The error occurred during MDSSRS system verify processing</td>
</tr>
<tr>
<td>40</td>
<td>RSNSRSFL</td>
<td>The error occurred during MDSSRS FAILSOFT processing</td>
</tr>
</tbody>
</table>

**Note:** Values 0 to 28 apply to MDS FCT processing. Values 32, 36, and 40 apply to MDSSRS FCT processing.

The contents of the other registers on entry are:

- **R1**: RESQUEUE address - mapped by macro IATYRSQ
- **R2-10**: Irrelevant
- **R11**: IATYFCT address
- **R12**: IATYTVT address
- **R13**: MDS data area address - mapped by IATYMDS
- **R14**: Return address
- **R15**: Entry point address into IATUX61.

**Note:** The MDSSRS data area (IATYSRS) can be obtained from the TVTMDSRD field in the TVT if it is needed.

**Programming Considerations**

None.

**Refresh Status**

This user exit is refreshable through the command
*MODIFY,X,M=IATxxxx,REFRESH.

**Information Supplied on Exit**

The contents of the registers on exit are:

- **R0-13**: Same as on entry
- **R14**: Return address

The output from this exit routine is a return code in register 15. The valid values are:

- **0**: Place the job on the MDS error queue
- **4**: Cancel the job
- **16**: Dummy exit – place the job on the MDS error queue and do not call this exit again until JES3 is restarted.

If the job is placed on the MDS error queue, the *RESTART,SETUP,nnnn command can be used to restart the job through JES3 setup. The *RESTART,SETUP,nnnn,CI
command can be used to restart the job through C/I processing. For more information on the *RESTART,SETUP command, see [z/OS JES3 Commands](#).

---

**CSECT Name: IATUX62 (Verify a Mount Request)**

Exit routine IATUX62 is a main device scheduler (MDS) routine that verifies a mount request. This exit is invoked after the verification of a tape or disk mount from IATMDVE. The exit gives you the ability to accept or override JES3’s decision about whether a requested mount is valid.

**Environment**

This exit runs in the JES3 global address space.

This exit is called in 31-bit addressing mode. The entry point in register 15 is a 31-bit address.

**Linkage Conventions**

This exit routine is invoked with the ACALL macro using ASAVE linkage. To return from the exit, use the ARETURN macro without the RC= parameter.

**Information Supplied on Entry**

The input to IATUX62 is a code in register 0 indicating JES3’s decision about the mount request, and a parameter list address in register 1.

The register 0 values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>JES3 will accept the mount request unless overridden</td>
</tr>
<tr>
<td>4</td>
<td>JES3 will reject the mount request unless overridden</td>
</tr>
</tbody>
</table>

The contents of the other registers upon entry are:

- **R1**: Parameter list address
- **R2-10**: Irrelevant
- **R11**: IATYFCT address
- **R12**: IATYTVT address
- **R13**: Irrelevant
- **R14**: Return address
- **R15**: Entry point address into IATUX62.
**EXIT 62 Parameter List**

The parameter list pointed to by register 1 is mapped in macro IATYMDS and is defined as follows:

```
* PARAMETER LIST FOR INSTALLATION EXIT 62

UX62LIST DS  OF
UX62RSQ  DS  F       POINTER TO JOB'S RESQUEUE
UX62VOLR  DS  CL6     REQUESTED VOL SER
UX62LABR  DS  C       REQUESTED LABEL TYPE:
* A=ANSII  B=BYPASS LABEL PROC
   N=NON-LABEL  S=STANDARD LABEL
   X=NON-STANDARD LABEL
UX62RNGR  DS  C       REQUESTED RING STATUS
* R=RING  N=NO RING
UX62MVOL  DS  CL6     MOUNTED VOL SER
UX62MLAB  DS  C       MOUNTED LABEL TYPE
UX62MRNG  DS  C       ACTUAL RING STATUS
UX62TEXT  DS  CL16    ERROR TEXT TO BE MOVED
               TO MESSAGE IAT5310
UX62LEND  EQU  *      END OF PARM LIST
```

The text in UX62TEXT will be printed as part of message IAT5310 if the exit routine returns a return code of 8 or 12. The installation exit can modify this area. On entry this area will contain ‘REJECTED BY EXIT’, which will be printed if not modified by the exit.

**Programming Considerations**

The exit is invoked using ASAVE linkage.

**Refresh Status**

This user exit is refreshable through the command

*MODIFY,X,M=IATxxxx,REFRESH.

**Information Supplied on Exit**

The contents of the registers on exit are:

- **R0-13**: Same as on entry
- **R14**: Return address

The output from this exit routine is a return code in register 15. The valid values are:

- **0**: Use the JES3 decision
- **4**: Accept the mount regardless of JES3 decision
- **8**: Reject the mount regardless of JES3 decision (JES3 will issue message IAT5310 and suspend the job, pending operator action.)
- **12**: Reject the mount regardless of JES3’s decision and put the job on the error queue. Message IAT5310 will be issued to JESMSG only.
- **16**: Dummy exit – JES3’s decision will be used and the installation exit will not be called again until JES3 is restarted.
CSECT Name: IATUX63 (Provide SSI Subsystem Installation String Information)

Exit routine IATUX63 allows an installation the option of providing its own information to applications that request subsystem version information (through SSI code 54), and to override the information passed by JES3. It does this by providing the Installation String Information that contains the information related to the subsystem version.

Environment

This exit runs during JES3 global and local address space initialization after the processor has been IPLed. It also runs on a JES3 local address space during connect processing.

This exit will not be called for JES3 C/I FSS address spaces.

This exit is called in 31-bit addressing mode. The entry point in register 15 is a 31-bit address.

Linkage Conventions

This exit routine is invoked with the ACALL macro using ASAVE linkage. To return from the exit, use the ARETURN macro without the RC= parameter.

Information Supplied on Entry

The contents of the registers on entry are:

- R0: Irrelevant.
- R1: A pointer to the IATUX63 parameter list.
  - On entry, this register points to the IATUX63 parameter list mapped by the mapping macro IATYUX63. Addressability on symbol IATYUX63 can be used.
- R2-10: Irrelevant.
- R11: IATYFCT address.
- R12: IATYTVT address.
- R13: Pointer to IATYINT during an initialization call, unpredictable otherwise.
- R14: Return address.
- R15: Entry point address into IATUX63.

Programming Considerations

The parameter list pointed to in register 1 is mapped in macro IATYUX63. The macro contains the following information at the labels indicated:

- YUX63LNP: The length of the string buffer.
  - The system programmer must code IATUX63 to check that the length of the string that is being passed back to JES3 is large enough for the exit to use (less than or equal to the size in YUX63LNP). If it is not large enough, the exit should bypass the creation of the string.

- YUX63LNR: This field contains the length of the string being passed that is pointed to by field YUX63ADR. IATUX63 sets this field before returning to JES3.
This is a pointer to the address of the string buffer provided by JES3. IATUX63 puts the string in this address (YUX63ADR).

The user exit may return an installation defined string at this address to provide either additional information or to override that information supplied by JES3 to SSI callers, if appropriate. For more information, see z/OS MVS Using the Subsystem Interface.

This bit is set by the exit to let JES3 know that a character string was returned by IATUX63.

Bit that indicates IATUX63 is running on the global.

Bit that indicates IATUX63 is running on a local.

Bit that indicates that a cold start in progress.

Bit that indicates that a warm start in progress.

Bit that indicates that a hot start in progress.

Pointer to the IATYTVT.

Pointer to the IATYINT during an initialization call and it is set to zero during a connect call.

Bit that indicates that a connect is in progress. A call to IATUX63 with this bit on is made only on a local. If you created an installation defined string during initialization and the information in your user string is sensitive to a global restart or a DSI, you can update or rebuild the installation defined string during a connect call.

You should include code to test YUX63CON and return it if the YUX63CON bit is on, rather than rebuild a string that has not changed since initialization.

This user exit is not refreshable through the command *MODIFY,X,M=IATxxxx,REFRESH.

The contents of the registers on exit are:

R0 Irrelevant
R1 Same as on entry
R2-10 Irrelevant
R11 Same as on entry
R12 Same as on entry
R14 Return address
R15 Return code - must be 0

Use ARETURN RC=0 to return to the caller.

To use IATUX63, YUX63LNR must contain the length of the returned string. Bit YUX63FPI must be set to indicate that a string is being returned. During a connect call, set these symbols if you have rebuilt the string at a new address or changed the length, but do not set them if you have updated an existing string in place without changing the length.
CSECT Name: IATUX66 (Determine Transmission Priority for a SNA/NJE Stream)

This exit, called from module IATOSBP, allows you to assign a transmission priority to a SNA/NJE stream. You can either specify a transmission priority for the SNA/NJE stream or allow JES3 to use the job priority (specified either on the CLASS initialization statement or in the job's JCL) as the transmission priority. See z/OS JES3 Initialization and Tuning Reference for additional information about where a job's priority can be set.

Environment

This exit runs in the JES3 global address space.

This exit is called in 31-bit addressing mode. The entry point in register 15 is a 31-bit address.

Linkage Conventions

This exit routine is invoked with the ACALL macro using ASAVE linkage. To return from the exit, use the ARETURN macro with the RC= parameter.

Information Supplied on Entry

The contents of the registers on entry are:

- R0: Irrelevant
- R1: Address of parameter list mapped by IATYUX66
- R2-R10: Irrelevant
- R11: Address of IATYFCT
- R12: Address of IATYTVT
- R13: Irrelevant
- R14: Return address
- R15: Entry point address for IATUX66

Programming Considerations

Register 1 points to a parameter list (defined in IATYUX66) that contains the following information at the labels indicated.

- **YUX66JBN**: The job name or APPC transaction program name assigned to the SNA/NJE transmission stream.
- **YUX66JBI**: The job identifier assigned to the SNA/NJE transmission stream.
- **YUX66USI**: The userid of the TSO user or the userid associated with the transaction program that submitted the job.
- **YUX66SBL**: The security label associated with the SNA/NJE transmission stream.
- **YUX66TPI**: The transaction program identifier of the data sets in the SNA/NJE transmission stream (for APPC SYSOUT only).
- **YUX66JBD**: The requested destination of the SNA/NJE transmission stream.
- **YUX66JSI**: The job id from the JSAB (for SYSOUT created with spinoff job names only).
- **YUX66JBP**: The job priority of the SNA/NJE transmission stream.
- **YUX66DSP**: The highest priority assigned to any of the data sets in the SNA/NJE transmission stream.
YUX66DSN  The number of data sets in the SNA/NJE transmission stream.
YUX66LNC  The total line count of the data sets in the SNA/NJE transmission stream.
YUX66PGC  The total page count of the data sets in the SNA/NJE transmission stream.
YUX66RCC  The total record count of the data sets in the SNA/NJE transmission stream.
YUX66BYC  The transmission stream byte count. This value is the number of spool buffers used. Multiply this value by the contents of field SIZEBUF to obtain the actual byte count.
YUX66FL1  Indicates the type of information the SNA/NJE transmission stream contains.
          YUX66SPN  The transmission stream contains a spinoff data set.
          YUX66JBS  The transmission stream is a job stream.
          YUX66SYS  The transmission stream is a SYSOUT stream.
          YUX66APC  The transmission stream is APPC generated.
          YUX66JSB  The transmission stream contains SYSOUT created with spinoff job names.
YUX66XMP  The new transmission priority of the SNA/NJE transmission stream set by the user. On return to IATOSBP, this value will be used to set the transmission priority of the transmission stream.

Refresh Status
This user exit is refreshable through the command
*MODIFY,X,M=IATxxxx,REFRESH.

Information Supplied on Exit
The contents of the registers on exit are:
R0-1     Irrelevant
R2-13    Same as on entry
R14      Return address
R15      Return code

0        The installation has specified a new transmission priority (in field YUX66XMP) to be assigned to this SNA/NJE transmission stream.
4        The installation has decided to let the transmission priority of the SNA/NJE transmission stream default to the job priority.
8        Dummy installation exit; this exit will not be invoked again.
CSECT Name: IATUX67 (Determine Action When Remote Data Set Is Rejected by RACF)

This exit, called from modules IATNTSF and IATNTRS, allows you to override RACF's security decision when RACF determined that the data set cannot be processed by the home node. You can direct the SYSOUT data set to be purged (default), held for TSO, or allow JES3 to process the job normally (for example, when a data set needs to print).

When the installation decides to hold the data set for TSO or to process it normally, the exit can provide a new security token. Also, the following JESSPOOL class entity name qualifiers can be replaced with new values:

- **userid** second qualifier – YUX67USR
- **dsname** last qualifier – YUX67DNM

Environment

This exit runs in the JES3 global address space.

This exit is called in 31-bit addressing mode. The entry point in register 15 is a 31-bit address.

Linkage Conventions

This exit routine is invoked with the ACALL macro using ASAVE linkage. To return from the exit, use the ARETURN macro without the RC= parameter.

Information Supplied on Entry

The contents of the registers on entry are:

- **R0** Irrelevant
- **R1** Address of parameter list mapped by IATYUX67
- **R2-R10** Irrelevant
- **R11** Address of IATYFCT
- **R12** Address of IATYTVT
- **R13** Address of NJESF/NJEROUT data CSECT (modules IATNTFD mapped by macro IATYNFD and IATNTRD mapped by macro IATYNRS)

Notes:

1. If the caller is IATNTSF, then NJESF is present.
2. If the caller is IATNTRS, then NJEROUT is present.

- **R14** Return address
- **R15** Entry point address for IATUX67

Programming Considerations

Register 1 points to a parameter list (defined in IATYUX67) that contains the following information at the labels indicated.

- **YUX67VSN** The parameter list version.
- **YUX67UTK** The address of the current user security token (UTOKEN).
- **YUX67RTK** The address of the current resource security token (RTOKEN).
- **YUX67NOD** The current node name of the JESSPOOL entity name.
- **YUX67USR** The current userid of the JESSPOOL entity name.
YUX67JNM The job name of the JESSPOOL entity name.
YUX67JID The jobid of the JESSPOOL entity name.
YUX67DNO The dsnumber of the JESSPOOL entity name.
YUX67DNM The current dsname of the JESSPOOL entity name.
YUX67NDH The address of the NJE data set header.

Refresh Status
This user exit is refreshable through the command
*MODIFY,X,M=IATxxxx,REFRESH.

Information Supplied on Exit
The contents of the registers on exit are:

<table>
<thead>
<tr>
<th>Register</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R0-1</td>
<td>Irrelevant</td>
</tr>
<tr>
<td>R2-R13</td>
<td>Same as entry</td>
</tr>
<tr>
<td>R14</td>
<td>Return address</td>
</tr>
<tr>
<td>R15</td>
<td>Return code</td>
</tr>
</tbody>
</table>

- 0: The data set is to be purged.
- 4: The data set is to be held for TSO (overriding RACF).
- 8: The data set is to be processed normally (overriding RACF).
- 12: Reserved (treated like RC16).
- 16: Dummy installation exit; the exit will not be called again until the system is restarted. The data set is purged.

CSECT Name: IATUX68 (Modify Local NJE Job Trailers)
This installation exit allows you to view and modify a job trailers (mapped by IATYNJT) for a network stream containing SYSOUT after JES3 or MVS/BDT receives the network stream. The trailer is used to update job information pertaining to the SYSOUT.

This installation exit can be used to process BSC/NJE jobs and SNA/NJE jobs.

Environment
This installation exit routine runs in the JES3 global address space.

This exit is called in 31-bit addressing mode. The entry point in register 15 is a 31-bit address.

Linkage Conventions
This exit is called from module IATNTJS using ASAVE linkage with the ACALL and ARETURN macros. To return from the exit, use the ARETURN macro without the RC= parameter.

Programming Considerations
None.
Installation Exit 68

Information Supplied on Entry

The contents of the registers on entry are:

- **R0-2**: Irrelevant
- **R3**: Job trailer (NJTSTART)
- **R4-9**: Irrelevant
- **R10**: Module base
- **R11**: Pointer to the IATYFCT
- **R12**: Pointer to the IATYTVT
- **R13**: Network receiver data area (IATYNRD)
- **R14-15**: Irrelevant

Refresh Status

This user exit is refreshable through the command

```
*MODIFY,X,M=IATxxxx,REFRESH.
```

Information Supplied on Exit

The contents of the registers on exit should be:

- **R0-14**: Irrelevant
- **R15**: Return code:
  - 0: Job trailer processing occurred and is complete.
  - 4: Dummy exit; the exit will not be called again until JES3 processes the next NJE job.

CSECT NAME: IATUX69 (Determine If a Message is to be Sent to the JES3 Global Address Space)

This installation exit receives control from the JES3 Write to Operator (WTO) subsystem interface routine (module IATSIWO). This exit allows you to examine a message and determine if the message needs to be sent to the JES3 global address space for additional processing in exit 70. The message is sent to the JES3 global if the release level of the JES3 Global is below HJS7705 or if the release level of the local processor is below HJS7705.

The following is a summary of the types of WTOs/WTORs that are presented to exit 69 and any special considerations that exist for the type of message. Flags in the exit parameter list indicate the type of request being passed to exit 69.

- **Multi-Line WTO**: The first line of a multi-line WTO (the major line) is passed to exit 69. Subsequent minor lines are NOT passed to exit 69. Using the major line, the exit determines whether the message should be sent to the global.
  
  If the exit determines that the message should be sent to the global, the major line and all minor lines are presented to exit 70 on the global. A separate call to exit 70 is made for each minor line.

- **Operator Commands**: The text of an operator command (command echo) is presented to exit 69.

- **Write to Operator With Reply (WTOR)**: The text of a WTOR is presented to exit 69.

- **Responses to WTORs**: The full text of a WTOR response is presented to exit 69 as message IEE600I. For example:

  ```
  IEE600I REPLY TO 13 IS;GO
  ```

JES3 defines this exit to the MVS dynamic exit facility with the name IAT_EXIT69. By default, JES3 does not define any exit routines to this exit. You can use the
EXIT statement of the PROGxx parmlib member, the SETPROG EXIT operator command, or the CSVDYNEX macro to control this exit and its exit routines. JES3 allows multiple exit routines to exist for this exit. JES3 provides module IATUX69 which is a skeleton of an exit routine for exit 69.

Installing an Exit Routine

Use the following steps to install and define an exit routine to IAT_EXIT69.
- Linked the exit routine into a system library
- On the EXIT statement of the PROGxx member of SYS1.PARMLIB, or on the SETPROG EXIT operator command, specify:
  - IAT_EXIT69 as the exit name.
  - The exit routine's load module name as the module name.

Environment

This exit routine runs in the address space of the issuer of the WTO/WTOR macro. This exit runs on the global and local processors.

This exit is called in 31-bit addressing mode.

The exit routines run in Supervisor State and Key 0.

The exit routines are loaded into CSA and must be reentrant.

Information Supplied on Entry

The contents of the registers on entry are:

<table>
<thead>
<tr>
<th>Register</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R0</td>
<td>Irrelevant</td>
</tr>
<tr>
<td>R1</td>
<td>Address of the Exit Parameter List (IATYUX69)</td>
</tr>
<tr>
<td>R2-R12</td>
<td>Irrelevant</td>
</tr>
<tr>
<td>R13</td>
<td>Register Save Area</td>
</tr>
<tr>
<td>R14</td>
<td>Return Address</td>
</tr>
<tr>
<td>R15</td>
<td>Exit routine entry point address</td>
</tr>
</tbody>
</table>

The following is the contents of the parameter list.

**YUX69ID**
- The parameter list identifier 'YUX69'.

**YUX69VER**
- The parameter list version indicator

**YUX69TXP**
- The pointer to the message text

**YUX69TXL**
- The length of the message text

**YUX69SVT**
- The address of SSVT

**YUX69SEQ**
- WTO sequence number (DOM/Connect ID)
Installation Exit 69

YUX69WRK
A 20 byte work area for use by the exit routine(s). JES3 initializes this area to zero before calling the exit routine(s).

If multiple exit routines exist for this exit, this work area can be used to pass information from one routine to another. If the information can fit within the 20 byte area, then the information may be placed directly in the work area. Otherwise, the exit routines can obtain storage and store the address of the obtained storage in this work area. The exit routines are responsible for freeing any storage obtained.

YUX69JBD
The job identifier of the WTO/WTOR issuer

YUX69JBN
The job name of the WTO/WTOR issuer

YUX69SYS
JES3 processor/system name

YUX69KEY
The WTO retrieval key

YUX69TYP
The message type flag which may have following values:

YUX69SIN
Indicates a single line request

YUX69MAJ
Indicates the major line of a multi-line message

YUX69TOR
Indicates the message is a WTOR

YUX69CMD
Indicates the message is an operator command

YUX69REP
Indicates the message is a WTOR reply

NOTE: The flags in YUX69TYP are mutually exclusive. Only one of these flags is set for each call to exit 69.

YUX69R00
An equate for return code 0

YUX69R04
An equate for return code 4

Programming Considerations
The exit routine should not issue a WTO or WTOR macro.

This exit is only called on the system that the message originates on (i.e. it is not called on other systems on which a message is presented).

The exit routines must be reentrant.

Depending on the sequence and timing of WTO processing, the minor lines for a multi-line message may not be passed in sequence to exit 69. Exit 69 may be called for unrelated messages before all lines of a multi-line message have been passed.
This exit is called only when JES3 is active on the processor issuing the message.

**Refresh Status**

This user exit is not refreshable through the command
*MODIFY,X,M=IATxxxx,REFRESH.

**Information Supplied on Exit**

<table>
<thead>
<tr>
<th>R0-R13</th>
<th>Same as on entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>R14</td>
<td>Return Address</td>
</tr>
<tr>
<td>R15</td>
<td>Return Code</td>
</tr>
</tbody>
</table>

The exit routines must set one of the following return codes when returning:

- **RC=0** Indicates that no additional processing of the message is needed.
- **RC=4** Indicates that IATSIWO should send the message to the JES3 global address space for further processing. On the global, installation exit 70 is called to process the message.

If multiple exit routines exist for this exit, JES3 uses the highest return code set by all of the exit routines in determining whether the message should be sent to the global.

If an undefined return code is received, JES3 will perform default processing (RC=0).

**CSECT NAME: IATUX70 (Perform Additional Message Processing)**

This installation exit allows you to perform additional processing as a result of a certain message being issued. This exit is called in the global JES3 address space when installation exit 69 determines that a message requires further processing and was not written to the JESMSGGLG data set on the local processor. The message is written to the JESMSGLG data set on the local when the JES3 global and the local processor are both at a minimum HJS7705 release level.

JES3 defines this exit to the MVS dynamic exit facility with the name IAT_EXIT70. By default, JES3 does not define any exit routines to this exit. You can use the EXIT statement of the PROGxx parmlib member, the SETPROG EXIT operator command, or the CSVDYNEX macro to control this exit and its exit routines. JES3 allows multiple exit routines to exist for this exit. JES3 provides module IATUX70 which is a skeleton of an exit routine for exit 70.

**Installing an Exit Routine**

Use the following steps to install and define an exit routine to IAT_EXIT70.

- Linkedit the exit routine into a system library
- On the EXIT statement of the PROGxx member of SYS1.PARMLIB, or on the SETPROG EXIT operator command, specify:
  - IAT_EXIT70 as the exit name.
  - The exit routine’s load module name as the module name.
Installation Exit 70

Environment

This exit routine runs in the JES3 global address space under the CONSERV FCT.

This exit is called in 31-bit addressing mode.

Information Supplied on Entry

The contents of the registers on entry are:

- **R0**: Irrelevant
- **R1**: Address of the Exit Parameter List (IATYUX70)
- **R2-R12**: Irrelevant
- **R13**: Address of a Register Save Area
- **R14**: Return Address
- **R15**: Exit Routine Entry Address

The following is the contents of the parameter list.

**YUX70ID**
- The parameter list identifier ‘YUX70’.

**YUX70VER**
- The parameter list version indicator

**YUX70PTX**
- The pointer to primary message text. If the current message is a single line WTO or the major line of a multi-line WTO, this field contain the address of the message text and YUX70STX is zero.
- If the current message is a minor line of a multi-line WTO, this field contains the address of the original major line and YUX70STX contains the address of the text of the current minor line.

**YUX70STX**
- The pointer to the secondary text. When called for a minor line of a multi-line message, this field contains the address of the text of the current minor line and YUX70PTX contains the address of the original major line.

**YUX70PTL**
- The length of the primary message text

**YUX70STL**
- The length of the secondary message text

**YUX70SEQ**
- The WTO sequence number (DOM/Connect ID)

**YUX70TVT**
- The address of the JES3 TVT

**YUX70FCT**
- The address of the current FCT

**YUX70WRK**
- A 20 byte work area for use by the exit routine(s). JES3 initializes this area to zero before calling the exit routine(s).
- If multiple exit routines exist for this exit, this work area can be used to pass information from one routine to another. If the information can fit within the 20 byte area, then the information may be placed directly in the work area.
Otherwise, the exit routines can obtain storage and store the address of the obtained storage in this work area. The exit routines are responsible for freeing any storage obtained.

**YUX70JBD**
- The job identifier of the WTO/WTOR issuer

**YUX70JBN**
- the job name of the WTO/WTOR issuer

**YUX70SYS**
- The JES3 processor/system name from where the message originated

**YUX70KEY**
- The WTO retrieval key

**YUX70TYP**
- The message type flag which may contain one of the following:
  - **YUX70SIN**
    - Indicates a single line request
  - **YUX70MAJ**
    - Indicates the major line of a multi-line message
  - **YUX70TOR**
    - Indicates the message is a WTOR
  - **YUX70CMD**
    - Indicates the message is an operator command
  - **YUX70REP**
    - Indicates the message is a WTOR reply
  - **YUX70MIN**
    - Indicates the minor line of a multi-line message

**NOTE:** The flags in YUX70TYP are mutually exclusive. Only one of these flags is set for each call to exit 70.

**YUX70MLW**
- Indicates a multi-line message flag

**YUX70CON**
- Indicates a control line

**YUX70LBL**
- Indicates a label line

**YUX70DAT**
- Indicates a data line

**YUX70END**
- Indicates an end line

**Programming Considerations**

The exit routines are NOT invoked with ASAVE linkage.

Each line of a multi-line message is passed on separate calls to exit 70. For each minor line of a multi-line message the text of the original major line is also passed. Depending on the sequence and timing of WTO processing, the minor lines for a
multi-line message may not be passed in sequence to exit 70 (i.e. exit 70 may be called for unrelated messages before all lines of a multi-line message have been passed).

Refresh Status

This user exit is not refreshable through the command
*MODIFY,X,M=IATxxxx,REFRESH.

Information Supplied on Exit

<table>
<thead>
<tr>
<th>Register</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R0-R13</td>
<td>Same as on entry</td>
</tr>
<tr>
<td>R14</td>
<td>Return Address</td>
</tr>
<tr>
<td>R15</td>
<td>Irrelevant</td>
</tr>
</tbody>
</table>

CSECT Name: IATUX71 (Modify a Tape Request Setup Message)

This exit is a main device scheduler (MDS) routine that can be used to modify one of the following JES3 tape request setup messages: IAT5110, IAT5210, IAT5410, and IAT5420. If you choose to modify one of these messages, you can return either:

- A 1- to 6-character text string to replace the volume serial (volser) in the message, or
- A 1- to 13-character text string to be appended to the message if enough space is available. (If there is not enough space, JES3 appends only as many characters as can fit within the maximum message size.)

When the exit is invoked for the IAT5210 (mount) message, you can also return a 1- to 6-character text string to replace the volser in the message display on the tape drive (for MSGDISP).

Environment

This exit runs in the JES3 global address space.

This exit is called in 31-bit addressing mode. The entry point in register 15 is a 31-bit address.

Linkage Conventions

This exit routine is invoked with the ACALL macro using ASAVE linkage. To return from the exit, use the ARETURN macro without the RC= parameter.

Information Supplied on Entry

The input to IATUX71 is a 10-word parameter area.

The contents of the registers upon entry are:

<table>
<thead>
<tr>
<th>Register</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R0</td>
<td>Irrelevant</td>
</tr>
<tr>
<td>R1</td>
<td>Parameter list address</td>
</tr>
<tr>
<td>R2-10</td>
<td>Irrelevant</td>
</tr>
<tr>
<td>R11</td>
<td>IATYFCT address</td>
</tr>
<tr>
<td>R12</td>
<td>IATYTVT address</td>
</tr>
<tr>
<td>R13</td>
<td>IATYMDS address</td>
</tr>
</tbody>
</table>
Programming Considerations

The exit is invoked using ASAVE linkage.

The parameter list pointed to by register 1 is mapped by macro IATYMDS and defined as follows:

<table>
<thead>
<tr>
<th>Label</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UX71LIST</td>
<td>DSECT name</td>
</tr>
<tr>
<td>UX71MESG</td>
<td>Address of the message area. Byte 1 of the message area contains the message length, and bytes 2-120 contain the message text.</td>
</tr>
<tr>
<td>UX71JNAM</td>
<td>Address of the job name</td>
</tr>
<tr>
<td>UX71VOL</td>
<td>Address of the volser in the message area</td>
</tr>
<tr>
<td>UX71DSN</td>
<td>Address of the data set length followed by the data set or 0 to indicate no data</td>
</tr>
<tr>
<td>UX71DEV</td>
<td>Address of the EBCDIC device number, or 0 if none</td>
</tr>
<tr>
<td>UX71MDSP</td>
<td>MSGDISP insertion text - output area</td>
</tr>
<tr>
<td>UX71INST</td>
<td>message insertion text - output area</td>
</tr>
<tr>
<td>UX71APND</td>
<td>message appendage - output area</td>
</tr>
</tbody>
</table>

Notes:
1. UX71MDSP is initialized to blanks, before calling the exit. If you modify this field, the new value is passed to MSGDISP to be used in the tape message display.
2. Do not use this exit to modify the message text itself. JES3 modifies the message text based on the return code and the value in UX71INST or UX71APND.

Refresh Status

This user exit is refreshable through the command *MODIFY,X,M=IATxxxx,REFRESH.

Information Supplied on Exit

The contents of the registers on exit are:

<table>
<thead>
<tr>
<th>Register</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R0-R1</td>
<td>Irrelevant</td>
</tr>
<tr>
<td>R2-13</td>
<td>Same as on entry</td>
</tr>
<tr>
<td>R14</td>
<td>Return address</td>
</tr>
</tbody>
</table>

The output from this exit routine is a return code in register 15. Valid values are:

0  Do not modify the message. The tape message display is replaced by the value, if any, specified in field UX71MDSP.
4  Replace the volume serial in the message with the value returned in field UX71INST. The tape message display is replaced by the value, if any, specified in field UX71MDSP.
8  Append the value returned in field UX71APND to the message.
Installation Exit 71

The tape message display is replaced by the value, if any, specified in field UX71MDSP.

Note: If the append causes the message to exceed the maximum message size (119 characters), JES3 only appends as many characters as fit.

16 Dummy exit - the message is not modified and the exit is not called again until JES3 is restarted.

CSECT Name: IATUX72 (Examine/Modify a Temporary OSE or an OSE Moved to Writer Queue)

This exit provides you with the contents of the output service element (OSE) constructed for a SYSOUT data set. You have the opportunity of examining and, if required, changing the information in the OSE.

When called from the output service resident driver (module IATOSDO), the OSE is a temporary OSE constructed for a SYSOUT data set. IATUX72 is called before IATUX19 and may be used instead of IATUX19 to process the OSE before it is written to spool.

When called from the output service modify implementation (module IATMOOI), sysout API (SAPI) processor (module IATOSO), processor sysout (PSO) scheduler (module IATOSPC), or SSI 70 driver (module IATGR70), the OSE is an existing OSE that is being moved from the hold queue to the writer queue. You have the opportunity of examining and, if required, changing the information before the OSE is rewritten to spool.

When a job enters output service, or when a sysout data set is spun off by an active job, output service will call IATUX72 to process the OSE. IATUX19 will also be called if IATUX72 issues a return code 8 or if IATUX72 is a dummy exit.

When sysout is modified so that it is moved from the hold queue to the writer, BDT queue, or TSO queue, the OSE might be rebuilt. In this case output service will call IATUX72 to process the OSE. IATUX19 will also be called if IATUX72 issues a return code 8 or if IATUX72 is a dummy exit.

If JES3 determines that it is not necessary to rebuild the OSE, then output service will not be invoked. In this case, the active function (operator *MODIFY, PSO, SAPI, or SSI 70) will call IATUX72 to process the OSE and IATUX19 will not be called.

Examples of modifications that do not require the OSE to be rebuilt include:
- Output that was built using an output JCL statement or dynamic output descriptor
- Output that had earlier been moved from the writer queue to the hold queue
- Output that was not built with a /*FORMAT statement and whose sysout class is not being modified

Environment

This exit runs in the JES3 global address space under the MODOSFCT, PSODSP, SAPIDSP, SJFFCT, OR OUTSERV FCT.

This exit is called in 31-bit addressing mode. The entry point in register 15 is a 31-bit address.
Linkage Conventions

This exit routine is invoked with the ACALL macro using ASAVE linkage. To return from the exit, use the ARETURN.

Information Supplied on Entry

The contents of the registers upon entry are:

- **R0**: Irrelevant
- **R1**: Parameter list address
- **R2-10**: Irrelevant
- **R11**: Address of the active FCT
- **R12**: IATYTVT address
- **R13**: Irrelevant
- **R14**: Return address
- **R15**: Entry point address into IATUX72.

Programming Considerations

The exit is invoked using ASAVE linkage.

The parameter list pointed to by register 1 is mapped by the IATYUX72 mapping macro and defined as follows:

<table>
<thead>
<tr>
<th>Label</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>YUX72ID</td>
<td>The parameter list identifier ‘YUX72’</td>
</tr>
<tr>
<td>YUX72VER</td>
<td>The parameter list version number</td>
</tr>
<tr>
<td>YUX72CUR</td>
<td>An equated symbol indicating the current version number</td>
</tr>
<tr>
<td>YUX72RQA</td>
<td>A pointer to the job’s RESQUEUE (IATYRSQ)</td>
</tr>
<tr>
<td>YUX72JDS</td>
<td>A pointer to the SYSOUT data set’s JDS (IATYJDS) entry (JDSENTRY DSECT). This field may be zero if the calling FCT was not using the JDS entry to process the modification.</td>
</tr>
<tr>
<td>YUX72OSV</td>
<td>A pointer to the OSE (IATYOSE) variable section (OSEENTRY DSECT).</td>
</tr>
<tr>
<td>YUX72OSD</td>
<td>A pointer to the OSE (IATYOSE) data set section (starts at OSEJDSPT within the OSEENTRY DSECT). If this field is zero, more than one OSE data set section is being modified.</td>
</tr>
<tr>
<td>YUX72MOS</td>
<td>If the YUX72MOD bit is set then this is the address of the caller’s IATMOOS/IATMOOI data area (IATYMOOS). Otherwise, this field will contain zero.</td>
</tr>
<tr>
<td>YUX72WSP</td>
<td>If the YUX72PSO bit is set then this is the address of the caller’s Writer Selection Parameter Area (IATYWSP). Otherwise this field will contain zero.</td>
</tr>
<tr>
<td>YUX72SDW</td>
<td>If the YUX72SAP bit is set then this is the address of the caller’s SAPI DSP Work Area (IATYSDW). Otherwise this field will contain zero.</td>
</tr>
</tbody>
</table>
Installation Exit 72

YUX72OSA  If the YUX72OUS bit is set then this is the address of the caller’s Output Service Data Area (IATYOSA). Otherwise this field will contain zero.

YUX72SMW  If the YUX72SJF bit is set then this is the address of the caller’s SSI 70 Modify/Merge work area (IATYSMW). Otherwise this field will contain zero.

YUX72FL1  A flag byte which may contain one of the following:

- **YUX72MOD**  The active FCT is MODOSFCT (Operator *MODIFY).
- **YUX72PSO**  The active FCT is PSODSP (Process SYSOUT).
- **YUX72SAP**  The active FCT is SAPIDSP (SYSOUT Application Programming Interface, or SAPI).
- **YUX72OUS**  The active FCT is OUTSERV (Output Service Driver)
- **YUX72SJF**  The active FCT is SJFFCT (SSI 70)

YUX72OSV points to the OSE variable section. (The OSE defines all the characteristics of the SYSOUT data set, for example, scheduling and setup information.) Because fields changed in the OSE may not be propagated into the related control blocks (such as the JDS), it is the user’s responsibility to determine the effects of changing any fields in the OSE, and to determine if the required results will be achieved.

In order to avoid a lockout with the MAIN FCT, the job’s JDAB (IATYJDA) should not be read while the active FCT holds the job’s JDS.

The JDS entry has not been updated on spool. Do not release the JDS with the JDSREL macro without issuing JDSPUT first.

Do not perform I/O for the spool OSE chain for this job. Only the OSE being modified by the caller should be modified by the exit.

Do not use the IATXOSUP macro in this installation exit. Upon entry, the RQOSESUP bit is on. This bit should remain on for the duration of the exit.

When an OSE is initially constructed, or when it is rebuilt, either IATUX19 or IATUX72 can be used to examine or modify it. When an OSE is moved to the writer queue without being rebuilt, only IATUX72 can be used to examine or modify it.

**Notes:**

1. Installations that do not need to process an existing OSE that is moved to the writer queue can use the IBM-supplied IATUX72, which is a dummy exit. IATUX19 will then be used to process OSEs when they are built or rebuilt.

2. Installations that want to use an existing IATUX19 to process OSEs, but also need to process output that is moved to the writer queue, can code IATUX72 so that it uses return code 8 when called under the OUTSERV FCT. This directs the output service driver to call IATUX19.

3. Installations that want to use a single exit to process OSEs both when they are built/rebuilt and when they are moved to the writer queue can use IATUX72 for that purpose. IATUX72 should use return code 0 or 4 so that IATUX19 is not called by the output service driver.
Refresh Status

This user exit is refreshable through the command
*MODIFY,X,M=IATxxxx,REFRESH.

Information Supplied on Exit

The contents of the registers on exit are:

- **R0-R13**: Irrelevant
- **R14**: Return address
- **R15**: Return code

Use the return code in register 15 to specify how JES3 is to proceed as follows:

- **0**: The output should be deleted.
- **4**: Continue normal processing.
- **8**: Continue normal processing. If the exit was called by the output service driver, allow IATUX19 to process the output.
- **12**: Dummy exit; the exit will not be called again until the JES3 system is restarted.
Installation Exit 72
Chapter 2. JES3 Macro Instructions

Introduction

Most JES3 functions are called by macro instructions (also called macros) that invoke service routines. The assembler program processes these macros using macro definitions placed in the macro library when JES3 is generated.

The processing of the macros by the assembler program results in a macro expansion, generally consisting of executable instructions and data as assembler language statements. The data fields are the parameters to be passed to the JES3 function. The executable instructions generally consist of a branch around the data fields, instructions to load registers, and a branch instruction to give control to the proper routine. The exact macro expansion appears as a part of the assembler listing.

Some executable macros do not give control to another routine. These macros generate all necessary executable code “inline,” that is, in the macro expansion.

JES3 mapping macros are described at the end of this topic, following the descriptions of the JES3 executable macros. Not all JES3 mapping macros are included in this manual; they are described in JES3 Data Areas Volume 1 to Volume 4.

Coding Macro Instructions

JES3 macros are written in the assembler language and, as such, are subject to the rules contained in OS/VS, DOS/VSE and VM/370 Assembler Language. JES3 macros, like all assembler language instructions, are written in the following form:

<table>
<thead>
<tr>
<th>Name</th>
<th>Operation</th>
<th>Parameters</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>symbol or blank</td>
<td>Macro name</td>
<td>None, one, or more parameters separated by commas</td>
<td>Any appropriate comment (a blank character must separate the comment from the parameters.</td>
</tr>
</tbody>
</table>

The label entry can be omitted unless specifically required by the macro, in which case a proper label must be assigned.

The macro-name entry is required and must be the name of the macro in capital letters.

In the macro format used in this manual, the parameters specify services or options to be used. The macro description shows the parameters according to the general rules given in the following sections.

How to read syntax diagrams

This section describes how to read syntax diagrams. It defines syntax diagram symbols, items that may be contained within the diagrams (keywords, variables, delimiters, operators, fragment references, operands) and provides syntax examples that contain these items.
Syntax diagrams pictorially display the order and parts (options and arguments) that comprise a command statement. They are read from left to right and from top to bottom, following the main path of the horizontal line.

**Symbols**
The following symbols may be displayed in syntax diagrams:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>/SM590000/SM590000───</td>
<td>Indicates the beginning of the syntax diagram.</td>
</tr>
<tr>
<td>───/SM590000</td>
<td>Indicates that the syntax diagram is continued to the next line.</td>
</tr>
<tr>
<td>/SM590000───</td>
<td>Indicates that the syntax is continued from the previous line.</td>
</tr>
<tr>
<td>───/SM590000/SM630000</td>
<td>Indicates the end of the syntax diagram.</td>
</tr>
</tbody>
</table>

**Syntax items**
Syntax diagrams contain many different items. Syntax items include:

- **Keywords** - a command name or any other literal information.
- **Variables** - variables are italicized, appear in lowercase and represent the name of values you can supply.
- **Delimiters** - delimiters indicate the start or end of keywords, variables, or operators. For example, a left parenthesis is a delimiter.
- **Operators** - operators include add (+), subtract (-), multiply (*), divide (/), equal (=), and other mathematical operations that may need to be performed.
- **Fragment references** - a part of a syntax diagram, separated from the diagram to show greater detail.
- **Separators** - a separator separates keywords, variables or operators. For example, a comma (,) is a separator.

Keywords, variables, and operators may be displayed as required, optional, or default. Fragments, separators, and delimiters may be displayed as required or optional.

<table>
<thead>
<tr>
<th>Item type</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Required</td>
<td>Required items are displayed on the main path of the horizontal line.</td>
</tr>
<tr>
<td>Optional</td>
<td>Optional items are displayed below the main path of the horizontal line.</td>
</tr>
<tr>
<td>Default</td>
<td>Default items are displayed above the main path of the horizontal line.</td>
</tr>
</tbody>
</table>

**Syntax examples**
The following table provides syntax examples.

<table>
<thead>
<tr>
<th>Item</th>
<th>Syntax example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Required item.</td>
<td>Required items appear on the main path of the horizontal line. You must specify these items.</td>
</tr>
</tbody>
</table>
### Table 15. Syntax examples (continued)

<table>
<thead>
<tr>
<th>Item</th>
<th>Syntax example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Required choice.</td>
<td>A required choice (two or more items) appears in a vertical stack on the main path of the horizontal line. You must choose one of the items in the stack.</td>
</tr>
<tr>
<td></td>
<td><img src="image1" alt="Syntax diagram" /></td>
</tr>
<tr>
<td>Optional item.</td>
<td>Optional items appear below the main path of the horizontal line.</td>
</tr>
<tr>
<td></td>
<td><img src="image2" alt="Syntax diagram" /></td>
</tr>
<tr>
<td>Optional choice.</td>
<td>A optional choice (two or more items) appear in a vertical stack below the main path of the horizontal line. You may choose one of the items in the stack.</td>
</tr>
<tr>
<td></td>
<td><img src="image3" alt="Syntax diagram" /></td>
</tr>
<tr>
<td>Default.</td>
<td>Default items appear above the main path of the horizontal line. The remaining items (required or optional) appear on (required) or below (optional) the main path of the horizontal line. The following example displays a default with optional items.</td>
</tr>
<tr>
<td></td>
<td><img src="image4" alt="Syntax diagram" /></td>
</tr>
<tr>
<td>Variable.</td>
<td>Variables appear in lowercase italics. They represent names or values.</td>
</tr>
<tr>
<td></td>
<td><img src="image5" alt="Syntax diagram" /></td>
</tr>
<tr>
<td>Repeatable item.</td>
<td>An arrow returning to the left above the main path of the horizontal line indicates an item that can be repeated.</td>
</tr>
<tr>
<td></td>
<td><img src="image6" alt="Syntax diagram" /></td>
</tr>
<tr>
<td>Fragment.</td>
<td>The ─ fragment ─ symbol indicates that a labelled group is described below the main syntax diagram. Syntax is occasionally broken into fragments if the inclusion of the fragment would overly complicate the main syntax diagram.</td>
</tr>
<tr>
<td></td>
<td><img src="image7" alt="Syntax diagram" /></td>
</tr>
</tbody>
</table>

### Meanings of Parameter Variables

The following describes the meaning of each notation used to show how a parameter can be coded.

**hexexp**

When this notation is shown, it represents a *hexadecimal number* in the range 00 through FF. The number must be specified using the format X'nn'.

**address**

When this notation is shown, the parameter can be specified as either of the following:
• Any address that can be written as a valid assembler-language A-type address constant
• Any of the general registers 2 through 9

All registers as parameters must be coded in parentheses. For example, if register 3 is coded, it is coded as (3) or (R3). Unless otherwise specified in the description of a macro, when a register is used as a parameter, the assembler program assumes the register contains the address.

**number**
When this notation is shown, the parameter can be specified as either of the following:
• Any decimal digit up to the maximum allowed for the specific parameter being described
• Any of the general registers 2 through 9

All registers as parameters must be coded in parentheses. For example, if register 3 is coded, it is coded as (3) or (R3). When a register is used as a parameter, the assembler program assumes the register contains the required value, unless otherwise specified in the description of the macro.

(reg)
When this notation is shown, a general register must be coded. The assembler program assumes that the general register already contains the address or value, unless otherwise specified in the description of the macro.

(x) or (Rx)
When this notation is shown, the assembler program assumes that general register x already contains the address or value, unless otherwise specified in the description of the macro.

**Rules for Continuation Lines**
The parameter field of a macro can be continued on one or more lines as follows:
1. Enter a continuation character (not blank, and not part of the parameter coding) in column 72 of the line.
2. Continue the parameter field on the next line, starting in column 16. All columns to the left of column 16 must be blank.

The parameter field being continued can be coded in one of two ways:
1. The parameter field can be coded through column 71, with no blanks, and continued in column 16 of the next line.
2. The parameter field can be truncated by a comma, where a comma normally falls, with at least one blank before column 71. It can then be continued in column 16 of the next line.

An example of each method is shown in the following illustration:

<table>
<thead>
<tr>
<th>Label</th>
<th>Macro Name</th>
<th>Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME1</td>
<td>OP1</td>
<td>PARM1, PARM2, PARM3, PARM4, PARM5, PARM6, PARM7, PARM8, PARM9</td>
</tr>
<tr>
<td>NAME2</td>
<td>OP2</td>
<td>PARM1, PARM2, PARM3, PARM4, X, X</td>
</tr>
</tbody>
</table>
Rules for Register Usage

Many macro expansions include instructions that use a base register previously defined by a USING statement. The USING statement must establish addressability so that the macro expansion can include a branch around the inline parameter list, if present, and refer to data fields and addresses specified in the macro parameters.

When a parameter is specified as a register, the program that uses the macro must have inserted the value or address to be used into the register as follows:

- If the register is to contain a value, it must be placed in the low-order part of the register unless the macro description states otherwise. Any unused bits in the register must be set to zero.
- If the register is to contain an address, all four bytes of the register must be used for the address.
- If the register is to contain an address for an item located below 16 megabytes in virtual storage, the address must be placed in the low-order 3 bytes of the register, and the high-order byte of the register must be set to zero.

Restrictions and Conventions

Registers 0, 1, 2, 14, and 15 are used by most macro expansions and must not be used to pass parameters unless specifically noted in the descriptions of the macro. Registers 11 and 12 must be set up with the addresses of the FCT and TVT, respectively, before you issue a JES3 macro.

Several macros are provided to define the fields of JES3 control blocks. Each routine that uses such JES3 control blocks should use the macros for field definition to ensure consistency in the labels used and to simplify changes in the size or format of the control blocks.

Several of the control blocks consist of a fixed area plus a variable number of entries. Use the following addressing conventions for maximum flexibility in control block expansion with as little reassembly as possible.

To access the fixed area:

- Load a register with the address of the control block.
- Issue a USING statement on this register for the fixed area DSECT name.

To access the first entry in the variable area:

- Load a second register with the address of the fixed area. (You can omit this step if you make no further reference to the fixed area).
- Add to this register the halfword length of the fixed area (contained in the fixed area).
- Issue a USING statement on this register for the variable area DSECT name.
- To access succeeding entries in the variable area, add to the variable area base register the halfword length of the current entry. This length field may occur either in the fixed or the variable area depending on the individual table.

ASAVE Linkage

Most JES3 macros (unless indicated) use the JES3 standardized linkage convention that saves and traces register contents. This convention results in calling a JES3 routine named ASAVE (in module IATGRSV), that saves registers 0 through 15 and passes registers 0-13 and 15 to the called program unchanged. Register 14 contains the return address in ASAVE.
The ARETURN macro reloads registers 2 through 14. Registers 15, 0, and 1 are returned to the caller as set by the called routine.

Registers 11 and 12 must be set up with the addresses of the FCT and TVT, respectively, before you issue the ACALL, ARETURN, or any other JES3 macro that uses ASAVE linkage.

ASAVE linkage creates a trace of the entry and the return in the JES3 trace tables.

See “ASAVE Linkage and Save Area Use” on page 26 and the description of the ARETURN macro for more details on ASAVE linkage.

Rules for Addressing Mode

When your program issues a JES3 macro, your program must be in 31-bit addressing mode. The one exception to this rule is macro IATXLPI, which must be called in 24-bit addressing mode.

If you use an inner macro (or any macro with a TYPE parameter) to generate a CSECT within a module, be sure to include the AMODE and RMODE statements related to the CSECT name in the beginning of the module. If a module contains more than one CSECT, you must include an AMODE and an RMODE statement in the module for every CSECT in the module. This includes data CSECTs generated from macros.

Executable Macro Instructions

This section describes the function and parameters of each JES3 macro that is used primarily to generate executable code. The descriptions are in alphabetic order of the macro names.

The functional areas of JES3 in which the macros are used are shown in Table 16.

The macros are presented to aid in understanding the JES3 code; the information contained herein does not define a particular programming interface.

Table 16. JES3 Executable Macro Instructions by JES3 Function

<table>
<thead>
<tr>
<th>Dump Formatting Macro Instructions</th>
<th>Spool Data Management Macro Instructions</th>
<th>JES3 Checkpoint Macro Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABNCODE</td>
<td>ABNGET</td>
<td>ABNVRFY</td>
</tr>
<tr>
<td>ABNCVDEC</td>
<td>ABNPUT</td>
<td>IATXMGET</td>
</tr>
<tr>
<td>ABNCVHEX</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>APOINT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>APURGE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>APUTBUF</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ARELEASE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ATRACK</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AWRITE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>IATXBMIO</td>
<td></td>
</tr>
<tr>
<td></td>
<td>IATXCV</td>
<td></td>
</tr>
<tr>
<td></td>
<td>IATXIOX</td>
<td></td>
</tr>
<tr>
<td></td>
<td>IATXCPYF</td>
<td></td>
</tr>
<tr>
<td></td>
<td>IATXJR</td>
<td></td>
</tr>
<tr>
<td></td>
<td>IATXRDC</td>
<td></td>
</tr>
<tr>
<td></td>
<td>IATXRELC</td>
<td></td>
</tr>
<tr>
<td></td>
<td>IATXSAS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>IATXSTTA</td>
<td></td>
</tr>
<tr>
<td></td>
<td>JESREAD</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PURCHAIN</td>
<td></td>
</tr>
<tr>
<td></td>
<td>WRTCHAIN</td>
<td></td>
</tr>
</tbody>
</table>
Table 16. JES3 Executable Macro Instructions by JES3 Function (continued)

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Instruction</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>IATXCKPT</td>
<td>IATXCKPNT</td>
<td>JESCKPNT</td>
</tr>
<tr>
<td>IATXPTCK</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**DSP Program Management Macro Instructions**

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Instruction</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACALL</td>
<td>IATXSF</td>
<td>IATXWRE</td>
</tr>
<tr>
<td>ADELETE</td>
<td>IATXFRQ</td>
<td>JESTAE</td>
</tr>
<tr>
<td>ALOAD</td>
<td>IATXGENF</td>
<td>RQTAADD</td>
</tr>
<tr>
<td>ARETURN</td>
<td>IATXGFC</td>
<td>RQTADEL</td>
</tr>
<tr>
<td>FAILDSP</td>
<td>IATXGRO</td>
<td>RQTAPUT</td>
</tr>
<tr>
<td>IATXATF</td>
<td>IATXIGA</td>
<td>VIOLATE</td>
</tr>
<tr>
<td>IATXCNT</td>
<td>IATXIOE</td>
<td></td>
</tr>
</tbody>
</table>

**Resource Control Macro Instructions**

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Instruction</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADEQ</td>
<td>ATEST</td>
<td>NCKADD</td>
</tr>
<tr>
<td>AENQ</td>
<td>AWAIT</td>
<td>NCKDEL</td>
</tr>
</tbody>
</table>

**Storage Management Macro Instructions**

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Instruction</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGETMAIN</td>
<td>IATXBGM</td>
<td>IATXMVXM</td>
</tr>
<tr>
<td>APUTMAIN</td>
<td>IATXBPL</td>
<td>IATXPXGM</td>
</tr>
<tr>
<td>IATXBFM</td>
<td>IATXDPL</td>
<td>IATXRCL</td>
</tr>
<tr>
<td></td>
<td>IATXGCL</td>
<td>IATXSQE</td>
</tr>
</tbody>
</table>

**Communication Macro Instructions**

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Instruction</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONCNVRT</td>
<td>IATXCNDB</td>
<td>INTERCOM</td>
</tr>
<tr>
<td>CONREVRT</td>
<td>IATXCNRS</td>
<td>JESMSG</td>
</tr>
<tr>
<td>CONSPROF</td>
<td>IATXSIMG</td>
<td>JSSERV</td>
</tr>
<tr>
<td>DEQMSG</td>
<td>IATXIWIT</td>
<td>LOGIN</td>
</tr>
<tr>
<td>DLOCOFF</td>
<td>IATXMD</td>
<td>MDSMAG</td>
</tr>
<tr>
<td>DLOCON</td>
<td>IATXMMLWO</td>
<td>MDSMSG</td>
</tr>
<tr>
<td>DSQLOC</td>
<td>IATXNTS</td>
<td>MESSAGE</td>
</tr>
<tr>
<td>IATXIO</td>
<td>IATXRCLV</td>
<td>SSISERV</td>
</tr>
<tr>
<td></td>
<td>IATXSCN1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>IATXSCN2</td>
<td></td>
</tr>
</tbody>
</table>

**Job Management Macro Instructions**

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Instruction</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>AJOBNUM</td>
<td>IATXRABD</td>
<td>MDSALLOC</td>
</tr>
<tr>
<td>GETUNIT</td>
<td>IATXRABP</td>
<td>MDSDSN</td>
</tr>
<tr>
<td>IATXADD</td>
<td>IATXSCCH</td>
<td>MDSERRQ</td>
</tr>
<tr>
<td>IATXARL</td>
<td>IATXSEC</td>
<td>MDSJERR</td>
</tr>
<tr>
<td>IATXBNQ</td>
<td>JDSADD</td>
<td>MDSJGET</td>
</tr>
<tr>
<td>IATXDEL</td>
<td>JDSGET</td>
<td>MDSJST</td>
</tr>
<tr>
<td>IATXELA</td>
<td>JDSHOLD</td>
<td>MDSVGET</td>
</tr>
<tr>
<td>IATXELD</td>
<td>JDSPOINT</td>
<td>MDSVLM</td>
</tr>
<tr>
<td>IATXELS</td>
<td>JDSPUT</td>
<td>NCBTAADD</td>
</tr>
<tr>
<td>IATXGET</td>
<td>JDSREL</td>
<td>NCBTAFND</td>
</tr>
<tr>
<td>IATXJCT</td>
<td>JNADD</td>
<td>NCBTGET</td>
</tr>
<tr>
<td>IATXJDS</td>
<td>JNCBHLD</td>
<td>NCBTAPUT</td>
</tr>
<tr>
<td>IATXJMR</td>
<td>JNCBREL</td>
<td>NCBTAREL</td>
</tr>
<tr>
<td>IATXJQE</td>
<td>JNDEL</td>
<td>PUTUNIT</td>
</tr>
<tr>
<td>IATXLOC</td>
<td>JNGET</td>
<td>SUPFLAG</td>
</tr>
<tr>
<td>IATXRABC</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Output Service Macro Instructions**

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Instruction</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>IATXDST</td>
<td>IATXOFP</td>
<td>IATXPRMD</td>
</tr>
<tr>
<td>IATXGOSE</td>
<td>IATXOSPC</td>
<td>IATXSMF</td>
</tr>
<tr>
<td>IATXOSCI</td>
<td>IATXOSPM</td>
<td>IATXSPR</td>
</tr>
<tr>
<td>IATXIPOCO</td>
<td>IATXOSSC</td>
<td>IATXSR</td>
</tr>
<tr>
<td>IATXOSG</td>
<td>IATXOSUP</td>
<td>SPINOFF</td>
</tr>
<tr>
<td>IATXOSOI</td>
<td>IATXOSWS</td>
<td>WRITEOSE</td>
</tr>
<tr>
<td>IATXOSO00</td>
<td>IATXPOSE</td>
<td></td>
</tr>
</tbody>
</table>
### JES3 Initialization Macro Instructions

<table>
<thead>
<tr>
<th>Macro</th>
<th>Instruction</th>
<th>Instruction</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>IATXVAL</td>
<td>INITMWLE</td>
<td>ITREAD</td>
<td>IATXOPEN</td>
</tr>
<tr>
<td>ICARDDR</td>
<td>IPURGE</td>
<td>IWRITE</td>
<td>JESOPEN</td>
</tr>
<tr>
<td>ICONVBIN</td>
<td>ISCAN1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ICONVHEX</td>
<td>ISCAN2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ISORT</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### JES3 Validation/Restart Macro Instructions

<table>
<thead>
<tr>
<th>Macro</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>IATXVFDB</td>
<td>IATXVSRE</td>
</tr>
<tr>
<td>IATXVMSG</td>
<td>IATXVSERV</td>
</tr>
</tbody>
</table>

### Unit Record and Tape I/O Macro Instructions

<table>
<thead>
<tr>
<th>Macro</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>JESCLOSE</td>
<td>JESOPEN</td>
</tr>
</tbody>
</table>

### SNA RJP Macro Instructions

<table>
<thead>
<tr>
<th>Macro</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>IATXCDVE</td>
<td>IATXRPT</td>
</tr>
<tr>
<td>IATXCRPL</td>
<td>IATXMBFE</td>
</tr>
<tr>
<td>IATXEQ</td>
<td>IATXNRS</td>
</tr>
<tr>
<td>IATXENQ</td>
<td>IATXPOST</td>
</tr>
<tr>
<td>IATXENT</td>
<td>IATXPSCL</td>
</tr>
<tr>
<td>IATXERCK</td>
<td>IATXRST</td>
</tr>
<tr>
<td>IATXFCB</td>
<td>IATXRUGT</td>
</tr>
<tr>
<td>IATXFSV</td>
<td>IATXRUP</td>
</tr>
<tr>
<td>IATXFWSB</td>
<td>IATXSNMGR</td>
</tr>
<tr>
<td>IATXGSV</td>
<td>IATXSNM</td>
</tr>
<tr>
<td>IATXLRGT</td>
<td>IATXSCM</td>
</tr>
</tbody>
</table>

### JES3 Inner Macro Instructions

<table>
<thead>
<tr>
<th>Macro</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>IATXTREG</td>
<td>IATZLOAD</td>
</tr>
<tr>
<td>IATZCALL</td>
<td>IATZMEXC</td>
</tr>
<tr>
<td>IATZHEX</td>
<td>IATZMNOD</td>
</tr>
<tr>
<td></td>
<td>IATZMOVE</td>
</tr>
<tr>
<td></td>
<td>IATZPARM</td>
</tr>
<tr>
<td></td>
<td>IATZPCHK</td>
</tr>
<tr>
<td></td>
<td>RQTAGEN</td>
</tr>
</tbody>
</table>

### Multitasking Macro Instructions

<table>
<thead>
<tr>
<th>Macro</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>IATXATDE</td>
<td>IATXSTMD</td>
</tr>
<tr>
<td>IATXTMD</td>
<td>IATXTEST</td>
</tr>
</tbody>
</table>

### BSC Services Macro Instruction

<table>
<thead>
<tr>
<th>Macro</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>IATXJLOC</td>
<td>IATXOSBM</td>
</tr>
</tbody>
</table>

### MVS/BDT Macro Instruction

<table>
<thead>
<tr>
<th>Macro</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>IATXLP</td>
<td>IATXOSBM</td>
</tr>
</tbody>
</table>

### FSS Service Macro Instructions

<table>
<thead>
<tr>
<th>Macro</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>IATXFSS</td>
<td>IATXPDQ</td>
</tr>
</tbody>
</table>

### Miscellaneous Macro Instructions

<table>
<thead>
<tr>
<th>Macro</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACVX</td>
<td>IATXJBNO</td>
</tr>
<tr>
<td>ATIME</td>
<td>IATXJOB</td>
</tr>
<tr>
<td>DEVSCAN</td>
<td>IATXLPJ3</td>
</tr>
<tr>
<td>IATXAMDV</td>
<td>IATXMNMX</td>
</tr>
<tr>
<td>IATXCUE</td>
<td>IATXMPC</td>
</tr>
<tr>
<td>IATXCVB</td>
<td>IATXPHEX</td>
</tr>
<tr>
<td>IATXCVD</td>
<td>IATXPJ3</td>
</tr>
<tr>
<td>IATXDEV</td>
<td>IATXRES</td>
</tr>
<tr>
<td>IATXDYH</td>
<td>IATXPRG</td>
</tr>
<tr>
<td>IATXDYT</td>
<td>IATXPTCH</td>
</tr>
<tr>
<td>IATXFIFO</td>
<td>IATXRMEP</td>
</tr>
<tr>
<td></td>
<td>IATXSTAT</td>
</tr>
<tr>
<td></td>
<td>IATXSYSU</td>
</tr>
<tr>
<td></td>
<td>IATXTOD</td>
</tr>
<tr>
<td></td>
<td>IATXTRC</td>
</tr>
<tr>
<td></td>
<td>MDSID</td>
</tr>
<tr>
<td></td>
<td>MTRDDEF</td>
</tr>
<tr>
<td></td>
<td>MOVEDATA</td>
</tr>
<tr>
<td></td>
<td>MTBL</td>
</tr>
<tr>
<td></td>
<td>ZEROCORE</td>
</tr>
</tbody>
</table>
**ERROR and NORMAL Parameters**

For some macros, the parameter “ERROR=address” can be coded. This parameter specifies the address to which the macro processing routine returns if an error occurs. When control is passed to the address, register 0 contains a dump code and register 15 contains a return code, unless otherwise specified. The return codes and some of the dump codes are described following the macro description. For a description of every dump code, see 

If an error occurs and ERROR was not specified, the dynamic support program (DSP) will be ended using the FAILDSP macro.

For most macros, the parameter “NORMAL=address” can be coded. This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

If NORMAL=address is not coded, the issuing program resumes execution at the instruction that immediately follows the macro.

The ERROR and NORMAL parameters are shown in all applicable macro syntax diagrams.

**Dump Codes**

All dump codes are described in 

Some of the dump codes returned from a macro are described following the macro description. The dump code is sometimes shown as “nnn,” which represents any number from 0 through 999. This specifies that an error occurred in another macro call and that the dump code will be the one issued by the other macro.

**ABACKR (Reposition Pointer in Multirecord File)**

The ABACKR macro repositions the file pointer in a multirecord file (MRF) to a record preceding the current record or preceding the end-of-file (EOF) indicator.

**Syntax**

```plaintext
label| ABACKR™-FDB™=FDB address ,EOD™=address |
```

```plaintext
| ,LINECNT™=number ,NORMAL™=address ,ERROR™=address |
```

**Parameters**

- **FDB=**
  Specifies the address of the file description block (FDB) for the multirecord file to be repositioned.
EOD=
Specifies the address to which the routine returns if end-of-data (EOD) has been reached.

LINECNT=
Specifies the number of records backward the file is to be repositioned. A zero indicates the file pointer should be set to the record just passed by the ADEBLOCK macro.

NORMAL=
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

ERROR=
This parameter specifies the address to which the macro processing routine returns if an error occurs. When control is passed to the address, register 0 contains a dump code and register 15 contains a return code, unless otherwise specified. The return codes and dump codes are described below. For a description of every dump code, see z/OS JES3 Diagnosis.

If an error occurs and ERROR was not specified, the dynamic support program (DSP) will be ended using the FAILDSP macro.

Return Codes
Following are the return codes and dump codes returned in registers 15 and 0, respectively:

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Dump Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>04</td>
<td>711</td>
<td>FDB not opened.</td>
</tr>
<tr>
<td>08</td>
<td></td>
<td>Failure on internal call (APUTBUF). See the APUTBUF macro description for the dump code.</td>
</tr>
<tr>
<td>0C</td>
<td>701</td>
<td>FDB not in file directory.</td>
</tr>
<tr>
<td>10</td>
<td>725</td>
<td>Unrecoverable I/O error.</td>
</tr>
<tr>
<td>14</td>
<td>720</td>
<td>Issued against output file.</td>
</tr>
<tr>
<td>18</td>
<td>748</td>
<td>Incorrect note passed.</td>
</tr>
</tbody>
</table>

Linkage
This macro uses ASAVE linkage.

Restrictions
Do not issue this macro in any routine that might run under the JES3 auxiliary task.

ABLOCK (Add Record to a Multirecord File)
The ABLOCK macro adds either a logical record or a record section of a spanned record to a multirecord file (MRF). (A spanned record is a record that is larger than the spool buffer.) This macro may be issued only after an ALOCATE macro has been issued.

Syntax

```
label ABLOCK FDB=(R1) address, COUNT=(R0) number
```
Parameters

**FDB=**
Specifies the address of the FDB for the MRF being blocked.

**COUNT=**
Specifies the number of bytes to be blocked. The number must be in the range from 1 to the buffer size minus 44.

**SPAN=**
Specifies whether the data to be blocked is spanned, and, if spanned, which record section of the spanned record this ABLOCK macro pertains to.

- **FIRST**  
  Specifies that this data is the first record section of a spanned record.

- **LAST**  
  Specifies that this data is the last record section of a spanned record.

- **NTH**  
  Specifies that this data is a record section between the first and last record sections of a spanned record.

- **NO**  
  Specifies that this record is not a spanned record.

**LRECL=**
Specifies whether the logical record length is to be taken from FCTLRECL for non-spanned records. It is required to pass the logical record length for blank truncated records as the COUNT parameter contains only the truncated length.

For spanned records, this parameter will be ignored (logical record length always taken from FCTLRECL).

For non-spanned records, logical record length will be taken from FCTLRECL if LRECL=YES and it will be taken from COUNT if LRECL=NO.

**NORMAL=**
This parameter specifies that after macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

**ERROR=**
This parameter specifies the address to which the macro processing routine returns if an error occurs. When control is passed to the address, register 0 contains a dump code and register 15 contains a return code, unless otherwise specified. The return codes and dump codes are described below. For a description of every dump code, see [z/OS JES3 Diagnosis](https://www.ibm.com/docs/en/zos/2.5.0).

If an error occurs and ERROR was not specified, the dynamic support program (DSP) will be ended using the FAILDSP macro.

**Return Codes**
Following are the return codes and dump codes returned in registers 15 and 0, respectively:
<table>
<thead>
<tr>
<th>Return Code In Hex</th>
<th>Dump Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>04</td>
<td>718</td>
<td>No preceding ALOCATE macro.</td>
</tr>
<tr>
<td>08</td>
<td></td>
<td>Failure on internal call (APUTBUF). See the APUTBUF macro description for the dump code.</td>
</tr>
<tr>
<td>0C</td>
<td>712</td>
<td>Count too large.</td>
</tr>
<tr>
<td>10</td>
<td>725</td>
<td>Unrecoverable I/O error.</td>
</tr>
<tr>
<td>14</td>
<td>729</td>
<td>Spool space could not be allocated to the multirecord file because of an incorrect job track allocation table (TAT) or data set TAT.</td>
</tr>
<tr>
<td>08</td>
<td>721</td>
<td>Seek search address not valid, or a program check during CCW build.</td>
</tr>
</tbody>
</table>

### Linkage

This macro uses ASAVE linkage.

### Restrictions

Do not issue this macro in any routine that might run under the JES3 auxiliary task.

### ABNCODE (Generate Global Symbol for Format Module)

The JES3 abend formatting routines use the ABNCODE macro to control the CSECT name of a module. The module entry code uses ABNCODE to set a global symbol to the CSECT name. The format routines then use ABNCODE to pick up the CSECT name from the global symbol.

### Syntax

```
/SM590000/SM630000
ABNCODE

lable

CODE type

ENTRY

EXIT

EXITONLY

WORKAREA
```

### Parameters

**CODE=**

**ENTRY**

Causes generation of a CSECT statement and initialization code for the format module named by the statement label. (If CODE=ENTRY is coded, this statement must contain a label.) CODE=ENTRY is valid only once per module.

**EXIT**

Generates the code to exit the module and also generates the local abend work area used by the module.
EXITONLY
Generates the code to exit the module (the local abend work area is not generated)

WORKAREA
Generate the local abend work area only

If the CODE parameter is omitted, a CSECT statement is generated using the statement label coded on the previous CODE=ENTRY form.

Linkage
This macro expands in-line.

Restrictions
Do not issue this macro in any routine that might run under the JES3 auxiliary task.

ABNCVDEC (Convert Data to Decimal)
The JES3 abend processing routines, exclusively, use the ABNCVDEC macro to convert a fullword to ten unpacked decimal digits.

On return, register 1 contains the address of the 10-byte unpacked decimal field.

Syntax

```
> SM590000
ABNCVDEC
label
(R1) DATA=(Rx)
address
NORMAL=address
```

Parameters

DATA=

(R1 or Rx)
Specifies the register containing the data to be converted.

address
Specifies the address of the data to be converted.

NORMAL=
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

Linkage
This macro issues a BALR R14,R15 for linkage.

Restrictions
Do not issue this macro in any routine that might run under the JES3 auxiliary task.
ABNCVHEX

ABNCVHEX (Convert Data to Hexadecimal)

The JES3 abend processing routines use the ABNCVHEX macro to convert four bytes of data to an 8-byte character string. The character string represents the converted data in hexadecimal notation.

On return, register 1 contains the address of the 8-byte character string.

Syntax

```
/SM590000/SM590000
ABNCVHEX
/SM590000/SM630000
```

Parameters

**DATA=**

(R1 or Rx)

Specifies the register containing the data to be converted.

**address**

Specifies the address of the data to be converted.

**NORMAL=**

This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

Linkage

This macro issues a BALR R14,R15 for linkage.

Restrictions

Do not issue this macro in any routine that might run under the JES3 auxiliary task. Only the JES3 abend processing routines may use this macro.

ABNGET (Get Storage Address of a Specified Area)

The ABNGET macro gets the address of an area in the JES3 address space during JES3 abend formatting. The routine that services the request is in module IATABN0 for JES3 abend formatting and in module IATABPR for IPCS (interactive problem control system) dump formatting.

On return, register 1 contains the address of the area. The area has the length of the alignment attribute of the address specified by the AREA parameter. For example, doubleword alignment obtains 8 bytes of data, halfword alignment obtains 2 bytes.

**Note:** The IATXMGET macro provides a more efficient way to get a storage address than the ABNGET macro. Use the IATXMGET macro instead of the ABNGET macro whenever possible.
### Syntax

```plaintext
ABNGET AREA=label

(R1) BASE=(reg), REF=label, WREG=(reg)

ERROR= address, NORMAL= address
```

### Parameters

**AREA=**

Specifies the address of the area in the dump to be obtained. If AREA=address is specified, then BASE= and REF= must be specified if this is the first expansion of ABNGET in this module.

**BASE=**

Specifies the base register to be used in performing the calculation of the address specified for AREA.

**REF=**

Specifies the label in the control block which the BASE register points to.

**WREG=**

Specifies a work register to be used during the address calculation. If not specified, it defaults to R1. The address calculation is done as follows:

```plaintext
LAWREG, AREA-REF
ALRWREG, BASE
```

where the values of WREG, AREA, REF, and BASE are symbolically substituted in the above instructions.

The values specified for BASE and REF are saved in global assembler variables. If on an invocation of ABNGET, BASE or REF is not specified, then it will default to its last specified value.

The values specified for BASE and WREG must be different.

Registers 0, 1, 14, and 15 are changed by the macro and should not be specified for BASE.

**ERROR=**

Specifies the address of a routine to which control returns when: (1) IPCS cannot retrieve the requested address from a dump; or (2) during JES3 abend formatting, the specified area was not obtained or the area does not have the JES3 storage protection key.

**NORMAL=**

This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.
ABNGET

Linkage

This macro issues a BALR R14,R15 for linkage.

Restrictions

Do not issue this macro in any routine that might run under the JES3 auxiliary task.

ABNPUT (Write a Line to the Dump Data Set)

The ABNPUT macro writes a line to the dump data set. The routine that services the request is in module IATABN0 for JES3 abend formatting and in module IATABPR for IPCS (interactive problem control system) dump formatting.

Syntax

```
label ABNPUT PCW Data address, NORMAL= address
```

Data address

```
(R1) address
```

Parameters

PCW=

Specifies the address of the print control word for the line to be written to the dump data set.

NORMAL=

This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

Linkage

This macro issues a BALR R14,R15 for linkage.

Restrictions

Do not issue this macro in any routine that might run under the JES3 auxiliary task.

ABNVRFY (Verify Specified Address)

The ABNVRFY macro determines the validity of a specified address. An address is valid if:

1. It addresses a fullword,
2. It is within the boundaries of the area dumped, and
3. The storage protection key of the issuing program is either 0 or the same as the addressed location.
Syntax

```
\textbf{ABNVRFY} - \text{DATA} = (\text{Rx}), \text{ERROR} = \text{address}, \text{NORMAL} = \text{address}
```

Parameters

- **DATA=**
  Specifies the register containing the address to be verified.

- **ERROR=**
  Specifies the address to which the routine returns if the address is incorrect.

- **NORMAL=**
  This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

Linkage

This macro issues a BALR R14,R15 for linkage.

Restrictions

Do not issue this macro in any routine that might run under the JES3 auxiliary task.

ACALL (Call a Routine)

The ACALL macro allows the calling module to enter a routine.

Syntax

```
\textbf{ACALL} - \text{label}
```

Called Routine

```
\text{EOD} = \text{address}, \text{EOF} = \text{address}, \text{NAVAIL} = \text{address},
\text{ERROR} = \text{address}, \text{IPL} = \text{address}, \text{BUSY} = \text{address},
\text{REJECT} = \text{address}, \text{RJPCAN} = \text{address}, \text{NORMAL} = \text{address}
```
ACALL

Called Routine

\[
\begin{array}{c}
\text{ENTER} = \begin{cases}
(R15) \\
\text{reg} \\
\text{address} \\
\text{label}, \ A \\
\end{cases} \\
\text{EPLOC=address}
\end{array}
\]

Parameters

ENTER=
Specifies the address of the routine to be called.

When you specify ACALL ENTER=(label,A), the code generates an ADCON to get the address of the routine. When you specify ACALL ENTER=(label,R), the code generates a LARL to get the address of the routine. The "A" or "R" subparameter is not allowed when a register is specified for ENTER=. Below are examples of this usage (lines with "+" are lines generated when the macro expands).

Using the "A" or "R" parameter

Without the "A" or "R" subparameter:

\[
\begin{align*}
\text{ACALL } & \text{ENTER=} \text{DMREADIO} \\
+ & \text{LA } R15, \text{DMREADIO} \\
+ & \text{L } R14, \text{ASAVE} \\
+ & \text{BALR } R14, R14
\end{align*}
\]

With the "A" subparameter:

\[
\begin{align*}
\text{ACALL } & \text{ENTER=} (\text{DMREADIO}, A) \\
+ & \text{LA } R15, =A(\text{DMREADIO}) \\
+ & \text{L } R14, \text{ASAVE} \\
+ & \text{BALR } R14, R14
\end{align*}
\]

With the "R" subparameter:

\[
\begin{align*}
\text{ACALL } & \text{ENTER=} (\text{DMREADIO}, R) \\
+ & \text{LARL } R15, \text{DMREADIO} \\
+ & \text{L } R14, \text{ASAVE} \\
+ & \text{BALR } R14, R14
\end{align*}
\]

EPLOC=
Specifies the name of a label that contains the address of the routine to be called. If the EPLOC parameter is specified, the EP parameter cannot be specified and vice versa. Below is an example (lines with "+" are lines generated when the macro expands).

Example

VIWRDRTN DC A(0) Address of the READ routine

\[
\begin{align*}
\text{ACALL } & \text{EPLOC=} \text{VIWRDRTN} \\
+ & \text{L } R15, \text{VIWRDRTN} \\
+ & \text{L } R14, \text{SAVE} \\
+ & \text{BALR } R14, R14
\end{align*}
\]
SAVE=

**YES**  Specifies that entry is to be made through the ASAVE linkage routine, ensuring the integrity of registers 2 through 13.

**NO**  Specifies that entry is to be made through BALR R14, R15.

TRACE=

**YES**  The call will be traced in the JES3 trace table.

**NO**  The call will not be traced in the JES3 trace table.

This parameter is only valid with SAVE= YES.

EOD=  EOF=  NAVAIL=  ERROR=  IPL=  BUSY=  REJECT=  RJPCAN=  NORMAL=

Specifies return points. Only those return points known by the called routine can be specified. Each use of a parameter generates a branch instruction, in order of their appearance as parameters of this macro, following the branch and link to the called routine.

**Linkage**

ASAVE linkage for this macro is optional (see the SAVE parameter description below).

**Restrictions**

Before issuing the ACALL macro, you must ensure that register 11 points to the FCT and register 12 points to the TVT.

**ACLOSE (Close a Multirecord File)**

The ACLOSE macro closes a multirecord file (MRF).

**Syntax**

```
ACLOSE label FDB=FDB address, NORMAL=address
```

Error Return

ACLOSE

FDB address

 Error Return

Parameters

FDB=
Specifies the address of the FDB for the MRF being closed.

NORMAL=
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

ERROR=
This parameter specifies the address to which the macro processing routine returns if an error occurs. When control is passed to the address, register 0 contains a dump code and register 15 contains a return code, unless otherwise specified. The return codes and dump codes are described below. For a description of every dump code, see z/OS JES3 Diagnosis.

If an error occurs and ERROR was not specified, the dynamic support program (DSP) will be ended using the FAILDSP macro.

Return Codes
Following are the return codes and dump codes returned in registers 15 and 0, respectively:

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Dump Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>04</td>
<td>701</td>
<td>FDB not open or not in file directory.</td>
</tr>
<tr>
<td>0C</td>
<td>724</td>
<td>First or last spool address changed; the calling routine should checkpoint the FDB.</td>
</tr>
<tr>
<td>10</td>
<td>725</td>
<td>Unrecoverable I/O error on last buffer.</td>
</tr>
<tr>
<td>14</td>
<td>721</td>
<td>Program check during CCW build (DMFR received control)</td>
</tr>
<tr>
<td>18</td>
<td>729</td>
<td>Error accessing job/DS TAT</td>
</tr>
</tbody>
</table>

Linkage

This macro uses ASAVE linkage.

Restrictions

Do not issue this macro in any routine that might run under the JES3 auxiliary task.
ACVX (Convert Binary Word to Hex)

The ACVX macro generates a subroutine to convert four bytes of data to an 8-byte character string which represents the converted data in hexadecimal notation. When called, the subroutine assumes that the address of the 4 bytes to be converted is in register 1. On return from the subroutine, register 1 contains the address of the 8-byte character string. The subroutine returns to the address in register 14.

**Note:** Addressability of the subroutine must be established before calling the subroutine. Also, label the macro statement.

**Syntax**

```
ACVX (Convert Binary Word to Hex)
```

```
| label | ACVX
```

**Linkage**

This macro does not use ASAVE linkage.

ADEBLOCK (Retrieve a Record from a Multirecord File)

The ADEBLOCK macro retrieves a logical record or a record section of a spanned record from a multirecord file (MRF). (A spanned record is a record that is larger than the spool buffer.) When a logical record spans a buffer boundary, the ADEBLOCK routine combines the record into a contiguous area.

When the ADEBLOCK macro retrieves a spanned logical record, the data is retrieved in “record sections.” The length of the first and last record sections varies from 1 byte to the buffer size minus 44 bytes. The length of the middle record sections is always the buffer size minus 44 bytes.

To determine what type of data is being retrieved, check flag byte FCTSPFLG in the FCT.

- If FCTSPFLG=FCTNOSP, the data is a normal logical record.
- If flag FCTSPAN is on, the data is part of a spanned record.
- If FCTSPFLG=FCTSPFIR, the data is the first record section of a spanned record. In this case, field FCTSPLRL in the FCT contains the logical record length.
- If FCTSPFLG=FCTSPNTH, the data is a record section between the first and last record sections of a spanned record.
- If FCTSPFLG=FCTSPNTH, the data is the last record section of a spanned record.

To determine the carriage control information, check flag byte FCTCCFLG in the FCT.
ADEBLOCK

On return, register 1 contains the address of the record, register 0 contains the length of the record, and register 15 contains the original logical record length of the record. Original logical record length is also available in FCTLRECL while return from the macro.

**TRUNC=NO**

If TRUNC=NO is specified in the BUFFER or SYSOUT initialization statements, then the length of the record and original logical record length will be the same for non-spanned records. That is, register 0 and register 15 will have the same value on return.

**TRUNC=YES**

If TRUNC=YES is specified, then the length of the record will be different from the original logical record length, when trailing blanks existed in the record. That is, register 0 will have truncated length whereas register 15 will have the non-truncated length (original logical record length).

For spanned records, register 0 contains the 'record section' length whereas register 15 contains the original logical record length. FCTSPLRL and FCTLRECL will contain the same value, if TRUNC=NO. However, they will be different, if TRUNC=YES and trailing blanks exist in the spanned record. For pre=HJSS7708, DATCC extension does not exist in the record. Original logical record length is not available. In such cases, register R15 and FCTLRECL will be zero.

**Syntax**

```
ADEBLOCK label FDB=(R1) address, EOD=address

SPAN=NO, ERROR=address, NORMAL=address

ADEBLOCK label EPLOC=(R0) address

EP=name, CSECT=NO, REUSE=YES

(RE1) address

CDE=NO, R=NO
```

**Linkage**

This macro uses ASAVE linkage.

**Restrictions**

Do not issue this macro in any routine that might run under the JES3 auxiliary task.

ADELETE (Delete a Module)

The ADELETE macro deletes a module from storage or decreases a module's use count if more than one program is using a reentrant module.

**Syntax**

```
ADELETE label EPLOC=(R0) address

EP=name, CSECT=NO, REUSE=YES

(RE1) address

CDE=NO, R=NO
```

190 z/OS V1R11.0 JES3 Customization
ADELETE

Parameters

**EPLOC=**
Specifies the address of the name of the module to be deleted. The name must be left-justified in an 8-character field padded on the right with blanks.

**EP=**
Specifies the 1- to 8-character name of the module to be deleted.

**CSECT=**
Specifies whether the module is an executable module or a data module.
- **NO**
  Specifies that the module is not a data module. (The module will be considered reentrant if REUSE=NO.)
- **YES**
  Specifies that the module is a data module. (The module will never be considered reentrant.)

  **Note:** If CSECT=YES is specified, REUSE=YES must not be specified.

**REUSE=**
Specifies whether the module is serially reusable or not.
- **NO**
  Specifies that the module is not serially reusable.
- **YES**
  Specifies that the module is serially reusable.

  **Notes:**
  1. If REUSE=YES is specified, CSECT=YES must not be specified.
  2. REUSE=YES should be used only when deleting the DSP initialization and termination routines in module IATGRJR.

**CDE=**
Specifies the address of the JES3 directory element (JDE) for the CSECT to be deleted. The ALOAD macro returns this address in register 1.

**R=**

- **NO**
  Specifies that the current copy is not to be refreshed.
- **YES**
  Specifies that a fresh copy of the module is to be loaded when the next ALOAD macro is issued for the module.

**NORMAL=**
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

**Note:** Errors occurring during ADELETE processing result in a FAILDSP macro being issued. The three errors listed here can occur during ADELETE processing and cause DM abends. In all three cases, the JES3 dump code is returned in register one. The return code (in hexadecimal) is found in register 15. The meanings of the return codes are:
ADELETE

1. Return code 04 – The JES3 directory element (JDE) was not found in a search of the JDE queue of elements.

2. Return code 0C – An ADELETE request was made against a module that did not have any outstanding users (that is, the use count was zero), causing a delete synchronization error. In this case, register 0 contains the address of the JDE for the module being deleted.

3. Return code 10 – An incorrect delete request was made because the module name in the JDE did not match the module name provided. If a routine requests ADELETE services, it must provide a module name and, optionally, the address of a JES3 directory element. If both are provided, the name in the JDE must match the module name provided. If there is no match, an incorrect delete request has been made. For data CSECTs, an FCT may delete only those that it has loaded using the ALOAD macro. If a request is made to delete any other data CSECT, an incorrect delete request has been made. For incorrect delete requests, register 0 contains the JDE address.

Linkage

This macro uses ASAVE linkage.

Restrictions

This macro may be used only to delete a module that was loaded using the ALOAD macro. Do not issue this macro in any routine that might run under the JES3 auxiliary task.

ADEQ (Release a JES3 Resource)

The ADEQ macro releases a JES3 resource (for example, the FCT, or SYSUNITS table) previously obtained using the AENQ macro.

Syntax

```
ADEQ label
NAME name
(FCT)
(NORMAL) (address)
```

Parameters

**NAME=**

name

Specifies the name of the JES3 resource to be released. See the IATYRSC macro expansion in the TVT for the valid names.

(Rx)

Specifies a register containing the address of the name of the JES3 resource to be released.

**FCT=**

Specifies that all resources enqueued on the specified FCT are to be released. Also, if any JCT entries are still held, they are to be freed and the DSP holding the entries is to be failed. Any held job queue element (JQE) priority levels are also to be freed.
NORMAL=

This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

See the section on “Enqueueing and Locks” in Enqueueing and Locks on page 31 for additional information about this macro and an example of its use.

Linkage

This macro uses ASAVE linkage.

Restrictions

Do not use this macro in any routine that might run under the JES3 auxiliary address space.

AENQ (Obtain a JES3 Resource)

The AENQ macro obtains use of a JES3 resource (for example, the FCT, RESQUEUE, or SYSUNITs table).

Syntax

```
AENQ NAME=name (Rx) WAIT,BUSY=address
```

Parameters

NAME=

- **name**
  - Specifies the name of the JES3 resource to be obtained. See the IATYRSC macro expansion in the TVT for the valid names.

- **(Rx)**
  - Specifies a register containing the address of the name of the JES3 resource to be obtained.

BUSY=

- **WAIT**
  - Specifies that the macro processing routine is to wait until the resource becomes available.

- **address**
  - Specifies the address to which the routine is to return. If USE=EXC is specified and the calling routine wants to wait for the resources to become available, it can issue the AWAIT TYPE=OFF macro. The ECF address needed is returned in register 0. The ECF mask needed is returned in the low-order byte of register 1, with the other three bytes containing zeros. If USE=SHR is specified, the ECF mask and ECF address are not returned.
AENQ

NORMAL=
This parameter specifies that after the macro processing routine executes, the
issuing program can resume execution at an instruction other than the
instruction that immediately follows the macro.

USE=
Specifies how the issuing program wants to use the resource, either as
exclusive or as shared.

EXC
Specifies that the issuing program wants exclusive use of the JES3
resource.

SHR
Specifies that the issuing program wants shared use of the JES3 resource.
If USE=SHR is specified when BUSY=(address), the ECF mask and ECF
address are not returned.

See the section on “Enqueuing and Locks” in “Enqueuing and Locks” on page 31
for additional information about this macro and an example of its use.

Linkage
This macro uses ASAVE linkage.

Restrictions
Do not use this macro in any routine that might run under the JES3 auxiliary task.

AGETBUF (Get Buffer from JSAM Buffer Pool)
The AGETBUF macro obtains a buffer from the JSAM buffer pool of the calling
routine’s address space. If a buffer is not available, then the routine will wait until
one becomes available unless you specify the BUSY parameter. The buffer is
cleared to binary zeros before control returns to the calling module, unless bit
AIOGETBF is set in the TVT (which specifies it is not necessary to set the buffer to
zero).

The buffer address is returned in register 0.

Syntax

Parameters
BUSY=
Specifies the address to which control returns when a JSAM buffer is
unavailable. If the calling routine wants to wait for a buffer to become available,
it can issue the AWAIT TYPE=ON macro. The ECF address needed is returned
in register 0. The ECF mask needed is returned in the low-order byte of register
1, with the other three bytes containing zeros.

NORMAL=
This parameter specifies that after the macro processing routine executes, the
issuing program can resume execution at an instruction other than the
instruction that immediately follows the macro.
Linkage

This macro uses ASAVE linkage.

Restrictions

Do not use this macro in any routine that might run under the JES3 auxiliary task.

AGETMAIN (Get a Contiguous Storage Area)

The AGETMAIN macro gets an area of contiguous storage. On return, register 0 and register 1 both contain the address of the obtained storage.

Syntax

```
AGETMAIN label SIZE=(RO) (reg) address number, LOC=(BELOW,ANY) (ANY), SP=(R1) (reg) number, DBLWD, BNDRY=PAGE, BUSY=address
```

Parameters

**SIZE=**

Specifies the number of bytes of storage to obtain. If the address form is specified, it must be the address of a fullword containing the number of bytes required. If you use register 0, the number of bytes must be in the low-order three bytes of the register.

The SIZE and SP parameters can not specify the same register.

For storage obtained above 16 megabytes, the size value specified may be greater than 16 megabytes. The amount available is determined by the REGION parameter on the JCL used to start JES3 or by the default set by the IEFUSI exit.

**LOC=**

Defines the residence requirements of the storage being obtained, using the same manner as LOC= on the MVS GETMAIN macro. This is as follows:

**BELOW**

Obtain storage from below 16 megabytes in virtual storage. Pages are backed below 16 megabytes in central storage when fixed.

**(BELOW,ANY)**

The same as BELOW except that pages may be backed anywhere in real when fixed.

**ANY**

Obtain storage from anywhere in virtual storage (normally above 16 megabytes). Pages may be backed anywhere when fixed.

**(ANY,ANY)**

The same as ANY.
RES
Obtain storage from the same residence as the calling program. If the calling program resides below 16 megabytes, the storage is obtained below 16 megabytes virtual, and central storage is backed below 16 megabytes central when fixed. If the calling program resides above 16 megabytes, the storage is obtained from anywhere in virtual storage and is backed anywhere in central storage when fixed.

(RES,ANY)
The same as RES except storage may be backed anywhere in central storage when fixed regardless of the residence of the calling program.

SP=
Specifies the number of the subpool from which to get the storage. If SP= is omitted, JESPOOL (defined in IATYTVT) is assumed.

See “Using Storage Subpools” located in [z/OS MVS Programming: Assembler Services Guide]. If SP is specified as a register, the subpool number must be in the low-order byte of the register. The other bytes may contain any values. If in register 1, the other bytes will be changed by the macro expansion.

The SP and SIZE parameters can not specify the same register.

BNDRY=

DBLWD
Specifies that the obtained storage is to be on a doubleword boundary.

PAGE
Specifies that the obtained storage is to be on a page boundary.

BUSY=
Specifies the address to which the routine returns if there is insufficient storage to satisfy the request. The busy routine can wait for space to become available by issuing an AWAIT TYPE=ON macro using FCTAGET as the ECF mask and FCTFLAG1 as the ECF address.

If BUSY is not specified, then the AGETMAIN service routine waits for the requesting FCT until enough storage is freed to satisfy the request.

Linkage
This macro uses ASAVE linkage.

Restrictions
None.

AJOBNUM (Find or Return Job Number)
The AJOBNUM macro performs any of the following functions:
• Allocates the next available job number
• Determines if a specific job number is in use
• Deallocates a job number
• Allocates a specific job number, as requested
• Redefines the next available job number

If you include the JNUM parameter, the job number will be returned in binary form in register 0. If your installation requires job numbers to be in EBCDIC format (for example, when used in an operator command), you can use the IATXJOB macro to convert the number from binary to EBCDIC.
Syntax

Parameters

JNUM,
Specifies that a job number is to be returned from the pool of available job numbers. The job number, in binary, is returned to the caller in register 0. If no job numbers are available, the routine returns to the address specified by the ERROR parameter.

This parameter is a positional parameter.

Note: The maximum value for JNUM is 999999.

A=
Specifies that a specific job number is to be allocated. If the specified job number is not available, the routine returns to the address specified by the ERROR parameter.

(R0) or (reg)
Specifies the register containing the job number to be allocated.

callerid=address
Specifies the address of a 4-byte field containing the job number to be allocated.

CALLERID=address
Specifies the address of an 8-byte character string that identifies the job submitter or job owner for AJOBNUM.

This parameter is only valid with JNUM or A=.
**AJOBNUM**

**F=** Specifies that a specific job number should be tested to determine whether it is currently in use. If it is not in use, the routine returns to the address specified by the ERROR parameter.

**(R0) or (reg)**

Specifies the register containing the job number to be tested.

**address**

Specifies the address of a 4-byte field containing the job number to be tested.

**R=**

Specifies that a job number is no longer in use and is to be returned to the job number table. If this parameter is coded, then the ERROR parameter may not be coded.

**(R0) or (reg)**

Specifies the register containing the job number to be returned.

**address**

Specifies the address of a 4-byte field containing the job number to be returned.

**S=**

Specifies that the job number allocation pointer is to be reset. Subsequent job numbers are to be allocated beginning with the next available job number equal to or greater than the specified job number. If the specified job number is outside the range of valid job numbers, the routine returns to the address specified by the ERROR parameter and the job number allocation pointer is unchanged.

**(R0) or (reg)**

Specifies the register containing the job number.

**address**

Specifies the address of a 4-byte field containing the job number.

**NORMAL=**

This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at the instruction whose address is specified here, rather than at the instruction that immediately follows the macro.

**ERROR=**

For the **JNUM** parameter, specifies the address to which the routine returns if there are no more job numbers available. This macro returns an ECF mask and address that may be used to wait until a job number becomes available. The 31-bit ECF address is returned in register 0. The ECF mask needed is contained in the low-order byte of register 1, with the other three bytes containing zeros.

For the **F=** parameter, specifies the address to which the routine returns if the job number is not being used.

This parameter may not be specified with the **R** parameter.

For the **A=** parameter, specifies the address to which the routine returns if the specified job number is not available for allocation.

For the **S=** parameter, specifies the address to which the routine returns if the specified job number is outside the range of valid job numbers.
EXEMPT=YES INO
Specifies whether the requester is allowed to obtain a job number after the maximum number of jobs allowed (threshold) is reached.

EXEMPT=YES means the requester is exempt from being limited to the threshold. EXEMPT=NO means that the requester is not exempt and the threshold will be observed.

The threshold is calculated by taking the number of jobs defined on the JES3 OPTIONS initialization statement minus the lesser number of either 100 or 5% of the number of jobs on the JES3 OPTIONS initialization statement.

For example: 3000 jobs have been defined on the JES3 OPTIONS initialization statement. 5% of 3000 = 150. 100 is less than 150. Therefore, there are 100 job numbers available to EXEMPT=YES requestors is 100. The threshold is 2900.

When there are no more available job numbers, for either an exempt or non-exempt requester, the system invokes the caller’s error routine.

This parameter is only valid with JNUM or A=.

Linkage
This macro uses ASAVE linkage.

Restrictions
Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task.

ALOAD (Load a Module into Storage)
The ALOAD macro loads a module into storage. If the module requested is reentrant and already in storage, the use count for the module will be increased by one.

On return, register 0 contains the entry point address of the module. Register 1 contains the address of the JES3 directory element (JDE) for the module.

Syntax

```
ALOAD

Called Routine

-REUSE=
-NO
-YES

,CSECT=

,DCB=(RI)

,ERROR=address

,NORMAL=address

,NO

,label
```

Chapter 2. JES3 Macros 199
ALOAD

Called Routine

\[ \text{EPLOC} \rightarrow \text{RO} \]
\[ \text{EP} \rightarrow \text{name} \rightarrow \text{address} \]

Parameters

**EPLOC=**
Specifies the address of the name of the module to be loaded. The name must be left-justified in an 8-byte field, padded on the right with blanks.

**EP=**
Specifies the 1- to 8-character name of the module to be loaded. All eight characters of the module name are used during the ALOAD macro processing; all module names (JES3 and user) must be unique.

**CSECT=**
Specifies whether the module is an executable module or a data module.

**NO**
Specifies that the module is not a data module. (The module will be considered reentrant if REUSE=NO.)

**YES**
Specifies that the module is a data module. (The module will never be considered reentrant.)

**Note:** If CSECT=YES is specified, REUSE=YES must not be specified.

**REUSE=**
Specifies whether the module is serially reusable or not.

**NO**
Specifies that the module is not serially reusable.

**YES**
Specifies that the module is serially reusable.

**Notes:**
1. If REUSE=YES is specified, CSECT=YES must not be specified.
2. REUSE=YES should be used only when loading the DSP initialization and termination routines in module IATGRJR.

**DCB=**
Specifies the address of the opened data control block (DCB) of the load module library from which the module is to be loaded.

**NORMAL=**
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

**ERROR=**
Specifies the address to which the macro processing routine returns if an error occurs.

If an error occurs and the ERROR parameter was not specified, the DSP is ended using the FAILDSP macro.

The three errors listed here can occur during ALOAD processing and cause DM abends. (In all three cases, register 1 contains the JES3 dump code.)
1. Return code **04** – The JES3 AGETMAIN for a storage buffer can fail. If an error return is specified, register 15 contains the reason code returned from the MVS GETMAIN macro, register 0 contains the ECF mask and the ECF address returned from a busy AGETMAIN, and register 1 contains the JES3 dump code (in hex). If an error return is not specified, register 4 contains the AGETMAIN busy return code (X'04'), and a FAILDSP will be issued with a DM083.

2. Return code **14** – The MVS LOAD macro may fail. If it does, message IAT6308 is issued indicating the MVS LOAD ABEND code and return code. If an error return is specified, register 1 will have the JES3 dump code (in hex) and register 15 will have the ALOAD return code for a LOAD failure (X'14'). If an error return is not specified, register 4 contains the ALOAD error return code (X'14') and a FAILDSP will be issued with a DM080.

   See [z/OS MVS Programming: Assembler Services Guide](https://www.ibm.com/support/docview.wss?uid=swg27025904) for more information about the ABEND and reason codes returned by the MVS LOAD macro.

3. Return code **18** – The MVS BLDL to find the module and obtain its size requirements can fail. If it does, message IAT6308 is issued indicating the reason code and return code from the BLDL macro. If an error return is specified, register 1 will contain the JES3 dump code (in hex) and register 15 will contain the BLDL failed return code (X'18'). If an error return is not specified, register 4 contains the BLDL error return code (X'18'), and JES3 will issue a FAILDSP with a DM081.

### Linkage

This macro uses ASAVE linkage.

### Restrictions

Do not use this macro in any routine that might run under the JES3 auxiliary task.

### ALOCATE (Locate Space for a Logical Record)

The ALOCATE macro locates space for a logical record in a multirecord file (MRF) that is open for output. The ALOCATE macro is used with the ABLOCK macro. There must be a corresponding ABLOCK macro after every ALOCATE macro for a given MRF.

On return, register 1 contains the address of the location within the JSAM buffer where the data is to be placed.

#### Syntax

```
ALOCATE label FDB=(R1) address, COUNT=(R0) number,

,NORMAL=address, ERROR=address
```

#### Parameters

- **FDB**=
  - Specifies the address of the FDB for the MRF in which space is being located.

- **COUNT**=
  - Specifies the length of the record for which space is being located. The number is in the range from 1 to buffer size minus 32.
ALOCATE

**NORMAL=**
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

**ERROR=**
This parameter specifies the address to which the macro processing routine returns if an error occurs. When control is passed to the address, register 0 contains a dump code and register 15 contains a return code, unless otherwise specified. The return codes and dump codes are described below. For a description of every dump code, see [z/OS JES3 Diagnosis](https://www.ibm.com/support/docview.wss?uid=swg21145894).

If an error occurs and ERROR was not specified, the dynamic support program (DSP) will be ended using the FAILDSP macro.

**Return Codes**
Following are the return codes and dump codes returned in registers 15 and 0, respectively:

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Dump Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>04</td>
<td>717</td>
<td>The preceding ALOCATE macro was not followed by an ABLOCK macro.</td>
</tr>
<tr>
<td>0C</td>
<td>711</td>
<td>The file is not open or is not in the file directory.</td>
</tr>
<tr>
<td>10</td>
<td>712</td>
<td>The value specified for the COUNT parameter is too large.</td>
</tr>
</tbody>
</table>

**Linkage**
This macro uses ASAVE linkage.

**Restrictions**
Do not use this macro in any routine that might run under the JES3 auxiliary task.

**ANOTE (Note Location in a Multirecord File)**
The ANOTE macro notes the current location in a multirecord file (MRF) for later repositioning to that location by the APOINT macro.

On return, register 1 contains the address of a doubleword field containing the information.

**Syntax**
```
label ANOTE FDB=(R1) address ,NORMAL=address
```

**Parameters**
**FDB=**
Specifies the address of the FDB for the MRF.

**NORMAL=**
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.
Linkage

This macro uses ASAVE linkage.

Restrictions

Do not use this macro in any routine that might run under the JES3 auxiliary task.

AOPEN (Open a Multirecord File)

The AOPEN macro opens a multirecord file (MRF) for later output (blocking) or input (deblocking) of data.

Syntax

```
AOPEN FDB=(R1) address, TYPE=IN OUT

PRTY=number
TATPTR=(R0) address
SKIP=YES
CARR=ASA MAC
OPTCD=J
ERROR=address
NORMAL=address
```

Parameters

FDB=
Specifies the address of the FDB for the MRF being opened.

TYPE=

IN  Specifies that the MRF is to be opened for input (deblocking).

OUT  Specifies that the MRF is to be opened for output (blocking).

PRTY=
Specifies the priority in which JES3 spool data management will service the requests for this MRF. The range is 0 to 15.

TATPTR=
Specifies the address of the track allocation table (TAT) FDB. If TATPTR is not coded, the TAT FDB from the caller’s resident job queue (RESQUEUE) will be used.

SKIP=
Specifies whether the record should be skipped and the following record read if a permanent read error occurs. SKIP is used with TYPE=IN.

NO  Specifies that no recovery is to be attempted.

YES  Specifies that recovery is to be attempted.

CARR=

ASA
Specifies that the MRF contains ASA carriage control characters.

MAC
Specifies that the MRF contains machine carriage control characters.

OPTCD=
Specifies that the MRF to be opened contains table reference characters.

ERROR=
This parameter specifies the address to which the macro processing routine returns if an error occurs. When control is passed to the address, register 0 contains a dump code and register 15 contains a return code, unless otherwise specified. The return codes and dump codes are described below. For a description of every dump code, see \textit{z/OS JES3 Diagnosis}.

If an error occurs and ERROR was not specified, the dynamic support program (DSP) will be ended using the FAILDSP macro.

NORMAL=
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

Return Codes
Following are the return codes and dump codes returned in registers 15 and 0, respectively:

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Dump Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>04</td>
<td>700</td>
<td>The FDB is already open and in the file directory, or the FDBDATA field is zero.</td>
</tr>
<tr>
<td>10</td>
<td>708</td>
<td>No job TAT was provided for an output MRF.</td>
</tr>
<tr>
<td>14</td>
<td>737</td>
<td>STT allocation requested on local or by an FSS.</td>
</tr>
</tbody>
</table>

Linkage
This macro uses \texttt{ASAVE} linkage.

Restrictions
Do not use this macro in any routine that might run under the JES3 auxiliary task.

\textbf{AOPEND (Open a Multirecord Output File)}

The AOPEND macro opens an existing multirecord file (MRF) for the later addition of more records at the end-of-file (EOF).

Syntax

\begin{verbatim}
  AOPEND FDB=(R1) address, TATPTR=(R0) address
\end{verbatim}
Parameters

FDB=
Specifies the address of the FDB of the MRF to be opened.

TATPTR=
Specifies the address of the track allocation table (TAT) FDB. If omitted, the TAT from the caller's resident job queue (RESQUEUE) is used.

NOEOD=

YES
Specifies that the MRF does not have an EOF indicator. The record pointer is set to the last record in the buffer.

NO
Specifies that the multirecord file has an EOF indicator.

NORMAL=
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

ERROR=
This parameter specifies the address to which the macro processing routine returns if an error occurs. When control is passed to the address, register 0 contains a dump code and register 15 contains a return code, unless otherwise specified. The return codes and dump codes are described below. For a description of every dump code, see z/OS JES3 Diagnosis.

If an error occurs and ERROR was not specified, the dynamic support program (DSP) will be ended using the FAILDSP macro.

CARR=

ASA
Specifies that the MRF contains ASA carriage control characters.

MAC
Specifies that the MRF contains machine carriage control characters.

OPTCD=
Specifies that the MRF file to be opened contains table reference characters.

Return Codes
Following are the return codes and dump codes returned in registers 15 and 0, respectively:

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Dump Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>04</td>
<td>715</td>
<td>FDB not closed</td>
</tr>
<tr>
<td>08</td>
<td>nnn</td>
<td>Failure on internal call (AOPEN, INPUT)</td>
</tr>
</tbody>
</table>
AOPEND

<table>
<thead>
<tr>
<th>Return Code In Hex</th>
<th>Dump Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0C</td>
<td>716</td>
<td>No end of data (EOD) in last buffer, and:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1. NOEOD=NO was specified or,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2. NOEOD=YES was specified, but the AOPEND macro was unable to position the record pointer correctly.</td>
</tr>
<tr>
<td>10</td>
<td>716</td>
<td>No EOD in last buffer, but NOEOD=YES was specified and the record pointer has been correctly positioned.</td>
</tr>
</tbody>
</table>

Linkage

This macro uses ASAVE linkage.

Restrictions

Do not use this macro in any routine that might run under the JES3 auxiliary task.

APOINT (Reposition in Multirecord File)

The APOINT macro opens and repositions a multirecord file (MRF) to the location that the user noted with an ANOTE macro.

Syntax

```
APOINT label FDB=(R1),POINTER=(R0) address,NORMAL=address,ERROR=address
```

Parameters

**FDB=**

Specifies the address of the file description block (FDB) for the MRF to be repositioned.

**POINTER=**

Specifies the address of a 3-word field. The first two words must contain the information returned by the ANOTE macro. The third word should be set to zeros if the file being opened is to become an input file. If the file is being opened for output, then the third word must contain the address of the FDB for the track allocation table (TAT) associated with the data set.

**NORMAL=**

This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

**ERROR=**

This parameter specifies the address to which the macro processing routine returns if an error occurs. When control is passed to the address, register 0 contains a dump code and register 15 contains a return code, unless otherwise specified. The return codes and dump codes are described below. For a description of every dump code, see [z/OS JES3 Diagnosis](#).

If an error occurs and ERROR was not specified, the dynamic support program (DSP) will be ended using the FAILDSP macro.
Return Codes
Following are the return codes and dump codes returned in registers 15 and 0, respectively:

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Dump Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>04</td>
<td>701</td>
<td>FDB not in the file directory</td>
</tr>
</tbody>
</table>

Linkage
This macro uses ASAVE linkage.

Restrictions
This macro must not reference an opened file. Do not use this macro in any routine that runs under the JES3 auxiliary task.

APURGE (Purge a Single Record File)
The APURGE macro purges a single record file (SRF) from the single track table (STT) or job track allocation table (job TAT) FDB.

Syntax
```
APURGE FDB=(R1) address, TATSTT=address, NORMAL=address, ERROR=address
```

Parameters
FDB=
Specifies the address of the FDB of the SRF being purged.

TATSTT=
Specifies the address of the FDB.

NORMAL=
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

On return, register 0 contains the number of track group records (M.Rs) purged. Register 1 contains the number of track groups (X.Gs) purged.

ERROR=
This parameter specifies the address to which the macro processing routine returns if an error occurs. When control is passed to the address, register 0 contains a dump code and register 15 contains a return code, unless otherwise specified. The dump codes are described below. For a description of every dump code, see [z/OS JES3 Diagnosis](https://www.ibm.com/support/docview.wss?rs=60&context=CSV&uid=swg21212997).

If an error occurs and ERROR was not specified, the dynamic support program (DSP) will be ended using the FAILDSP macro.
Dump Codes
Following are the dump codes returned in register 0.

<table>
<thead>
<tr>
<th>Dump Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>710</td>
<td>Duplicate track address</td>
</tr>
<tr>
<td>713</td>
<td>Incorrect FDB</td>
</tr>
<tr>
<td>742</td>
<td>Incorrect job TAT</td>
</tr>
<tr>
<td>743</td>
<td>Incorrect track group address (X.G format)</td>
</tr>
</tbody>
</table>

Linkage
This macro uses ASAVE linkage.

Restrictions
Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task.

APUTBUF (Return a JSAM Buffer to the JSAM Buffer Pool)
The APUTBUF macro returns a JSAM buffer, obtained by a previous AGETBUF macro, to the JSAM buffer pool.

Syntax
```
APUTBUF BUFFER=(R0) address, NORMAL=address, ERROR=address
```

Parameters

**BUFFER=**
Specifies the address of the buffer being returned to the buffer pool.

**NORMAL=**
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

**ERROR=**
This parameter specifies the address to which the macro processing routine returns if an error occurs. When control is passed to the address, register 0 contains a dump code and register 15 contains a return code, unless otherwise specified. The return codes and dump codes are described below. For a description of every dump code, see **z/OS JES3 Diagnosis**.

If an error occurs and ERROR was not specified, the dynamic support program (DSP) will be ended using the FAILDSP macro.

Return Codes
Following are the return codes and dump codes returned in registers 15 and 0, respectively:
### Return Code

<table>
<thead>
<tr>
<th>In Hex</th>
<th>Dump Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>04</td>
<td>705</td>
<td>Incorrect buffer address</td>
</tr>
</tbody>
</table>

### Linkage

This macro uses ASAVE linkage.

### Restrictions

Do not use this macro in any routine that might run under the JES3 auxiliary task.

### APUTMAIN (Release a Contiguous Storage Area)

The APUTMAIN macro releases a contiguous area of storage, previously obtained by an AGETMAIN macro.

#### Syntax

```
APUTMAIN label, SIZE = (R0), AREA = (R1), SP = (R2)
```

#### Parameters

**SIZE=**

Specifies the number of bytes of main storage to be returned. If the register form is used, the size of the area to be released must be contained in the register. If the address form is used, it must be the address of a fullword containing the size of the area.

SIZE cannot specify register 1 or 2.

**AREA=**

Specifies the address of the contiguous storage area to be returned. AREA cannot specify register 0 or 2.

**SP=**

Specifies the subpool in which the area resides. If SP is not specified, JESPOOL (defined in IATYTVT) is assumed.

See “Using Storage Subpools” located in [z/OS MVS Programming: Assembler Services Guide](https://www.ibm.com/servers/z/os/mvs/aseg/) If SP is specified as a register, it must contain the subpool number in the low-order byte with zero in the other bytes. SP cannot specify register 0 or 1.

SIZE, AREA, and SP must specify different registers.

### Linkage

This macro uses ASAVE linkage. This macro uses register 2.
Restrictions

Be certain that you do not specify both the AREA= (storage area address) and SIZE= (bytes of storage) as zero.

ARELEASE (Restore Spool Address in FDB)

The ARELEASE macro restores the spool address in the FDB for a single record file (SRF) and returns the JSAM buffer to allow two successive JESREAD macros to be issued.

Syntax

```
label
ARELEASE FDB=(R1) (Rx)
,NORMAL=address
,ERROR=address
```

Parameters

**FDB=**

Specifies the address of the FDB of the SRF.

**NORMAL=**

This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

**ERROR=**

This parameter specifies the address to which the macro processing routine returns if an error occurs. When control is passed to the address, register 0 contains a dump code and register 15 contains a return code, unless otherwise specified. The return codes and dump codes are described below. For a description of every dump code, see ["z/OS JES3 Diagnosis"](https://www.ibm.com/servers/zseries/zos/jes3/diagnosis/).

If an error occurs and ERROR was not specified, the dynamic support program (DSP) will be ended using the FAILDSP macro.

Return Codes

Following are the return codes and dump codes returned in registers 15 and 0, respectively:

<table>
<thead>
<tr>
<th>Return Code In Hex</th>
<th>Dump Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>04</td>
<td>723</td>
<td>Incorrect buffer address</td>
</tr>
<tr>
<td>10</td>
<td>723</td>
<td>Zero FDB</td>
</tr>
<tr>
<td>18</td>
<td>736</td>
<td>CSBT used with a previous JESREAD</td>
</tr>
</tbody>
</table>

Linkage

This macro uses ASAVE linkage.

Restrictions

Do not use this macro in any routine that might run under the JES3 auxiliary task.
ARETURN (Generate a Return Sequence)

The ARETURN macro is used to return to a routine through the ASAVE linkage routine. The ACALL macro must have been issued previously by the routine to which control is being returned to get to the routine issuing the ARETURN macro.

Syntax

```
label
ARETURN
RC=(Rx)
number
TRACE=NO
```

Parameters

**RC=**

Specifies the return code, representing the displacement from the return point for the required macro exit. In other words, the ARETURN macro returns control to your program at the instruction following the ACALL macro, unless you’ve specified a return code value with the RC= parameter. For example, if you specify RC=8, your program receives control at 8 bytes after the ACALL macro.

The value of RC= must be 0 or a multiple of 4. The default is 0.

If the return code is placed in a register, registers 0, 11, 12, or 14 cannot be specified. The return code cannot exceed 64.

**TRACE=**

**YES**

The return will be traced in the JES3 trace table.

**NO**

The return will not be traced in the JES3 trace table.

Linkage

This macro uses ASAVE linkage.

Restrictions

Before issuing the ARETURN macro (or ACALL macro), be sure that register 11 points to the FCT and register 12 points to the TVT.

ATEST (Test Availability of JES3 Resource)

The ATEST macro tests whether a JES3 function has issued the AENQ macro and obtained exclusive or shared use of a JES3 resource.

Syntax

```
label
ATEST
NAME=name
(Rx)
```
Parameters

NAME=

name
Specifies the name of the JES3 resource to be tested. See the IATYRSC macro expansion in the TVT for the valid names.

(Rx)
Specifies a register containing the address of the name of the JES3 resource to be released.

ERROR=
Specifies the address to which the routine returns if the caller does not have the resource.

TYPE=

FCT
Specifies that the address of the FCT that is using the resource is to be returned in register 1. If the resource is not being held by any function, register 1 contains zero on return.

TEST
Specifies that the service routine is to test whether the function issuing this macro holds the resource. If the issuing function holds the resource, control returns to the address specified using the NORMAL parameter. If not, control returns to the address specified using the ERROR parameter.

NORMAL=
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

BUSY=
Specifies that the AWAIT macro should be issued if the resource is held by any function other than the function issuing this macro. Control returns to the instruction following the macro call.

See the section on “Enqueuing and Locks” additional information about this macro and an example of its use.

Linkage

This macro uses ASAVE linkage.

Restrictions

Do not use this macro in any routine that might run under the JES3 auxiliary task.
ATIME (Standard Form: Request Asynchronous Processing)

The standard form of the ATIME macro either requests that an action be done or cancels a previous request. The action can be done:

- After a specified time interval has elapsed
- At a specified time of day

Three types of actions are supported:

ENTER=
Asynchronous entry to a timer appendage routine. The routine is given control after the specified interval has elapsed. JES3 expects the routine to do some processing and then return to the caller. This routine may not issue the ATIME macro.

ECFADD=
An ECF (event control flag) is posted after the specified interval has elapsed.

WAIT=
An FCT (function control table) entry is placed in a wait state until the requested interval has elapsed.

Syntax

```
ATIME label
  TIME=(R0)
  Selections
  Options

Selections:
  ,ENTER=(reg)
  ,FCT=(reg)
  ,YES
  ,WAIT=NO
  ,REUSE=NO
  ,ECFADD=(reg)
         ECFMASK
         TYPE

Options:
  ,AREA=(reg)
  ,ID=name

Return:
  ,BUSY=(reg)
  ,NORMAL=(reg)
```
ATIME

TYPE:

ECFMASK:

Parameters

TIME=

CANCEL

Specifies that a previously requested time interval is to be canceled.

(R0)

number

Specifies the time interval that is to elapse before the requested action takes place. The interval is specified in hundredths of seconds. The maximum interval is 8639999 (24 hours). A zero interval is equal to CANCEL.

TOD=

Specifies the time of day when the requested action is to take place. This value is represented in decimal digits of the form HHMMSSTH where:

- HH is hours (24-hour clock)
- MM is minutes
- SS is seconds
- T is tenths of seconds
- H is hundredths of seconds

If the requested time of day is earlier than the time the request is issued, the interval will expire the following day.

SELECTIONS

ENTER=

Specifies the address of the routine to be entered at the specified time. The contents of the registers on entry to the routine are:

R1

Address of a timer queue element (TQE). The TQE contains the following information:

<table>
<thead>
<tr>
<th>Offset</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>+0</td>
<td>ATIME queue chain field with a value of 0.</td>
</tr>
<tr>
<td>+4</td>
<td>ATIME queue element identifier field with a value equal to the value specified on the ID= parameter. If ID= is not coded, the value is 0.</td>
</tr>
<tr>
<td>+8</td>
<td>ATIME queue element interval of 0.</td>
</tr>
<tr>
<td>+10</td>
<td>ATIME queue element exit address.</td>
</tr>
<tr>
<td>+C</td>
<td>ATIME queue element flag byte with the following possible values set:</td>
</tr>
</tbody>
</table>

- X'40' Timer interval complete
The timer entry area is provided by the user.

The timer entry is within the FCT.

The identifier is in the timer entry.

The location of the TQE is dependent on the following:

- If AREA= is specified on the ATIME macro, the TQE is built at the address specified by the AREA= parameter. If ID= was also specified, the specified ID is inserted at offset +4 of the TQE area.
- If ID= is specified, but AREA= is not, storage available to Timer services is used for the TQE and the identifier specified by the ID= parameter is inserted at offset +4 of the TQE area.
- If neither AREA= or ID= is specified, the TQE is built at the area starting at label FCTTNEXT in the caller's FCT.

R5 Address of the FCT that issued the ATIME macro
R14 Return address
R15 Address of the appendage routine

Either the ENTER= parameter or one of the following parameters or sets of parameters must be coded:

- TIME=CANCEL
- WAIT=YES
- ECFADD= and ECFMASK=

FCT=
Specifies the FCT entry that this ATIME request applies to. If you omit this parameter, it defaults to the calling FCT entry.

WAIT=
If WAIT=YES is specified, the FCT the ATIME macro is issued under is put into a wait state until the specified interval has elapsed. Either WAIT=YES or one of the following sets of parameters must be coded:

- TIME=CANCEL
- ENTER=
- ECFADD= and ECFMASK=

REUSE=
Specifies whether an ATIME queue element is to be reused or not. REUSE=YES indicates that an ATIME queue element is to be reused for this request if the queue element with the same ID (or no ID=specified) exists. REUSE=NO indicates that a new ATIME queue element is always created.

Note: REUSE=keyword is valid only with WAIT=YES.

ECFADD=
Specifies the address of an ECF that is to be posted at the specified time. The ECFMASK= parameter must also be coded. Either the ECFADD= and ECFMASK= parameters or one of the following parameters must be coded:

- TIME=CANCEL
- ENTER=
**ATIME**

- **WAIT=YES**

**ECFMASK=**
Specifies a 1-byte ECF mask that indicates which bits in the corresponding ECF are to be posted. If a *decimal value* (nnn) is specified, it must be a value less than 256. If a *hex expression* is used, the mask must be specified in the form X'hh'. If a *symbol* is used, it must be equated to a one-byte ECF mask value. The symbol is not the address of the ECF mask. The ECFADD= parameter must also be coded. Either the ECFADD= and ECFMASK= parameters or one of the following parameters *must* be coded:
  - **TIME=CANCEL**
  - **ENTER=**
  - **WAIT=YES**

**TYPE**
Specifies whether the ECF mask is to be turned on or off. This parameter can be used only with the ECFADD= and ECFMASK= parameters.

**OPTIONS**

**AREA=**
Specifies the address where the TQE is to be built.

**ID=**
Specifies the name to be associated with the ATIME queue element. The number of characters is limited to four.

**RETURN**

**BUSY=**
Specifies the address to which control is returned if storage is not available to process the timer request. An ECF mask and address are not returned by this parameter. BUSY is ignored if CANCEL is specified.

**NORMAL=**
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

**Linkage**
This macro uses ASAVE linkage.

**Restrictions**
Do not use this macro in any routine that might run under the JES3 auxiliary task.

**Example**
The following example causes a program to wait until the time is 11:30:15.00AM.

```
Setting the Program Wait Time

ATIME WAIT=YES,
    ID='STRT',
    TOD=STRTTIME

STRTTIME DC XL4'11301500'
```
ATRACK (Allocate a Track Group)

The ATRACK macro allocates a track group to a job track allocation table (TAT) or data set TAT for all spool activity. A previously allocated track group that was only used in part by the job or data set TAT may be used to satisfy this request.

The address of the first record in the allocated track group is returned to the caller in field FDBJTSPA of the job or data set TAT FDB. Field FDBRECCT contains the number of available records in the track group. Field FDBSPNDX of the job or data set TAT identifies the spool partition from which the spool space will be obtained.

Syntax

```
ATRACK FDB=address, NAVAIL=address, ERROR=address, BUSY=address, MINTR=NO, CKPT=YES, NORMAL=address
```

Parameters

**FDB=**
Specifies the address of the FDB for the job TAT or data set TAT where the track group allocation is to be recorded. The first record in the job or data set TAT (data area IATYJBT) must be in main storage; all subsequent records must be on spool.

**NAVAIL=**
Specifies the address to which control returns if the track group cannot be allocated because of a shortage of spool space. The caller can wait for the shortage to be relieved using AWAIT TYPE=OFF. The ECF address needed is returned in register 0. The ECF mask needed is contained in the low-order byte of register 1, with the other three bytes containing zeros.

**ERROR=**
Specifies the address to which control returns if an error occurs while processing the request.

**BUSY=**
Specifies the address to which the routine returns if the track group allocation cannot be performed at this time or if the CKPT=YES parameter is specified. The ECF address needed is returned in register 0. The ECF mask needed is contained in the low-order byte of register 1, with the other three bytes containing zeros.

**MINTR=**
Specifies whether a track group should be allocated if the spool partition from which the track group is requested is in a “minimal” spool space condition. If you specify MINTR=NO and the spool partition is in a “minimal” spool space condition, control returns to the address specified by the NAVAIL parameter.
ATRACK

CKPT=
Specifies whether the job TAT or data set TAT should be updated on spool with
the new track allocation information upon completion of the track group
allocation request.

If CKPT=YES is specified, the ATRACK service routine returns control to the
address specified by the BUSY parameter. The caller should issue the AWAIT
TYPE=ON macro to wait for checkpoint processing to complete. When the ECF
upon which the AWAIT macro is waiting is posted, the caller must reissue the
ATRACK macro to verify that the checkpoint processing did complete. The
ATRACK macro must also verify that no other JES3 function used the allocated
track group.

If CKPT=NO is specified, the function that issued this macro must update the
TAT on spool before using the allocated track group.

NORMAL=
This parameter specifies that after the macro processing routine executes, the
issuing program can resume execution at an instruction other than the
instruction that immediately follows the macro.

Return Codes
Following are the return codes and dump codes returned in registers 15 and 0,
respectively:

<table>
<thead>
<tr>
<th>Return Code In Hex</th>
<th>Dump Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>04</td>
<td>713</td>
<td>Incorrect FDB.</td>
</tr>
<tr>
<td>04</td>
<td>721</td>
<td>Spool I/O error when checkpointing the TAT.</td>
</tr>
<tr>
<td>04</td>
<td>742</td>
<td>Incorrect job TAT or data set TAT.</td>
</tr>
<tr>
<td>04</td>
<td>744</td>
<td>Incorrect spool partition</td>
</tr>
</tbody>
</table>

Linkage
This macro uses ASAVE linkage.

Restrictions
Routines running in a C/I FSS address space can issue this macro only to allocate
spool space to job TATs and data set TATs created by the JES3 global address
space. Do not use this macro in any routine that might run under the JES3 auxiliary
task.

AWAIT (Await Completion of One or More Events)
The AWAIT macro allows you to wait for the occurrence of one or more events. See
section “JES3 Function Synchronization” on page 28 for a description of these
events.

Syntax

```
label AWAIT TYPE=OFF

```

z/OS V1R11.0 JES3 Customization
Parameters

TYPE=

**ON**
Specifies that you want to wait for one of the ECF bits indicated by the mask to be turned on.

**OFF**
Specifies that you want to wait for all of the ECF bits indicated by the mask to be turned off.

ECFMASK=
Specifies a one-byte mask indicating which bits in the ECF are to be waited on. This parameter can be specified as a decimal value, a hex value, a symbol, or as a register. If a decimal value (nnn) is specified, it must be a value less than 256. If a hex expression is used, the mask must be specified in the form X’hh’. If a symbol is used, it must be equated to a one-byte ECF mask value. The symbol is not the address of the ECF mask. If a register is specified, the register must contain the one-byte mask value. The remainder of the register must be zero. If R1, R14 or R15 are used, the input value will be changed on return from the macro.

ECFADD=
Specifies the address of the one-byte ECF to be waited on. If coded, then the ECFMASK parameter must also be coded. ECFADD cannot specify the same register as ECFMASK.

ECFLIST=
Specifies the address of an event completion (ECF) list. The list contains one or more two-word entries plus a one-word list terminator. For each entry, the first word contains the 31-bit address of the ECF and the second word contains the ECF mask in the low-order byte. The other three bytes must be zero. The list terminator is a fullword of binary ones.

DATA=
Specifies 1 to 16 bytes of data associated with the AWAIT. For example, if you want to wait for a read I/O request to complete for the JST, you can specify DATA=’JST’.

You specify the DATA= keyword as follows:

DATA=label
The data is at the location identified by ”label”. The length of the data is the length of the filed ”label”.

Chapter 2. JES3 Macros 219
AWAIT

DATA='string'
   The data is the specified character string. The character string should not
   contain imbedded quotation marks or ampersands.

DATA=((register),len)
   The data is pointed to by the specified register. The length of the data is
   specified by the second subparameter.

DATA=((register),INREG)
   The data is contained within the register. The length of the data is assumed
   to be four bytes.

REASON=
   Specifies the reason why the FCT is waiting. The reason codes are
   documented in [z/OS JES3 Diagnosis Reference]. If you want to add a new
   reason code, then use the IATYAWR macro to make the update.

Note: ECFMASK cannot specify register 0. Both ECFADD and ECFLIST cannot
   specify register 1. REASON cannot specify register 0 or register 1. All other
   registers are allowed.

Linkage

This macro issues a BALR R14,R15 for linkage.

Restrictions

When issued by a routine running under the auxiliary task, this macro can be used
only to wait for events that result in the auxiliary task being posted.

AWRITE (Write a Single-Record File)

The AWRITE macro writes a single record file (SRF) to the spool data set.

Syntax

```
AWRITE FDB=(R1) address,PUTBUF=NO
,TATPTR=(R0) address,DISP=NO,BUSY=address
,AWAIT=(NO),NORMAL=address,ERROR=address
```

Parameters

FDB=
   Specifies the address of the FDB for the SRF to be written.

PUTBUF=
YES
Specifies that the buffer is to be returned to the buffer pool.

NO
Specifies that the buffer is not to be returned to the buffer pool.

TATPTR=
Specifies the address of the track allocation table FDB. To write an SRF into the single track table (STT), code TATPTR=MNTRKFDB.

If you do not specify the TATPTR parameter, JES3 will use the TAT FDB from the Resqueue as pointed to by the FCT (FCTRQAD). Make sure that the FCT is associated with the job whose control block is being written, even if the AWRITE is updating an already existing record on spool.

In case of a recoverable I/O error, the TAT FDB will be used to find another record on spool. If the TAT is from a different job, the spool space will be purged independent of the current job which may lead to spool overlays.

If spool data integrity checking is enabled, JES3 will verify all update WRITEs for all JES3 control blocks always specify the same TAT that was used to first allocate the record. If a mismatch is found the AWRITE will report a DM760 abend.

ID=
Specifies the name to be associated with the SRF. The number of characters is limited to four. ID is compared to the ID field in the buffer before the write. If they are not the same, the error exit is taken.

ROOT=
address
Specifies the address of a master FDB whose buffer contains the FDB being written; the DISP parameter is required.

RQ
Specifies that the FDB is from the resident job queue (RESQUEUE); the DISP parameter is not applicable.

DISP=
Specifies the displacement, within the master buffer, of the FDB that points to the next buffer (the “chain FDB”).

AWAIT=

YES
Specifies that AWRITE should wait for the I/O to complete.

NO
Specifies that AWRITE should not wait for I/O to complete. When I/O is started, AWRITE will return to the normal return. The caller can then continue or can wait for the I/O to complete. The ECF address needed is returned in register 0. The ECF mask needed is returned in the low-order byte of register 1, and the address is returned in register 0. If AWAIT=NO is specified, then the BUSY parameter must also be specified.

BUSY=
Specifies the address to which control returns if I/O is not possible. The caller can wait for the condition preventing I/O to clear using AWAIT TYPE=ON. When the wait is satisfied, the caller can reissue the AWRITE macro. The ECF address needed is returned in register 0. The ECF mask
AWRITE

needed is contained in the low-order byte of register 1, with the other three bytes containing zeros. If BUSY is coded, AWAIT=NO must also be coded.

NORMAL=
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

ERROR=
This parameter specifies the address to which the macro processing routine returns if an error occurs. When control is passed to the address, register 0 contains a dump code and register 15 contains a return code, unless otherwise specified. The return codes and dump codes are described below. For a description of every dump code, see z/OS JES3 Diagnosis.

If an error occurs and ERROR was not specified, the dynamic support program (DSP) will be ended by the FAILDSP macro.

Return Codes
Following are the return codes and dump codes returned in registers 15 and 0, respectively:

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Dump Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>04</td>
<td>700</td>
<td>The FDB address for the file already exists in the file directory, or the FDBDATA field is zero.</td>
</tr>
<tr>
<td>08</td>
<td>703</td>
<td>Incorrect buffer address: the flag FDBONSP is on.</td>
</tr>
<tr>
<td>0C</td>
<td>721</td>
<td>Incorrect buffer or spool address.</td>
</tr>
<tr>
<td>10</td>
<td>724</td>
<td>Recovered I/O error should checkpoint FDB.</td>
</tr>
<tr>
<td>14</td>
<td>725</td>
<td>Unrecoverable I/O error.</td>
</tr>
<tr>
<td>18</td>
<td>725</td>
<td>Error during ROOT I/O error recovery.</td>
</tr>
<tr>
<td>1C</td>
<td>703</td>
<td>Zero FDB.</td>
</tr>
<tr>
<td>20</td>
<td>703</td>
<td>IDs did not match.</td>
</tr>
<tr>
<td>24</td>
<td>708</td>
<td>No job TAT was provided for a new SRF.</td>
</tr>
<tr>
<td>28</td>
<td>729</td>
<td>Spool space could not be allocated to a new single record file (SRF) because of a job or data set TAT error.</td>
</tr>
<tr>
<td>2C</td>
<td>737</td>
<td>Request on a local processor.</td>
</tr>
<tr>
<td>34</td>
<td>703</td>
<td>Request for a multirecord file (flag FDBMULT is on)</td>
</tr>
<tr>
<td>44</td>
<td>736</td>
<td>CSBT used with a previous JESREAD.</td>
</tr>
</tbody>
</table>

Linkage
This macro uses ASAVE linkage.

Restrictions
Do not use this macro in any routine that might run under the JES3 auxiliary task.

Do not use this macro in any routine that writes OSEs to spool. Instead, use the WRITEOSE macro.

CONCNVRT (Convert Console Destination Class)
The CONCNVRT macro converts a console destination class to a displacement and mask suitable for use in the MESSAGE macro.
On return, register 0 contains the displacement and mask in the low-order 2 bytes. If the value returned is zero, the class name was invalid.

**Syntax**

```
CONCNVRT CLASS=(R0) address NORMAL=address
```

**Parameters**

- **CLASS=**
  Specifies the address of the destination class name.

- **NORMAL=**
  This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

**Linkage**

This macro uses ASAVE linkage.

**Restrictions**

Do not use this macro in any routine that might run under the JES3 auxiliary task.

---

**CONREVRT (Convert Code to Console Destination)**

The CONREVRT macro converts a 2-byte code obtained from the MESSAGE macro into a console destination class name.

The name is returned left-justified in register 1.

**Syntax**

```
CONREVRT CLASS=(R1) address NORMAL=address
```

**Parameters**

- **CLASS=**
  Specifies that the two-byte destination code from the MESSAGE macro is to be loaded into the low-order two bytes of register 1 or into the address specified.

- **NORMAL=**
  This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

**Linkage**

This macro uses ASAVE linkage.

**Restrictions**

Do not use this macro in any routine that might run under the JES3 auxiliary task.
CONSPROF (Set Console Profile Mask in FCT)

The CONSPROF macro sets the console profile mask in the FCT for a DSP.

Syntax

```
<table>
<thead>
<tr>
<th>Set the Profile Mask</th>
</tr>
</thead>
<tbody>
<tr>
<td>QMSG= 1</td>
</tr>
</tbody>
</table>
```

Parameters

QMSG=
Specifies the number of messages to be queued from the FCT before the 'MESSAGE ENQUEUED' message is issued from console services. If any number other than 1 or 2 is specified, 1 is assumed and the specified number is ignored.

Linkage

This macro expands as inline code.

Restrictions

Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task.

DEQMSG (Dequeue Message Buffers from FCT)

The DEQMSG macro deletes operator action messages and removes input message buffers queued from an FCT.

Syntax

```
<table>
<thead>
<tr>
<th>DEQMSG</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTION</td>
</tr>
<tr>
<td>BUFFER=address</td>
</tr>
<tr>
<td>FCT=reg</td>
</tr>
<tr>
<td>NORMAL=address</td>
</tr>
</tbody>
</table>
```

Parameters

TYPE=
Specifies the type of message to be processed.

ACTION
Delete action messages that have been retained. These are messages that
were issued as either ACTION=YES, ACTION=CRITICAL, or with descriptor code 1, 2, 3, or 11 on the MESSAGE macro. Specifying TYPE=ACTION generates a DOM request.

**INPUT**

Dequeue input messages.

**ALL**

Used by the job segment scheduler (JSS) to ensure removal of all queued messages when a DSP has completed processing. Specifying TYPE=ALL generates a DOM request.

**Note:** There is no correspondence between the TYPE=ALL parameter and TOKEN= parameter on the DOM macro.

**BUFFER=**

Specifies the address of the buffer to be dequeued. If omitted, all buffers of the specified type are dequeued. This parameter is not valid with TYPE=ALL.

If the buffer is for an action message, then this address is that of the DOM ID.

**FCT=**

Specifies the address of the FCT from which to dequeue the buffers. If omitted, the active FCT is assumed. Register 1 may not be specified.

**NORMAL=**

This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

### Linkage

This macro uses ASAVE linkage.

### Restrictions

Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task. The contents of register 2 are destroyed by this macro. Do not use register 2 as a parameter register on this macro.

### DEVSCAN (Analyze Operands of IN and OUT Keyword Parameters)

The DEVSCAN macro analyzes the operands of the IN and OUT keyword parameters allowed by the callable DSPs and returns the parameters in a 16-byte field.

Upon return, if the parameter was enclosed in parentheses, the first field is in the first 8 bytes of TO, and the second field is in the second 8 bytes. If a parameter was omitted, its corresponding field is set to zero. If the parameter was not enclosed within parentheses, the first 8 bytes of TO are zero and the second 8 bytes contain the parameter field. Register 1 contains the address of the byte following the delimiter which ended the scan.

### Syntax

```
label: DEVSCAN FROM=(R0) address, TO=(R1) address
```
Parameters

FROM=
Specifies the address at which the scan is to begin; that is, the byte following the equal sign of IN= or OUT=.

TO=
Specifies the address of the 16-byte field to which the parameters are to be moved.

ERROR=
Specifies the address to which the routine returns if the scan cannot be completed. Register 15 contains one of the following error codes in hex:
- 04 – Length error
- 08 – Delimiter error
- 0C – No operand

NORMAL=
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

Linkage

This macro issues a BALR R14,R15 for linkage.

Restrictions

Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task.

DLOCOFF (Mark a Destination Queue Entry Inactive)

The DLOCOFF macro marks a function’s queue entry as inactive in the destination routing table or, for a functional subsystem (FSS) or functional subsystem application (FSA), in the dynamic destination queue table. This action prevents staging areas from accumulating in CSA for a function that is not active and thus unable to receive them. (Staging areas are unsolicited communications from a user address space, from JES3 on another system in the processor complex, or from a FSS or FSA in another address space.)

Syntax

```
DLOCOFF DEST=DEST

Parameters

DEST=
Specifies the name equated with the JES3 destination code of the queue entry to be marked inactive.
FSID=
Specifies the fullword identifier (or the register containing the address of the fullword identifier) of the FSS or FSA associated with the dynamic destination queue entry to be marked inactive. For an FSS, you can find the identifier in field FSSFSID of the IATYFSS control block. For an FSA, you can find the identifier in field FSAFSID of the IATYFSA control block.

This parameter is valid only when the DEST parameter specifies a dynamic destination queue entry.

Linkage

This macro expands as inline code.

Restrictions

Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task.

DLOCON (Mark a Destination Queue Entry Active)

The DLOCON macro marks a function’s queue entry as active in the destination routing table or, for a functional subsystem (FSS) or functional subsystem application (FSA), in the dynamic destination queue table. Issuing this macro allows a function in the JES3 global or local address space to receive unsolicited communication (staging areas) from a user’s address space, from JES3 on another processor in the processor complex, or from an FSS or FSA in another address space.

Syntax

```
Mark a Function’s Queue Entry as Active

DEST= name (reg), ECF= (reg), FSID= fsid address, MDS= (reg)
```

Mark a Function’s Queue Entry as Active

```
label DLOCON
```

Parameters

DEST=
Specifies the name equated with the destination code of the queue entry to be marked active.

FSID=
Specifies the fullword identifier (or the register containing the address of the fullword identifier) of the FSS or FSA for which the destination queue entry is to be initialized. For an FSS, you can find the identifier in field FSSFSID of the IATYFSS control block. For an FSA, you can find the identifier in field FSAFSID of the IATYFSA control block.
DLOCON

This parameter is valid only when the DEST parameter specifies a dynamic destination queue entry.

ECF=
Specifies the address of a doubleword containing the ECF address in word 1 and the ECF mask in the low-order byte of word 2. The first three bytes of word 2 must be zero.

MDS
Specifies that this is a main device scheduling (MDS) queue and should be flagged as such.

Linkage

This macro expands as in-line code.

Restrictions

Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task.

DSQLOC (Locate Destination Queue Entry)

The DSQLOC macro locates a queue entry or the first staging area in the destination routing table or dynamic destination queue table for a specified destination code.

On return, register 1 contains the address of the first staging area chained to the queue for the destination code, unless the macro specified the LA parameter. (If the destination queue is empty, register 1 contains 0.) If the LA parameter is specified, register 1 contains the address of the queue header for the specified destination code, as modified by the OFFSET parameter.

Syntax

```
DSQLOC label
   DEST=(R1) name
   FSID=(reg) fsid
   OFFSET=number
   SVT=YES
   LA
```

Parameters

DEST=
Specifies the name equated with the destination code of the queue entry to be located.

FSID=
Specifies the fullword identifier (or the register containing the address of the fullword identifier) of the functional subsystem (FSS) or functional subsystem application (FSA) whose dynamic destination queue entry is being located. This parameter is valid only when the DEST parameter specifies a dynamic destination queue entry.

OFFSET=
Specifies the offset to be added to the address of the queue entry that is to be
returned in register 1. An offset of 0 leaves register 1 pointing to the queue entry; an offset of 4 results in register 1 pointing to the destination queue header. This parameter is valid only with the LA parameter.

SVT=

YES

Specifies that the subsystem vector table (SSVT) is to be used to address the destination routing table.

NO

Specifies that the TVT is to be used to address the destination routing table.

LA

Specifies that the address of the queue header for the destination specified by the DEST parameter be returned in register 1.

Linkage

This macro expands as inline code.

Restrictions

Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task.

This macro uses labels from the IATYDSQ mapping macro instead of hard-coded displacements. Therefore, if you use the DSQLOC macro in your code, you must also use the IATYDSQ mapping macro.

FAILDSP (Fail the Specified DSP)

The FAILDSP macro allows you to enter a JES3 failsoft routine and fail the specified DSP, with a failure code, upon recognition of an error.

Syntax

```
/SM590000/SM590000
FAILDSP
label
CODE=(R1)
   (reg)
DUMP=number
   (reg)
   FCT
   (R0)

,RDUMP=YES
   NO
   (reg)

,REASON=hex
   symbol
   number
   (reg)

,ASID=(reg)
    address
,MPC=(reg)
    address

,NORMAL=address
```

Parameters

CODE=

Specifies the failure code. If this subparameter is specified, then the IATYDUM mapping macro must be coded elsewhere in the module and it must be assembled before the assembly of the FAILDSP macro.
FAILDSP

(R1)
(reg)
Specifies that the failure code is contained in the indicated register.

FCT=
Specifies the address of the FCT to be failed. If omitted the active FCT is assumed.

DUMP=

YES
Specifies whether a JES3 abend dump is to be taken.

NO
If the register form is coded, then the same register must be coded as for the FCT parameter.

(reg)
If the contents of the register are positive, a dump will be taken. If the contents of the register are negative, it is assumed to contain the complement of the FCT address, and no dump will be taken.

REASON=
Specifies a reason code that you associate with the failure. You can specify the reason code as a hex value (hexexp), decimal value (nnn), a symbol or a register. If you use a decimal value, it must be a value less than 256.

ASID=
Specifies the address space identifier (ASID) of an address space which is to be included in the dump, if one is taken. If the specified address space is not executing on the current processor, the MPC= parameter should be used to identify the processor.

MPC=
Specifies the MPC address of the processor where the address space specified by the ASID= parameter is executing. If MPC= is omitted, the current processor is assumed. The ASID= parameter is required if the MPC= parameter is specified.

NORMAL=
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

Linkage
This macro uses ASAVE linkage.

Restrictions
This macro may be used under the JES3 auxiliary task only to end the currently active FCT running under the auxiliary task.

GETUNIT (Obtain One or More Devices)
The GETUNIT macro requests one or more JES3-managed devices. On return, register 1 contains the address of the satisfied sublist or contains error information.
Syntax

```
GETUNIT label, NAVAIL = (reg) address, ERROR = (reg) address
AMBIG = (reg) address, LIST = (R1) address
VALIDATE = NO, ALLO = YES
SUPUNIT = (R0) address, TYPE = FSS, NORMAL = (reg) address
```

Parameters

**NAVAIL=**
Specifies the address to which the routine returns if none of the sublists can be allocated. Register 1 contains the address of the first unavailable device in the last examined sublist entry of the list.

**ERROR=**
Specifies the address to which the routine returns when the last sublist contains a request for a device that does not exist in the support units (SUPUNITS) table and no previous sublist could be satisfied. Register 1 contains the address of the entry in error in the last examined sublist entry.

**AMBIG=**
Specifies the address to which the routine returns when a device number in the list (specified using the LIST parameter) is ambiguous. A device number is ambiguous when JES3 devices on different processors have the same device number. Upon return, register 1 contains the address of the SUPUNITS (mapped by macro IATYSUP) entry with the ambiguous device number. To issue an error message, use the IATXAMDV macro.

**LIST=**
Specifies the address of a list containing sublists in the format of the GETUNIT list. If omitted, the GETUNIT list whose address is in the FCT is assumed. Each sublist contains the 8-byte name fields with each sublist ended by a word containing X'FF'. The total list is ended by a word containing X'7F'.

**ALLO=**

**YES**
Specifies that the devices are to be allocated.

**NO**
Specifies that the routine is to check the validity of a unit request; no
devices will be allocated. The requesting function must clear the SUPUNITS table address in the list before relinquishing control using an AWAIT, RETURN, or calling another routine.

**VALIDATE=**

**YES**
Specifies that only the destination is being validated. No enqueuing on any resource is performed which provides a fastpath through destination validation.

If you specify VALIDATE=YES, then you must specify ALLO=NO. If you specify ALLO=YES with VALIDATE=YES, you will receive assembly errors. If you do not explicitly specify the ALLO= keyword with VALIDATE=YES, then ALLO=NO will be defaulted.

**NO**
Specifies that more than just the destination is being validated. VALIDATE=NO is the default.

**SUPUNIT=**
Specifies the address of the SUPUNITS table entry already found by other means. This allows a shorter path through the routine.

**TYPE=**
Specifies that the device not be allocated but be marked as being held by an output writer FSS.

**NORMAL=**
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

**Linkage**
This macro uses ASAVE linkage.

**Restrictions**
Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task.

**IATXADD (Add a Slot to the Active DJST Chain)**
The IATXADD macro initializes the active dynamic job summary table (DJST) when a new DJST buffer is obtained, puts the first slot on the active chain, and builds the step entry into that slot. This macro also creates and initializes the available chain. It creates the chain by chaining the first available slot from the header and all other available slots from the one immediately preceding it. Upon return, register 1 contains a pointer to the active slot.

**Syntax**

```
IATXADD label(DJST=(R0),STEP=(R1),ERROR=address)
```

GETUNIT
Parameters

**DJST=**
Specifies the address of the DJST which is mapped by macro IATYDJS (RQDYJFDB in IATYRSQ) (JSTDCHN in IATYJST).

**STEP=**
Specifies the address of the step entry to be built into the first slot.

**ERROR=**
This parameter specifies the address to which the macro processing routine returns if an error occurs.

If an error occurs and ERROR was not specified, the dynamic support program (DSP) will be ended using the FAILDSP macro.

Linkage

This macro issues the IATZCALL macro without using ASAVE linkage.

Restrictions

Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task.

IATXAMDV (Display an Ambiguous Device Number)

The IATXAMDV macro displays a message when a device is referenced using its device number and more than one device has that number. This type of device number is known as an ambiguous device number.

Syntax

```
label IATXAMDV SUPUNIT=(R1) address, CNDB=(Rx) address, NORMAL=address
```

Parameters

**SUPUNIT=**
Specifies the address of the SUPUNITS table entry containing the ambiguous device number.

**CNDB=**
Specifies the address of a Console Destination Block (IATYCNDB). The CNDB contains the destination information for the message.

This parameter can be specified as an address or as a register. If the register form is used, the register must contain the address of a CNDB area mapped by IATYCNDB.

**NORMAL=**
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.
Linkage

This macro uses ASAVE linkage.

Restrictions

Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task.

IATXARL (Build, Scan, or Delete an Allocation Requirements List)

The IATXARL macro builds, scans, or deletes an allocation requirements list (ARL).

Syntax

```
label IATXARL TYPE=TYPE=ADD Options, RQ=(R1) TYPE=SCAN Options, address
```

TYPE=ADD Options:

```
ADD, NAVAIL=(Rx) (address)
```

TYPE=SCAN Options:

```
SCAN, MPC=(R2) (address), BUSY=(Rx) (address), ALLBUSY=(Ry) (address)
```

Parameters

**TYPE=**

Specifies the action to be performed.

**ADD**

Builds and refreshes an ARL following an allocation failure

**SCAN**

Performs pre-allocation ARL scan

**DELETE**

Deletes the ARL

**RQ=**

Specifies the address of the RESQUEUE of the job for which ARL access is requested. This parameter is required, and must not use register 0. If RQ=addr is used, it must be the address of the field containing the RQ address.

**TYPE=SCAN Options**

**MPC=**

Specifies the address of the main processor control table (MPC) for the processor on which allocation is being considered. This parameter can only be used if you specify TYPE=SCAN, and is required if you specify TYPE=SCAN. If specified as a register, MPC= cannot be R0 or R1, or the same register as RQ=. If MPC=addr is used, it must be the address of the field containing the MPC address.
BUSY=
  Specifies the address to return to if the job should not be given an allocation attempt on the requested processor. This parameter is required if you specify TYPE=SCAN.

ALLBUSY=
  Specifies the address to return to if the job should not be given an allocation attempt on any processor. This parameter is required if you specify TYPE=SCAN.

TYPE=ADD Options

NAVAIL=
  Specifies the address to return to if storage for an ARL cannot be obtained. This parameter is required if you specify TYPE=ADD.

Linkage
  This macro uses ASAVE linkage.

Restrictions
  Do not use this macro in any routine that might run under the JES3 auxiliary task.

IATXARQ (Obtain, Release, or Query Resqueue Access)
  The IATXARQ macro is used to obtain access to a resqueue, release access from a resqueue, and query resqueue access (for example, is the resqueue currently being accessed with TYPE=OBTAIN).

Syntax

```
label IATXARQ TYPE=OBTAIN RELEASE QUERY,BUSY=(address)

RQ=reg address
```

Parameters

TYPE=
  Specifies whether resqueue access should be obtained, released, or queried.

  **OBTAIN**
    Requests that resqueue access should be obtained.

  **RELEASE**
    Requests that resqueue access should be released.

  **QUERY**
    Requests that resqueue access should be queried.

BUSY=
  Specifies the address where control should be given if the resqueue is in use (the resqueue is currently accessed through TYPE=OBTAIN). This parameter is valid only for TYPE=QUERY. If the resqueue is not in use, control is passed to the next sequential instruction.
IATXARQ

RQ=
    Specifies the register or field that contains the job’s resqueue address.

Linkage

This macro expands as inline code.

Restrictions

When this macro is used, the IATYRSQ mapping macro must be included in the module.

IATXASR (Obtain or return an OSE or WOSE buffer)

The IATXASR macro obtains a JSAM buffer for an OSE or WOSE, or returns an OSE or WOSE spool record, using the job’s ASR (available spool records) list.

Syntax

```plaintext
IATXASR FUNCTION=label, RQ=(R1), (R3-R9), address
```

Parameters

FUNCTION=
    Specifies the function to be performed.

    GETREC
        Obtains a JSAM buffer for an OSE or WOSE.
        If the job has an available spool record (ASR) chain and there are any ASR entries indicating spool records that are available for reuse, an available spool address will be assigned and stored in the OSE or WOSE FDB.

    RTNREC
        Returns an OSE or WOSE spool record.
        If the spool record was allocated from the single track table (STT), the spool record is purged. Otherwise, if the owning job has an available spool record (ASR) chain, the spool record address is added to a new ASR entry, allowing the spool record to be reused later.

    RQ=
        Specifies the address of the resident job queue (RESQUEUE) entry for the job owning the OSE or WOSE.

    FDB=
        Specifies the address of the file description block (FDB) that points to the OSE or WOSE record.

Linkage

This macro uses ASAVE linkage.
Restrictions

Do not use this macro in any routine that might run under the JES3 auxiliary task.

IATXATDE (ATTACH/DETACH an ATDE)

The IATXATDE macro creates or deletes an auxiliary task dispatching element (ATDE) for the caller’s FCT. The ATDE is the primary control block used by the multifunction monitor (MFM) to dispatch an FCT under the JES3 auxiliary (IATAUX) task. Once an ATDE is created for the FCT, the DSP running under the FCT may subsequently use the IATXSTMD macro to switch dispatching of its FCT between the JES3 primary (IATNUC) task and the JES3 auxiliary task. This macro uses register 0 as a parameter register.

If a DSP ends either normally or abnormally without having detached an ATDE associated with its FCT, JES3 DSP termination processing detaches the ATDE.

Syntax

```
label IATXATDE TYPE=ATTACH/DETACH
```

Parameters

**TYPE=**

Specifies the type of function requested.

**ATTACH**

Specifies that an ATDE is to be created for the caller’s FCT and chained to the auxiliary task dispatching queue. If an ATDE already exists for the FCT, control is immediately returned to the caller.

**Note:** Because ATTACH processing executes an AWAIT, the IATXATDE macro must not be used in DSP processing that must maintain control.

**DETACH**

Specifies that the ATDE associated with the caller’s FCT is to be unchained from the auxiliary task dispatching queue and freed. If no ATDE exists for the FCT, control is immediately returned to the caller.

Linkage

This macro uses ASAVE linkage.

Restrictions

An ATDE can be attached only by a DSP running on the JES3 global processor. If IATXATDE is issued by a DSP running on a JES3 local processor, the ATTACH/DETACH ATDE service routine immediately returns control to the DSP without performing the requested function.

Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task. (This macro may be issued only when running under the JES3 primary task.)
**IATXATF**

**IATXATF (Attach FCT to Active Chain)**

The IATXATF macro attaches an FCT to the active FCT chain in priority sequence.

**Syntax**

```
IATXATF label (R1) FCT = address, NORMAL = address
```

**Parameters**

- **FCT=**
  Specifies the address of the FCT to be attached.

- **NORMAL=**
  This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

**Linkage**

This macro uses ASAVE linkage.

**Restrictions**

Do not use this macro in any routine that might run under the JES3 auxiliary task.

**IATXBDSN (Create Qualified Data Set Names)**

The IATXBDSN macro creates qualified data sets names consisting of multiple qualifiers. The names have the following format:

```
userid.jobname.jobid.dsnumber.dsid
```

Five-part qualified data set names are used to specify the resource name (ENTITY=) on the IATXSEC macro. See "IATXSEC (Security Authorization Services)" on page 429 for information about using the IATXSEC macro to restrict authorization to SYSIN/SYSPUT data sets.

You can code all qualifiers or only some of the qualifiers on each macro invocation. When only some of the qualifiers are specified, they must be adjacent to each other in the data set name.

**Syntax**

```
IATXBDSN label (R1) WORK = (reg) address, USERID = address, JOBNAME = address, JOBID = address, NUMBER = address, NAME = address
```

```
,INENTITY = address
```

---

238  z/OS V1R11.0 JES3 Customization
Parameters

**WORK=**
Specifies the work area where the specified qualifier is to be placed. Use the WORK=addr or WORK=(reg) form of this parameter only for the first issuance of this macro for the specific qualified data set name. JES3 puts the address from the first invocation into register 1. For all of the following issuances, use WORK=(R1). Do not change the value of register 1 between invocations of IATXBDSN when building a specific five-part qualified data set name.

**USERID=**
Specifies the address of the userid that is to be inserted in the qualified data set name.

**JOBNAME=**
Specifies the address of the job name that is to be inserted in the qualified data set name.

**JOBID=**
Specifies the address of the jobid that is to be inserted in the qualified data set name.

**NUMBER=**
Specifies the address of the data set number that is to be inserted in the qualified data set name.

**NAME=**
Specifies the address of the data set identifier that is to be inserted in the qualified data set name.

**INENTITY=**
Specifies the address of a six-part entity name from which the five-part data set name should be extracted.

Linkage

This macro generates in-line code.

Restrictions

None

Example

The following example demonstrates how to use a sequence of two IATXBDSN macro invocations to construct a five part qualified data set name.

<table>
<thead>
<tr>
<th>Sequence of Two IATXBDSN Macros</th>
</tr>
</thead>
<tbody>
<tr>
<td>IATXBDSN WORK=workarea,USERID=userid,JOBNAME=jobname,JOBID=jobid</td>
</tr>
<tr>
<td>IATXBDSN NUMBER=dsnumber,NAME=dsid</td>
</tr>
</tbody>
</table>

IATXBFM (Free Storage in Subpool 231, 241, or 245)

The IATXBFM macro frees an area of storage in subpool 231 (in CSA), 241 (in CSA), or 245 (in SQA). The routine is page fixed and will obtain the SALLOC lock to use the branch entry point to FREEMAIN to release the storage.

Upon return, register 15 contains a return code (see following).
IATXBFM

Syntax

```
label IATXBFM R= C, U, LV= number, SP= number, A= address
          KEY= number
```

Parameters

R=
   C  Specifies a conditional STORAGE RELEASE.
   U  Specifies an unconditional STORAGE RELEASE.

LV=
   Specifies the size, in bytes, of the area to be released.

SP=
   Specifies the subpool in which the storage is contained.

A=
   Specifies the address of the area to be released.

KEY=
   Specifies the key in which the storage was obtained.

Return Codes

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Hex</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>00</td>
<td>Normal, no error</td>
</tr>
<tr>
<td>Other</td>
<td></td>
<td>See STORAGE macro</td>
</tr>
</tbody>
</table>

Linkage

This macro uses the MVS STORAGE macro.

Restrictions

None

IATXBGM (Obtain Storage in Subpool 231, 241, or 245)

The IATXBGM macro obtains an area of storage in subpool 231 (in CSA), 241 (in CSA), or 245 (in SQA).

On return, the address of the storage obtained is in register 1, and a return code is in register 15 (see following).

Syntax

```
label IATXBGM R= C, LV= (reg), SP= number
          , A= address
```
Parameters

**R=**
- \( C \) Specifies a conditional STORAGE obtain.
- \( U \) Specifies an unconditional STORAGE obtain.

**LV=**
Specifies the size, in bytes, of the area to be obtained. \((\text{reg})\) specifies the size is in the register.

**SP=**
Specifies the subpool in which the storage is contained. The number must be 231 or 241 or 245.

**KEY=**
Specifies the key with which the storage is to be obtained.

**BNDRY=**
- \( \text{DBLWD} \)
  Specifies that storage is to be obtained on a double word boundary. \( \text{DBLWD} \) is the default.
- \( \text{PAGE} \)
  Specifies that storage is to be obtained on a page boundary.

**LOC=**
Defines the residence requirements of the storage being obtained. This is as follows:
- \( \text{BELOW} \)
  Obtain storage in virtual storage below 16 megabytes. Pages are backed below 16 megabytes central when fixed.
- \( \text{(BELOW,ANY)} \)
  The same as \( \text{BELOW} \) except that pages may be backed anywhere in central when fixed.
- \( \text{ANY} \)
  Obtain storage from anywhere in virtual storage (normally above 16 megabytes). Pages may be backed anywhere when fixed.
- \( \text{(ANY,ANY)} \)
  The same as \( \text{ANY} \).
- \( \text{RES} \)
  Obtain storage from the same residence as the calling program. If the calling program resides below 16 megabytes, the storage is obtained below 16 megabytes virtual, and central storage is backed below 16 megabytes central when fixed. If the calling program resides above 16 megabytes, the storage is obtained from anywhere in virtual storage and is backed anywhere in central storage when fixed.
(RES,ANY)

The same as RES except storage may be backed anywhere in central storage when fixed regardless of the residence of the calling program.

Return Codes

<table>
<thead>
<tr>
<th>Return Code In Hex</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>Normal, no error</td>
</tr>
<tr>
<td>Other</td>
<td>See STORAGE macro</td>
</tr>
</tbody>
</table>

Linkage

This macro uses the MVS STORAGE macro.

Restrictions

None

IATXBKIO (Access Spool Data Sets Using the Block Spooler Interface)

The IATXBKIO macro provides an interface to the block spooler function. The block spooler enables a program to access spool data sets from a JES3 subtask or other address space that cannot use JSAM services to access spool.

When invoking the block spooler, your program must provide a parameter list indicating which function the block spooler is to do. This macro builds the block spooler parameter list and puts its address in register 1. The parameter list provides the information needed by the service routine IATDMBS, which accesses the spool.

Syntax

```
IATXBKIO

FUNCTION=ACCESS
  TYPE=MULT
  SRF

 STATUS=EOF
  EOM
  NONE

  BLK=(R1) address
  BUFFER=(reg) address

  SADR=(reg) address
  RECLN=(reg) address
  ID=(reg) address

  ASID=(reg) address
  CHAINFDB=(reg) address

  COUNT=(reg) address
  CALL=NO
```
Parameters

FUNC=
Specifies which function of the block spooler you are requesting.

ACCESS
Creates the control blocks needed by the block spooler to access spool.

READ
Specifies a read request that will move the data in the protected buffer (PBUF) to the user’s data area indicated by the BUFFER= parameter.
For a single read request, the block spooler obtains a single PBUF and reads one record from the spool.
For a multiple read request, the block spooler attempts to obtain enough PBUFS to read to the end of the track.

WRITE
Specifies that the block spooler is to write a single record in a multirecord format to the spool.

RELEASE
Specifies that the services of the block spooler are no longer needed or that an end-of-memory condition has occurred.

TYPE=
For read and write requests only. Specifies the type of function being requested.
MULT
A multirecord file request.
SRF
A single record file request.

STATUS=
For release requests only. Specifies the status of the function being requested.
EOM
An end-of-memory request.
EOF
An end-of-file request.
NONE
Resets the status.

BLK=
The address of the block spooler parameter list that this macro builds.

BUFFER=
The address of the user’s data to be written to spool. For the read function, the area is the location where the data is to be placed after being read from the spool.

SPADR=
For read and write requests only. The spool address (M.R) that points to the location on spool where the user data resides or will reside.

RECLN=
For write requests only. This parameter specifies the number of bytes of user data to be written to spool at the address indicated by the SPADR= parameter.

ID=
The 1- to 4-character control block identifier that is located in the single record file header for JES3 single-record spool files.
IATXBKIO

ASID=
For release requests only. The address space ID of the user.

CHAINFDB=
For SRF write requests only. Specifies the address of a chain FDB that is also to be written with the control block.

COUNT=
For write requests only. Specifies the address of a field containing the SRF count value of the buffer being written to spool.

CALL=

YES
Indicates that the block spooler should be invoked.

NO
Indicates that the block spooler should not be invoked until later after certain user-defined events have occurred. The default is CALL=YES.

Linkage
This macro issues a BALR R14,R15 for linkage. The address of an 18-word save area must be in register 13.

Restrictions
The routine that issues this macro must have established addressability to the SSVT. Do not use this macro in any routine that might run under the JES3 auxiliary task.

You must preserve the field BLKACQA in the IATYBLK across invocations of this macro.

Example of Use
The following is an example showing how you would use successive IATXBKIO macros to build a parameter list and invoke the block spooler.

IATXBKIO FUNC=ACCESS
Creates the control blocks needed for the block spooler to access spool.

IATXBKIO SPADR=FSDBRCA,
Sets the specified spool address in the block spooler parameter list.

IATXBKIO FUNC=READ,
Sets the specified function, type of function, and control block id in the block spooler parameter list.

IATXBKIO FUNC=WRITE,
Invokes the block spooler and supplies the address of the user's data to be written back to spool.

IATXBKIO FUNC=RELEASE
Ends the block spooler.

IATXBPL (Create or Extend Cell Pool)
The IATXBPL macro creates or extends a cell pool.
For a description of JES3 common quick cell services (module IATGRQC), see
z/OS JES3 Diagnosis Reference.

Syntax

```
label IATXBPL NORMAL=address,CPADDR=(R1)
      NAVAIL=address,ERROR=address
      PGRLSE=NO,AUTODEL=YES,PGFIX=NO
      SPAN=ANY,LOC=BELOW
```

Parameters

**NORMAL=**
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

**CPADDR=**
If a cell pool is being extended, this parameter specifies the primary cell pool control block address of the cell pool to be extended. The cell pool must have been created by a previous IATXBPL macro.

If a cell pool is being created, this parameter must specify zero. The address of the primary cell pool control block is returned in register 1. The LIST parameter must also be specified when creating a cell pool.

If this parameter is omitted, the IATXBPL service routine assumes that a cell pool is being created.

**NAVAIL=**
Specifies the address to which control is passed if the cell pool could not expand because storage was not available in the user-specified subpool. The service routine returns an ECF mask and address to wait on. This information is passed back in registers 0 and 1. Register 0 contains the 31-bit ECF address. Register 1 contains the ECF mask in the low-order byte. The other three bytes contain zeros.

**ERROR=**
This parameter specifies the address to which the macro processing routine returns if an error occurs. When control is passed to the address, register 15 contains a return code, unless otherwise specified. The return codes are described below.

**ERROR=** is a required keyword.
LIST=
  Specifies the address of the parameter list defining cell characteristics. This
  parameter is required when creating a cell pool. This list is mapped by the
  IATYCPP macro.

PGRLSE=
  Specifies whether page releasing is to be performed for cell pool pages when
  there are no active cells. If PGRLSE=YES is specified, then SPAN=YES must
  not be specified. If PGRLSE=YES is specified and the cell pool extent, as
  defined in the parameter list, does not end on a page boundary, JES3 increases
  the number of cells in the extent to force the extent to end on a page boundary.

AUTODEL=
  Specifies whether secondary extents of a user-managed cell pool are to be
  deleted when no cells are in use. A secondary extent will not be deleted until
  5% of the cells in the pool are available.

PGFIX=
  Specifies whether page fixing is to be performed for all cell pool extents.

  YES
    Specifies that page fixing is to be performed. If this option is specified, the
    SPAN=YES must not be specified. If this option is specified and the cell
    pool extent, as defined in the parameter list, does not end on a page
    boundary, JES3 increases the number of cells in the extent to force the
    extent to end on a page boundary.

  NO
    Specifies that page fixing is not to be performed.

SPAN=
  Specifies whether cells can span page boundaries. If then PGRLSE=YES must
  not be specified.

LOC=
  Specifies the residence requirements of the storage being obtained for the cell
  pool. ANY indicates the cell pool may reside anywhere in virtual storage.
  BELOW indicates that the cell pool is to reside below 16 megabytes in virtual
  storage. (BELOW,ANY) indicates that the cell pool is to reside below 16
  megabytes in virtual storage, but may be backed anywhere in central storage
  even when fixed. If the LOC= parameter is specified on a call to expand an
  existing cell pool, it is ignored.

Return Codes
  Following are the error return codes returned in register 15:

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Hex</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>04</td>
<td>04</td>
<td>The value specified for the CPADDR parameter is nonzero and does not point to</td>
</tr>
<tr>
<td></td>
<td></td>
<td>a primary cell pool control block (CPB).</td>
</tr>
<tr>
<td>08</td>
<td>08</td>
<td>The primary cell pool extent or secondary extents contain unallocated cells.</td>
</tr>
<tr>
<td>0C</td>
<td>0C</td>
<td>Storage is not available.</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>The cell size is not a multiple of 4.</td>
</tr>
<tr>
<td>14</td>
<td>14</td>
<td>The maximum number of extents is too large.</td>
</tr>
<tr>
<td>18</td>
<td>18</td>
<td>The cell size (in the parameter list or the primary CPB) is not within the range of</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 to 4096.</td>
</tr>
<tr>
<td>1C</td>
<td>1C</td>
<td>The LIST parameter was not specified. This parameter is required when creating</td>
</tr>
<tr>
<td></td>
<td></td>
<td>a cell pool.</td>
</tr>
<tr>
<td>Return Code In Hex</td>
<td>Meaning</td>
<td></td>
</tr>
<tr>
<td>-------------------</td>
<td>-------------------------------------------------------------------------</td>
<td>---</td>
</tr>
<tr>
<td>20</td>
<td>The PGRLE and SPAN parameters were specified when creating a new cell pool; this is not valid.</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>The number of cells for the secondary extent was not specified.</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>The extent size is greater than X'7FFFFFFF' bytes.</td>
<td></td>
</tr>
<tr>
<td>2C</td>
<td>SPAN=YES and PGFIX=YES were both specified; this is not valid.</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>The calling function requested more reserved cells than could be contained in the primary cell pool extent. The CRSVD parameter of the IATYCPP parameter list (used with the IATXBPL macro) indicates the number of cells in the primary extent to be reserved. For JES3 console support services, the number of reserved cells cannot exceed 30% of the primary cell pool extent. For all other functions, the maximum number of reserved cells cannot exceed the primary extent size. If the maximum is exceeded, message IAT3673 is issued indicating use of the default value.</td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>A non-zero subpool was specified along with a data space. Because subpools do not exist in data spaces, the specification is rejected. No data space has been built for the IATXBPL request.</td>
<td></td>
</tr>
</tbody>
</table>

**Linkage**

This macro uses ASAVE linkage.

**Restrictions**

Do not use this macro in any routine that might run under the JES3 auxiliary task. This macro uses register 2.

**Example of Use**

The following is an example of IATXBPL:

```assembler
IATXBPL CPADDR=0,
   SPECIFIES CELL POOL CREATE
NAVAIL=CPNAVAIL,
   NOT AVAILABLE ROUTINE
ERROR=CPERROR,
   ERROR ROUTINE
LIST=CPPADDR,
   CELL POOL PARAMETER LIST
LOC=(BELOW,ANY)
   BELOW 16MB, BACKED ANYWHERE
```

**IATXBQN (Create Qualified Names)**

The IATXBQN macro creates qualified names consisting of multiple qualifiers. Issue this macro once for each qualifier.

**Syntax**

```
IATXBQN label
   DATA=address, WORK=(R1)
   (reg)
   address
   ,LAST=YES, ,COMPRESS=NO
```

Chapter 2. JES3 Macros 247
IATXBQN

Parameters

DATA=
Specifies the address of the qualified data that the macro will insert into the address specified by the WORK= parameter.

WORK=
Specifies the work area where the specified qualifier data (DATA=) is to be placed. Use the WORK=addr form of this parameter only for the first issuance of this macro for the specific qualified name.

When WORK=addr is specified for the first qualifier, JES3 puts this value from the first invocation into register 1. Then for all of the following issuances, use WORK=(R1). Do not change the value of register 1 between invocations of IATXBQN.

LAST=
Specifies whether this is the last qualifier of the qualified data set name. JES3 will not insert a period after the last qualifier. This parameter is required for the last macro invocation for the specific qualified name.

COMPRESS=
Specifies whether JES3 should compress trailing blanks or trailing zeros before inserting the period between qualifiers. JES3 expects the qualifiers to have a maximum of eight characters.

Linkage

This macro generates inline code. The contents of register 0 are destroyed.

Restrictions

None

Example

The following example demonstrates how to use a sequence of five IATXBQN macro invocations to construct a five-part qualified data set name. In this case the data for the node name, userid, and the data set identifier need to be compressed.

IATXBQN DATA=RQNODE
IATXBQN DATA=RQTUSID
IATXBQN DATA=RQJNUM,COMPRESS=NO
IATXBQN DATA=JDSDNUM,COMPRESS=NO
IATXBQN DATA=JDSDSNAM,LAST=YES

IATXCDVE (Reset a SNA Device Entry Table)

The IATXCDVE macro resets an SNA device entry table (DVE) to its initial state.

Syntax

```
IATXCDVE label DVE=(reg) address
```

Parameters

DVE=
Specifies the address of the DVE to be reset.
Linkage

This macro generates inline code. The contents of register 1 are destroyed.

Restrictions

Do not use this macro in any routine that might run in a C/I FSS address space.

IATXcio (Communicate with a C/I FSS Address Space)

The C/I driver module and the job segment scheduler (JSS) use the IATXcio macro to communicate with a C/I FSS address space. The method of communication is called an ORDER. The IATXcio macro can tell the C/I FSS address space to perform any one of the following functions:

- Process, cancel, or fail a job
- Build the procedure library tables and dynamically allocate or unallocate them
- Attach the C/I subtasks and build interpreter control tables
- Update the maximum number of C/I subtasks that may be active in the C/I FSS address space and the maximum number of JCL statements that may be processed by the address space
- End normally when it finishes its work

This macro can also map the parameter list and equates used with the PARMS parameter.

Syntax

```
/SM590000 IATXcio TYPE= label
           PJCI
            PJCANCL
            PJFAIL
            DUMP
            NO
            YES
            PROCBLD
            PROCENBL
            PROCDSBL
            MDCTINIT
            MDCTNORM
            STOPFSSN
            DSECT
            /SM590000
            /SM630000

            PARMS=(reg)
            address
            NAVAIL=(reg)
            address
            NORMAL=(reg)
            address
            /SM590000
            /SM630000

            ERROR=(reg)
            address
```

Parameters

**TYPE=**

Specifies the type of ORDER to be sent to the C/I FSS address space.

**PJCI**

Specifies that the C/I FSS address space should process the job identified by the parameter list.
PJCANCL
Specifies that the C/I FSS address space should cancel the job identified by the parameter list.

PJFAIL
Specifies that the C/I FSS address space should fail the DSP processing the job identified by the parameter list.

PROCBLD
Specifies that the C/I FSS address space should build the procedure library tables and destroy the existing ones, if any. It should also dynamically allocate, concatenate, open, check, and issue the BLDL macro for the procedure libraries.

PROCENBL
Specifies that the C/I FSS address space should dynamically allocate the procedure library identified by the parameter list.

PROCDSBL
Specifies that the C/I FSS address space should dynamically unallocate (that is, no longer have access to) the procedure library identified by the parameter list.

MDCTINIT
Specifies that the C/I FSS address space should build the interpreter control tables (IATYICTs). It should also attach the number of C/I subtasks required by the DSP maximum count field in the FSS table pointed to by the PARMS parameter specification.

MDCTNORM
Specifies that the C/I FSS address space should update the maximum number of C/I DSPs and subtasks allowed to run in the address space and the maximum number of JCL statements that the address space may process.

STOPFSSN
Specifies that the C/I FSS address space should end normally when it finishes its work.

DSECT
Specifies that a mapping of the parameter list and equates used in communication with a C/I FSS address space should be generated. The equates define the reason codes generated by an error while executing the IATXCIO macro. This parameter must be used alone (without any other parameters).

PARMS=
Specifies the address of a parameter list. This parameter list is mapped by the IATXCIO macro specifying TYPE=DSECT. The parameters that must be included in the parameter list depend on what you specified for the TYPE parameter. The following table shows you which parameters to include for each subparameter.

<table>
<thead>
<tr>
<th>TYPE</th>
<th>1st PARAMETER</th>
<th>2nd PARAMETER</th>
<th>3rd PARAMETER</th>
</tr>
</thead>
<tbody>
<tr>
<td>PJCI</td>
<td>FSS TABLE ADDRESS</td>
<td>RESQ ADDRESS</td>
<td>JCT ADDRESS</td>
</tr>
<tr>
<td>PJCANCEL</td>
<td>FSS TABLE ADDRESS</td>
<td>RESQ ADDRESS</td>
<td>NONE</td>
</tr>
<tr>
<td>PJFAIL</td>
<td>FSS TABLE ADDRESS</td>
<td>RESQ ADDRESS</td>
<td>NONE</td>
</tr>
<tr>
<td>PROCBLD</td>
<td>FSS TABLE ADDRESS</td>
<td>NONE</td>
<td>NONE</td>
</tr>
<tr>
<td>PROCENBL</td>
<td>FSS TABLE ADDRESS</td>
<td>PROCLIB TABLE ADDRESS</td>
<td>NONE</td>
</tr>
<tr>
<td>TYPE=</td>
<td>1st PARAMETER</td>
<td>2nd PARAMETER</td>
<td>3rd PARAMETER</td>
</tr>
<tr>
<td>---------</td>
<td>-----------------------</td>
<td>--------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>PROCDSBL</td>
<td>FSS TABLE ADDRESS</td>
<td>PROCLIB TABLE</td>
<td>NONE</td>
</tr>
<tr>
<td></td>
<td>ADDRESS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MDCTINIT</td>
<td>FSS TABLE ADDRESS</td>
<td>NONE</td>
<td>NONE</td>
</tr>
<tr>
<td>MDCTNORM</td>
<td>FSS TABLE ADDRESS</td>
<td>NONE</td>
<td>NONE</td>
</tr>
<tr>
<td>STOPFSSN</td>
<td>FSS TABLE ADDRESS</td>
<td>NONE</td>
<td>NONE</td>
</tr>
</tbody>
</table>

**NAVAIL=**

Specifies the address the routine returns to if the C/I FSS address space is not ready to receive the ORDER.

**DUMP=**

When TYPE=PJFAIL is coded, this parameter specifies whether the DSP to be failed should be failed with a dump. Otherwise, this parameter is ignored.

**NORMAL=**

This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

**ERROR=**

Specifies the address the routine returns to if an error occurs. Dump code DM203 is returned in register 1 and a reason code is returned in register 15. The reason codes are defined in the equates for IATXcio, which you may obtain using the IATXcio mapping macro.

If this parameter is not specified, the reason code is returned in register 2 and the service routine issues the FAILDSP macro.

**Linkage**

This macro uses ASAVE linkage.

**Restrictions**

Only the C/I driver module and JSS may issue this macro.

**IATXCKPT (Request Checkpoint Record Access Method)**

The IATXCKPT macro requests checkpoint record access method functions.

This macro uses registers 0 and 1 for passing parameters and register 15 for a return code.

**Syntax**
Parameters

**TYPE**=
Specifies the type of checkpoint record access method function being requested.
- **RESERVE**
  Reserve the checkpoint data set(s) for the caller’s use.
- **RELEASE**
  Cancel a previous request to reserve the checkpoint data set(s).
- **WRITE**
  Write the checkpoint record specified by the RECORD parameter to the checkpoint data set(s).
- **READ**
  Read the record specified by the ID parameter from the checkpoint data set(s).
- **PURGE**
  Purge the record specified by the ID parameter from the checkpoint data set(s).
- **FREE**
  Free storage for the record.

**ID**=
- ‘recid’
- address
  Specifies the identifier (id) of a record or the address of a 4-character field containing the id of a record to be read or purged from the checkpoint data set(s). Do not specify 0, blank, ‘CMAP’, or ‘I/OE’.
- ‘ALL’
  Specifies that all records are to be purged from the checkpoint data set. This parameter is valid only with the TYPE=PURGE parameter.

**RECORD**=
Specifies the address of a checkpoint record to be written to the checkpoint data set(s). The address must begin on a fullword boundary. You must specify this parameter if you specify the TYPE=WRITE parameter. This parameter is incorrect with any of the other types of checkpoint record access method requests.

**MODE**=
- **FCT**
  Specifies that the issuer of this macro is running under the control of an FCT.
- **TASK**
  Specifies that the issuer of this macro is running directly under the control of the JES3 primary task or a task-level routine, such as JES3 recovery.

**ADDR**=
Specifies the control block (IATYTVT or IATYITK) to be used to get the address...
of the checkpoint record access method function. Addressability to the control block must be established by the calling routine. This parameter is valid only with the MODE=TASK parameter.

Return Codes
Following are the error return codes returned in register 15:

<table>
<thead>
<tr>
<th>Type of Request</th>
<th>Return Code in Hex</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESERVE</td>
<td>00</td>
<td>The request completed successfully.</td>
</tr>
<tr>
<td></td>
<td>04</td>
<td>The request could not be satisfied because another JES3 function is using the checkpoint record access method. The ECF address is returned in register 0. The ECF mask needed is contained in the low-order byte of register 1, with the other three bytes containing zeros. The ECF mask and address may be used to wait for the access method to become available.</td>
</tr>
<tr>
<td>RELEASE</td>
<td>00</td>
<td>The request completed successfully.</td>
</tr>
<tr>
<td></td>
<td>04</td>
<td>No previous RESERVE request was outstanding, therefore, a RELEASE could not be performed.</td>
</tr>
<tr>
<td>READ</td>
<td>00</td>
<td>The request completed successfully. Register 1 contains the address of the checkpoint record.</td>
</tr>
<tr>
<td></td>
<td>04</td>
<td>The request could not be satisfied because another JES3 function is using the checkpoint record access method. The ECF address is returned in register 0. The ECF mask needed is contained in the low-order byte of register 1, with the other three bytes containing zeros. The ECF mask and address may be used to wait for the access method to become available.</td>
</tr>
<tr>
<td></td>
<td>08</td>
<td>The checkpoint record name specified by the ID parameter is invalid. That is, the ID parameter specifies 0, blank, ‘CMAP’, or ‘I/OE’, or ‘ALL’ without TYPE=PURGE.</td>
</tr>
<tr>
<td></td>
<td>0C</td>
<td>No checkpoint record with the specified name could be found.</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>The requested record could not be read in because there was not enough virtual storage.</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>The specified checkpoint record is incorrect and cannot be read.</td>
</tr>
<tr>
<td></td>
<td>18</td>
<td>The specified checkpoint record could not be read because of a permanent I/O error.</td>
</tr>
<tr>
<td></td>
<td>1C</td>
<td>A back-level copy of the requested record has been read because the current copy of the record is incorrect or contains a permanent I/O error. Register 1 contains the address of the back-level copy of the record.</td>
</tr>
<tr>
<td>WRITE</td>
<td>00</td>
<td>The record was written successfully to at least one checkpoint data set.</td>
</tr>
<tr>
<td></td>
<td>04</td>
<td>The request could not be satisfied because another JES3 function is using the checkpoint record access method. The ECF address is returned in register 0. The ECF mask needed is contained in the low-order byte of register 1, with the other three bytes containing zeros. The ECF mask and address may be used to wait for the access method to become available.</td>
</tr>
<tr>
<td></td>
<td>08</td>
<td>The checkpoint record name specified in the ID field of the record is incorrect or the record length or address is incorrect.</td>
</tr>
<tr>
<td></td>
<td>0C</td>
<td>The record could not be written to either checkpoint data set because of insufficient storage space. No previous level of the checkpoint record exists in one checkpoint data set or more.</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>The record could not be written to either checkpoint data set because of insufficient storage space. A previous level of the record exists and was not altered.</td>
</tr>
</tbody>
</table>
## IATXCKPT

<table>
<thead>
<tr>
<th>Type of Request</th>
<th>Return Code in Hex</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td></td>
<td>The record could not be written to either checkpoint data set because of insufficient storage space. A previous level of the record existed and has been partially destroyed.</td>
</tr>
<tr>
<td>18</td>
<td></td>
<td>The record could not be written to either checkpoint data set because of a permanent I/O error.</td>
</tr>
<tr>
<td>1C</td>
<td></td>
<td>An error was detected when updating the track map(s) in the checkpoint data set(s). The status of the record is unpredictable.</td>
</tr>
<tr>
<td>PURGE</td>
<td>00</td>
<td>The record was successfully purged from at least one checkpoint data set.</td>
</tr>
<tr>
<td>04</td>
<td></td>
<td>The request could not be satisfied because another JES3 function is using the checkpoint record access method. The ECF address is returned in register 0. The ECF mask needed is contained in the low-order byte of register 1, with the other three bytes containing zeros. The ECF mask and address may be used to wait for the access method to become available.</td>
</tr>
<tr>
<td>08</td>
<td></td>
<td>The checkpoint record name specified by the ID parameter is invalid. That is, the ID parameter specifies 0, blank, ‘CMAP’, or ‘/IOE’, or ‘ALL’ without TYPE=PURGE.</td>
</tr>
<tr>
<td>0C</td>
<td></td>
<td>No checkpoint record with the specified name could be found.</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>An error was detected when updating the track map(s) in the checkpoint data set(s). The record was purged from the in-storage track map, therefore, the record is no longer accessible. The status of the record in the checkpoint data set(s) is unpredictable.</td>
</tr>
<tr>
<td>FREE</td>
<td>00</td>
<td>The record was successfully freed.</td>
</tr>
<tr>
<td>08</td>
<td></td>
<td>The specified checkpoint record ID, length or address is incorrect.</td>
</tr>
</tbody>
</table>

### Linkage

If this macro is issued from a routine running under control of an FCT, the macro uses ASAVE linkage; if the issuing routine is running under the control of a started task, the macro uses a BALR R14,R15 instruction and the checkpoint record access method saves the caller’s registers.

### Restrictions

This macro is a JES3 internal service macro. Use of this macro by installation exit routines and user-written DSPs can degrade system performance and is not recommended. Do not use this macro in any routine that might run in a C/I FSS address space, or under the JES3 auxiliary task, or in user code during initialization.

### IATXCNDB (Modify Console Destination Block)

The IATXCNDB macro allows you to modify a console destination block (mapping macro IATYCNDB) in the following ways:

- **Initialize**
  - Initialize the specified console destination block by supplying the console name, console identifier, routing code mask, command and response token (CART) and command indicator.

- **Transfer**
  - Copy the specified console destination block to the new specified area.
**Update**

Update the console identifier, routing code mask, command and response token (CART), and command indicator of the specified console destination block.

**Reset**

Set to zero the console identifier, routing code mask, command and response token (CART), and command indicator of the specified console destination block.

**Verify**

Verify that the console destination block is at the correct level for the current JES3 release.

**Extract**

Extract the console identifier, routing code mask, command and response token (CART), and command indicator of the specified console destination block.

**Syntax**

```
IATXCNDB

label

INITIALIZE

TRANSFER

UPDATE

RESET

VERIFY

EXTRACTCONSID

EXTRACTCONSNAME

EXTRACTCONSTYPE

EXTRACTROUT

EXTRACTCART

TRANSCONSID

TRANSROUT

-CNDB=(reg)

address

OUTCNDB=(reg)

address

INCNDB=(reg)

address

CONSID=(reg)

address

CONSNM=(reg)

address

CART=(reg)

address

OUTCONSID=(reg)

address

OUTCONSNAME=(reg)

address

OUTCONSTYPE=(reg)

address

OUTROUT=(reg)

address

OUTCART=(reg)

address

CMDIND=YES

ABEND=NO

RETC=(reg)

address

```
Parameters

OPERATION=
Specifies the type of operation that you want performed on the console destination block.

INITIALIZE
Indicates that the specified console destination block pointed to by the CNDB parameter should be initialized. Any information pointed to by the console identifier (CONSID), the console name (CONSNM), routing code mask (ROUT), the command response token (CART), and the command indicator (CMDIND) parameters is used to initialize this control block. CONSID and CONSNM are optional parameters and are mutually exclusive. CART, ROUT, and CMDIND are optional parameters.

TRANSFER
Indicates that the specified console destination block pointed to by the CNDB parameter should be verified and copied to the console destination block pointed to by the OUTCNDB parameter.

UPDATE
Indicates that the specified console destination block pointed to by the CNDB parameter should be updated with information contained in the console identifier (CONSID), the console name (CONSNM), routing code mask (ROUT), the command response token (CART), and the command indicator (CMDIND). This operation assumes that the console destination block has previously been initialized. CONSID and CONSNM are optional parameters and are mutually exclusive. CART, ROUT, and CMDIND are optional parameters.

RESET
Indicates that the specified console destination block pointed to by the CNDB parameter should be reset to contain only an eyecatcher and version indicator.

VERIFY
Verifies that the specified console destination block pointed to by the CNDB parameter contains the correct level for the current JES3 release. If it is not at the correct level, JES3 reformats this console destination block to the correct level. Modules using VERIFY must expand the IATYCNDDB macro.

EXTRACTCONSID
Indicates that the specified console destination block pointed to by the CNDB parameter should be verified and the console identifier should be extracted and copied to the area pointed to by the OUTCONSID parameter.

EXTRACTCONSNAME
Indicates that the specified console destination block pointed to by the
CNDB parameter should be verified and the console name should be extracted and copied to the area pointed to by the OUTCONSNAME parameter.

**EXTRACTCONSTYPE**
Indicates that the specified console destination block pointed to by the CNDB parameter should be verified and the console type should be extracted and copied to the area pointed to by the OUTCONSTYPE parameter.

**EXTRACTROUT**
Indicates that the specified console destination block pointed to by the CNDB parameter should be verified and the routing code mask should be extracted and copied to the area pointed to by the OUTROUT parameter.

**EXTRACTCART**
Indicates that the specified console destination block pointed to by the CNDB parameter should be verified and the command and response token (CART) should be extracted and copied to the area pointed to by the OUTCART parameter.

**TRANSCONSID**
Indicates that the specified console destination block pointed to by the CNDB parameter should be verified and the console identifier, console name and console type should be copied to the console destination block pointed to by the OUTCNDB parameter.

**TRANSROUT**
Indicates that the specified console destination block pointed to by the CNDB parameter should be verified and the routing code mask should be copied to the console destination block pointed to by the OUTCNDB parameter.

**CNDB**= Specifies the name or address in register (2)-(12), of the console destination block on which the specified operation will be performed. This is a required parameter.

**OUTCNDB**= Specifies the name or address in register (2)-(12), of the console destination block (or information from the input console destination block) is transferred. This parameter is only allowed for an OPERATION=TRANSFER, OPERATION=TRANSROUT or OPERATION=TRANSCONSID call and will be ignored for all other operations. The default is NULL.

**INCNDB**= Specifies the name or address in register (2)-(12), of the console destination block when the INITIALIZE operation is to use information from an input console destination block to initialize a second console destination block. The default is NULL.

**CONSID**= Specifies the name or address in register (2)-(12), of the console identifier. The CONSID parameter is only meaningful for an OPERATION=INITIALIZE or OPERATION=UPDATE call and will be ignored for all other operations. The default is NULL.

**ROUT**= Specifies the name or address in register (2)-(12), of a 17 byte routing code mask. Each bit in the first 16 bytes of the routing code mask represents a different routing code. The 17th byte is used to represent the JES3 destination.
classes that don’t have a routing code equivalent. The ROUT parameter is only meaningful for an OPERATION=INITIALIZE or OPERATION=UPDATE call and will be ignored for all other operations. MCS uses routing codes to route messages to those consoles directed to receive messages. The default is NULL.

CONSNM=
Specifies the name or address in register (2)-(12), of an eight character console name. The CONSNM parameter is only meaningful for an OPERATION=INITIALIZE or OPERATION=UPDATE call and will be ignored for all other operations. The default is NULL.

CART=
Specifies the name or address in register (2)-(12), of an eight character command and response token that is used to direct command responses back to the console of the command issuer. The parameter is only meaningful for an OPERATION=INITIALIZE or OPERATION=UPDATE call and will be ignored for all other operations. The default is NULL.

OUTCONSID=
Specifies the name or address in register (2)-(12), of a four byte area of storage in which to extract the console destination block console identifier. The OUTCONSID parameter is only meaningful for an OPERATION=EXTRACTCONSID and will be ignored on all other operations. The default is NULL.

OUTCONSNAME=
Specifies the name or address in register (2)-(12), of a eight byte area of storage in which to extract the console destination block console name. The OUTCONSNAME parameter is only meaningful for an OPERATION=EXTRACTCONSNAME and will be ignored on all other operations. The default is NULL.

OUTCONSTYPE=
Specifies the name or address in register (2)-(12), of a one byte area of storage in which to extract the console destination block console type. The OUTCONSTYPE parameter is only meaningful for an OPERATION=EXTRACTCONSTYPE and will be ignored on all other operations. The default is NULL.

OUTROUT=
Specifies the name or address in register (2)-(12), of a seventeen byte area of storage in which to extract the console destination block routing code mask. The OUTROUT parameter is only meaningful for an OPERATION=EXTRACTROUT and will be ignored on all other operations. The default is NULL.

OUTCART=
Specifies the name or address in register (2)-(12), of a eight byte area of storage in which to extract the command and response token (CART). The OUTCART parameter is only meaningful for an OPERATION=EXTRACTCART and will be ignored on all other operations. The default is NULL.

CMDIND={Yes|NO}
Specifies whether this console destination block is a command response. The CMDIND parameter is only meaningful for an OPERATION=INITIALIZE or OPERATION=UPDATE call and will be ignored for all other operations. No is the default.
ABEND=\{YES|NO\}
Specifies whether an ABEND should occur in case of error. Yes, the default,
results in an ABEND. Specifying No causes processing to continue and a return
and reason code to be passed back to the caller.

RETC=
Specifies where the IATXCNDB macro places the return code. Possible return
codes are:

<table>
<thead>
<tr>
<th>Reason Code</th>
<th>Return Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00</td>
<td>The operation requested was successfully performed.</td>
</tr>
</tbody>
</table>
| 4           | 00          | JES3 encountered an error while processing. See the reason
codes for additional information about the failure. |

RSNC=
Specifies where the IATXCNDB macro places the reason code.

<table>
<thead>
<tr>
<th>Reason Code</th>
<th>Return Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>00</td>
<td>Operation was successfully performed.</td>
</tr>
</tbody>
</table>
| 1C          | 00          | The IATYCNDB was successfully initialized, but the console
name or identifier passed to the INITIALIZE routine was not
found in the JES3 console table. The IATYCNDB has been
initialized with a "non-JES3" console type and the console
identifier and console name passed on the IATXCNDB
invocation and returned by MCS. |
| 30          | 00          | An error return greater than 4 was received from the call to
CONVCON by the IATXCNDB service VERIFY operation. |
| 34          | 00          | The IATYCNDB was successfully converted to an IATYCNDB at
the current release level, but the console name in the down
level CNDB was not found in the JES3 console table. The
IATYCNDB has been initialized with a "non-JES3" console type
and the console identifier returned by MCS. |
| 90          | 00          | The IATYCNDB was successfully updated, but the console
name or identifier passed to the UPDATE routine was not
found in the JES3 console table. The IATYCNDB has been updated
with a "non-JES3" console type and the console identifier and
console name passed on the IATXCNDB invocation and
returned by MCS. |
| 04          | 04          | The OPERATION requested is not supported. |
| 08          | 04          | An invalid IATXCNDB parameter list was passed to the
IATCNDB module. The eyecatcher in the parameter list is not
"YCNDB" |
| 0C          | 04          | An invalid IATXCNDB parameter list was passed to the
IATCNDB module. The version indicator in the parameter list
was not equal to the current version. |
| 10          | 04          | The INCNDB parameter passed to the INITIALIZE operation did
not have 'CNDB' as its eyecatcher. |
| 14          | 04          | The console identifier and console name were passed to the
INITIALIZE service. These parameters are mutually exclusive. |
| 18          | 04          | An error return (greater than 4) was received from a call to
CONVCON by the INITIALIZE operation. |
| 20          | 04          | The CNDB parameter passed to the TRANSFER operation was
zero. |
| 24          | 04          | The CNDB parameter passed to the TRANSFER operation did
not have 'CNDB' as its eyecatcher. |
| 28          | 04          | The OUTCNDB parameter passed to the TRANSFER operation
was zero. |
| 2C          | 04          | The CNDB parameter passed to the VERIFY operation did not
have 'CNDB' as its eyecatcher. |
<table>
<thead>
<tr>
<th>Reason Code</th>
<th>Return Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>38</td>
<td>04</td>
<td>The CNDB parameter passed to the TRANSCONSID operation was zero.</td>
</tr>
<tr>
<td>3C</td>
<td>04</td>
<td>The CNDB parameter passed to the TRANSCONSID operation did not have 'CNDB' as its eyecatcher.</td>
</tr>
<tr>
<td>40</td>
<td>04</td>
<td>The OUTCNDB parameter passed to the TRANSCONSID operation was zero.</td>
</tr>
<tr>
<td>44</td>
<td>04</td>
<td>The CNDB parameter passed to the TRANSROUT operation was zero.</td>
</tr>
<tr>
<td>48</td>
<td>04</td>
<td>The CNDB parameter passed to the TRANSROUT operation did not have 'CNDB' as its eyecatcher.</td>
</tr>
<tr>
<td>4C</td>
<td>04</td>
<td>The OUTCNDB parameter passed to the TRANSROUT operation was zero.</td>
</tr>
<tr>
<td>50</td>
<td>04</td>
<td>The CNDB parameter passed to the EXTRACTCONSNAME operation was zero.</td>
</tr>
<tr>
<td>54</td>
<td>04</td>
<td>The CNDB parameter passed to the EXTRACTCONSNAME operation did not have 'CNDB' as its eyecatcher.</td>
</tr>
<tr>
<td>58</td>
<td>04</td>
<td>The OUTCONSNAME address passed to the EXTRACTCONSNAME operation was zero.</td>
</tr>
<tr>
<td>5C</td>
<td>04</td>
<td>The CNDB parameter passed to the EXTRACTCONSNAME operation was zero.</td>
</tr>
<tr>
<td>60</td>
<td>04</td>
<td>The CNDB parameter passed to the EXTRACTCONSNAME operation did not have 'CNDB' as its eyecatcher.</td>
</tr>
<tr>
<td>64</td>
<td>04</td>
<td>The OUTCONSNAME address passed to the EXTRACTCONSNAME operation was zero.</td>
</tr>
<tr>
<td>68</td>
<td>04</td>
<td>The CNDB parameter passed to the EXTRACTCONSTYPE operation was zero.</td>
</tr>
<tr>
<td>6C</td>
<td>04</td>
<td>The CNDB parameter passed to the EXTRACTCONSTYPE operation did not have 'CNDB' as its eyecatcher.</td>
</tr>
<tr>
<td>70</td>
<td>04</td>
<td>The OUTCONSTYPE address passed to the EXTRACTCONSTYPE operation was zero.</td>
</tr>
<tr>
<td>74</td>
<td>04</td>
<td>The CNDB parameter passed to the EXTRACTROUT operation was zero.</td>
</tr>
<tr>
<td>78</td>
<td>04</td>
<td>The CNDB parameter passed to the EXTRACTROUT operation did not have 'CNDB' as its eyecatcher.</td>
</tr>
<tr>
<td>7C</td>
<td>04</td>
<td>The OUTROUT address passed to the EXTRACTROUT operation was zero.</td>
</tr>
<tr>
<td>80</td>
<td>04</td>
<td>The CNDB parameter passed to the UPDATE operation was zero.</td>
</tr>
<tr>
<td>84</td>
<td>04</td>
<td>The CNDB parameter passed to the UPDATE operation did not have 'CNDB' as its eyecatcher.</td>
</tr>
<tr>
<td>88</td>
<td>04</td>
<td>An attempt was made to pass the CONSID and CONSNM parameters to the UPDATE operation. These parameters are mutually exclusive.</td>
</tr>
<tr>
<td>8C</td>
<td>04</td>
<td>An error return (greater than 4) was received from the call to CONVCON by the UPDATE operation.</td>
</tr>
<tr>
<td>94</td>
<td>04</td>
<td>The CNDB parameter passed to the RESET operation was zero.</td>
</tr>
<tr>
<td>98</td>
<td>04</td>
<td>The CNDB parameter passed to the RESET operation did not have 'CNDB' as its eyecatcher.</td>
</tr>
<tr>
<td>A8</td>
<td>04</td>
<td>The CNDB parameter passed to the EXTRACTCART operation was zero.</td>
</tr>
<tr>
<td>AC</td>
<td>04</td>
<td>The CNDB parameter passed to the EXTRACTCART operation did not have 'CNDB' as its eyecatcher.</td>
</tr>
<tr>
<td>B0</td>
<td>04</td>
<td>The OUTROUT address passed to the EXTRACTCART operation was zero.</td>
</tr>
</tbody>
</table>
MF=
  Specifies the macro form.

S  Specifies the standard form of the macro. The standard form generates code to put the parameters in a parameter list and invoke the specified operation. It is the default.

L  Specifies the list form of the macro. The list form defines an area to be used for the parameter list.

  xmfctrl
    Specifies the name of a storage area for the parameter list. It is required input.

  xmfattr
    Specifies an optional character string that is used to force boundary alignment of the parameter list. The default is 0D which forces the parameter list to a doubleword boundary.

E  Specifies the execute form of the macro. The execute form generates code to put the parameters into the parameter list specified by xmfctrl.

  xmfctrl
    Specifies the name or address in register (1)-(12), of a storage area for the parameter list. It is required input.

COMPLETE
  Specifies the macro parameter syntax checking. A check is done for required macro keys.

Linkage

This macro uses BASSM-BAKR-PR linkage.

Restrictions

Do not use this macro in any routine that might run under the JES3 auxiliary task.

When the program invoking this macro is running in the users address space, the environment must be specified on the IATYMOD macro so that proper linkage may be established. This macro assumes addressability to IATYTVT through register 12, unless invoked from the user address space.

IATXCNS (Read a Console Input Buffer Record from Spool)

The IATXCNS macro reads a console input buffer record (field CONSMESS of data area IATYCNS) from the JES3 spool. The macro can either:
  • Read the job description accounting block (JDAB) associated with the job, and read and return the address of the console input buffer record, or
  • Release the console input buffer record returned by a previous macro call.

Note: The console input buffer is sometimes called an input message buffer or an input parameter buffer. Issuing the IATYCNS macro with TYPE=INPUT generates the buffer table.

Syntax

```
  label: IATXCNS TYPE=GET RELEASE,BUFFER=(reg) address
```

Chapter 2. JES3 Macros  261
IATXCNST

Parameters

TYPE=

GET
Specifies that the address of a console input buffer record is to be returned in register 1.

RELEASE
Specifies that a console input buffer record returned by a previous IATXCNST TYPE=GET macro call is to be released. The address of the console input buffer record must be specified by the BUFFER parameter.

If you specify TYPE=RELEASE, this macro will use register 2.

BUFFER=
Specifies the address of a console input buffer record returned by a previous IATXCNST TYPE=GET macro call.

ERROR=
Specifies the address of the routine to receive control if an error is detected during IATXCNST processing.

On return from a TYPE=GET request, the registers contain:
• For a normal return:
  – Register 0 contains zeros.
  – Register 1 contains the address of the console buffer containing the rebuilt input parameters.
  – Register 15 contains zeros.
• For an error return:
  – Register 0 contains dump code 105.
  – Register 1 contains zeros.
  – Register 15 contains a return code 4.

On return from a TYPE=RELEASE request, the registers contain:
• For a normal return:
  – Registers 0, 1, and 15 all contain zeros.
• For an error return:
  – Register 0 contains dump code 105.
  – Register 1 contains zeros.
  – Register 15 contains a return code 4.

Linkage
This macro uses ASAVE linkage.

Restrictions
Do not use this macro in any routine that might run under the JES3 auxiliary task.
IATXCNT (Set a Negative Use or Allocation Count to Zero)

The IATXCNT macro sets a negative use count or allocation count to zero. A negative use count can occur in the SYSUNITs, SETDSN, or SETVOL table. A negative allocation count can occur in the SETVOL table. The macro issues a message indicating which of these counts was set to zero.

Syntax

```
IATXCNT—INDEX=“label”

Index address , ENTRY= Group address , NORMAL= address
```

Index address

(R0)

address

Group address

(R1)

address

Parameters

INDEX=

Specifies the address of an index value. The index value indicates which count should be set to zero, as follows:

<table>
<thead>
<tr>
<th>Value</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>SYSUNITS table use count</td>
</tr>
<tr>
<td>8</td>
<td>SETDSN table use count</td>
</tr>
<tr>
<td>12</td>
<td>SETVOL table use count</td>
</tr>
<tr>
<td>16</td>
<td>SETVOL table allocation count</td>
</tr>
</tbody>
</table>

ENTRY=

Specifies the address of the relevant table.

NORMAL=

This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

Linkage

This macro uses ASAVE linkage.

Restrictions

Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task.
IATXCPFYF (Copy One Multi-Record File to Another)

Use IATXCPFYF to copy one multi-record file (MRF) to another. The source multi-record file can contain spanned records.

The caller can optionally request:
- That only those records in the source file having the specified key be copied. This allow selective copying of the file. If the file contains spanned records, JES3 checks only those records that are not spanned for the specified key; spanned records are ignored.
- A list of logical records that should not be copied from the source to target file (that is, a record exclusion list). There is an entry in the list for each logical record to allow exclusion from the copy. For spanned records, all record segments are considered part of the same logical record.

Note: If an error occurs, the caller is responsible for closing the source and target files, if they are open.

Syntax

```
IATXCPFYF

FROMFDB=(reg) address

TOFDB=(reg) address

TATFDB=(reg) address

APPEND=YES

EXCLLIST=(reg) address

CARR=ASA MAC

ERROR=address

NORMAL=address

MF=E, (R1) reg address

KEYVAL=(reg) address

KEYOFF=(reg) address

KEYLEN=(reg) address
```

Parameters

FROMFDB={reg | address}

Specifies the address or register that contains the address of the multi-record file FDB that is the source of the copy. That is, JES3 copies the data associated with this FDB to the spool data set associated with the TOFDB parameter.
The caller can open the file for input before invoking this service. If the file is not open, JES3 issues an AOPEN request.

**Note:** If you specify a register, you cannot use R1.

**TOFDB=(reg | address)**
Specifies the address or register that contains the address of the multi-record file FDB that is the target of the copy. That is, JES3 copies the data associated with the FROMFDB parameter to the spool data set represented by this FDB.

The caller can open the file for output before invoking this service. If the file is not open, JES3 issues an AOPEN or AOPEND request depending on whether you also specified APPEND=YES or APPEND=NO.

**Note:** If you specify a register, you cannot use R1.

**TATFDB=(reg | address)**
Specifies the address or register that contains the address of the TAT FDB for the multi-record file associated with the TOFDB parameter. The TAT FDB address is required because spool space might need to be allocated when the data is copied to the multi-record file specified by TOFDB=.

**Note:** If you specify a register, you cannot use R1.

**APPEND=(YES | NO)**
Specifies whether the data is to be appended to the target multi-record file (YES) or whether it should replace any existing information in the target multi-record file (NO).

**KEYVAL=(reg | address)**
Specifies the address or register that contains the address of the key field value that identifies the records JES3 will selectively copy. When a record in the source file contains this value in the key position (as determined by the KEYOFF= and KEYLEN= parameters), JES3 copies the record to the target file.

The key value can be up to eight bytes in length.

**Notes:**
1. If you specify a key for selectively copying records, and the source file contains spanned records, JES3 ignores the spanned records.
2. If you specify a register, you cannot use either R0 or R1.
3. If you specify KEYVAL, you must also specify both KEYOFF and KEYLEN.

**KEYOFF=(reg | address)**
Specifies the address or register that contains the address of a 2-byte field which contains the offset of the key field within the records JES3 will selectively copy.

**Notes:**
1. If a register is specified, R0 and R1 cannot be used.
2. If KEYOFF is specified, KEYVAL and KEYLEN must be specified.

**KEYLEN=(reg | address)**
Specifies the address or register that contains the address of a 2-byte field which contains the length of the key field within the records JES3 will selectively copy.

**Notes:**
1. If a register is specified, R0 and R1 cannot be used.
2. If KEYLEN is specified, KEYVAL and KEYOFF must be specified.
**EXCLLIST={reg | address}**
Specifies the address of an exclusion list that specifies the records in the source file that should not be copied to the target file.

The exclusion list consists of a fullword count field that contains the number of records in the list, followed by one or more fullword fields containing the logical record numbers of the records to be excluded. For spanned records, all record segments are considered part of the same logical record.

**Note:** If you specify a register, you cannot use either R0 or R1.

**CARR={ASA | MAC}**
Specifies the carriage control characters associated with the source and target files.

**ASA**
Indicates that the files have ASA carriage control characters.

**MAC**
Indicates that the files have machine carriage control characters.

If you do not specify CARR, JES3 assumes that the files do not have carriage control characters.

**ERROR=address**
Specifies the address to which JES3 returns control if processing is unsuccessful. If you do not specify this parameter and an error occurs, JES3 issues a FAILDSP.

**Note:** In all cases, if an error occurs, the caller is responsible for closing the source and target files if they are open.

**NORMAL=address**
Specifies the address to which JES3 returns control if processing is successful.

**MF=(E,(R1) | REG | address)) | L**
Specifies whether this is the execute or list form of the macro:

**(E,address)**
Specifies the execute form of the macro with an existing parameter list specified by address passed to the routine.

**L**
Specifies that JES3 generate the list form of the parameter list. You cannot specify any other parameters.

**Linkage**
This macro uses ASAVE linkage.

**Restrictions**
Do not use this macro in any routine that might run under the JES3 auxiliary task.

**IATXCRPL (Reset a VTAM Request Parameter List)**
The IATXCRPL macro resets a VTAM request parameter list (RPL) to its initial state before use in a subsequent VTAM request. If the RPL is active, this macro issues a FAILDSP macro (if the caller is a DSP) or a MVS ABEND macro.
Syntax

IATXCRPL

Syntax

\[\text{label} \rightarrow \text{IATXCRPL} - \text{RPL} \rightarrow \text{address} - \text{ID} = \text{(module, subpath)} \rightarrow \]

\[\text{address} - \text{SRD} = \text{(reg)} \rightarrow \]

Parameters

RPL=
Indicates the address of the RPL to be reset.

ID=
Indicates a 4-character field consisting of the last two characters of the name of the module issuing the macro followed by a number from 0-255 identifying the subpath within the module.

SRD=
Specifies the register containing the address of the SNARJP DSP’s data CSECT (SRD). If this parameter is omitted, register 15 is used.

Linkage

This macro generates inline code. The contents of registers 0, 1, and 2 are destroyed by this macro expansion. If the SRD parameter is not specified, the contents of register 15 are destroyed.

Restrictions

This macro requires that the IATYSRD, IATYSRT, and IATYTVT mapping macros be assembled as DSECTS in the calling module. Do not use this macro in any routine that might run in a C/I FSS address space.

IATXCSF (Execute Code Containing MVS WAIT Macros)

The IATXCSF macro provides the ability to execute code containing implicit or explicit MVS WAIT macros (for example: OPEN, CLOSE, FIND). This is done by passing control from one of the JES3 subtasks provided for this purpose to an appendage routine specified by the ENTER parameter.

The two types of JES3 subtasks that can be invoked by the IATXCSF macro are:

- General subtasks: Any caller can use these subtasks.
- Specific subtasks: Callers specifying the subtask ID can use these subtasks.

IATXCSF uses one of the eight general subtasks when the ID parameter is not specified. The caller receives control under one of the subtasks and has no control over which subtask is used. When running under a general subtask’s control, you must free (before relinquishing control) all resources that were acquired while running under control of the subtask.
IATXCSF uses a specific subtask when the ID parameter is specified. All programs that issue the IATXCSF macro and specify the same ID receive control under the same subtask. Resources acquired while running under a specific subtask are held by the subtask even after the caller relinquishes control. Therefore, the specific ID parameter should only be used when resources must be maintained under the same TCB. If a specific subtask is needed by more than one FCT, the caller must issue an ESTAE macro and provide an ESTAE exit to clean up and prevent the subtask from ending. If the subtask is allowed to end, all resources held for other FCTs are released.

**Note:** JES3 ASAVE-supported macros must not be issued from a subtask appendage.

### Syntax

```
IATXCSF—ENTER=label

address

,TERM=YES,address

,ASYNCH=YES,address

,NORMAL=address

,DESC=address

```

### Parameters

**ENTER=**

Specifies the address of the routine to be given control when the JES3 subtask is dispatched.

**ID=**

Specifies a number (1 through 255) to identify the appendage. This number can be used by independent calls to ensure that the same task control block (TCB)
is used for logically coupled appendage executions (for example, OPEN and CLOSE). Subtasks 1 through 150 are reserved for JES3; subtasks 151 through 255 are reserved for users.

JES3 subtask usage is documented in the Generalized Subtask Global Data Area (IATYGSIG) within module IATGRGS. Users are recommended to document their usage of a specific subtask in the subtask index to avoid conflict.

**Note:** The ID= keyword is required when you specify the TERM= keyword.

**TERM=**

Specifies that the subtask is to end when the appendage returns.

**Notes:**

1. The TERM= parameter is ignored if the ID= parameter is not specified.
2. When coding the TERM= parameter, you should be aware that your level of system performance can decrease.
3. If you specify the ASYNCH= parameter and the TERM=YES parameter, TERM=YES is ignored.

**NORMAL=**

This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

**ASYNCH=**

Use this parameter to allow an FCT that has invoked a subtask to continue running while the subtask processes work asynchronously.

Users of the ASYNCH= parameter should not pass control to an appendage routine that relies on register 11 (the FCT pointer). The FCT might have been deleted before the appendage gets control. Registers 2 through 10 and 13 can be used, but the appendage should ensure the registers point to storage that still exists. The appendage can use register 12.

**DESC=**

Use this parameter to specify a 16 byte description of the function being subtasked. The description may be specified as follows:

- **Label**
  - the field at the specified label contains the 16 byte description.

- **register**
  - the register contains the address of the 16 byte description. R0 and R1 cannot be specified. If ECFMASK is specified, R2 cannot be specified.

- **character string**
  - the description is enclosed in quotation marks and can be from 1 to 16 bytes long. The IATXCSF macro will pad the string with blanks if it is not 16 bytes long.

**Note:** An example of the IATXCSF macro appears in "Restrictions on MVS Services Available to DSPs and Installation Exit Routines" on page 27.

**Linkage**

Upon entry to the appendage, registers 2 through 13 are unchanged; register 14 contains the address to which the routine must return.
IATXCSF

On return to the routine issuing the IATXCSF macro, registers 0, 1, and 15 are the same as when the appendage completed execution. Registers 11 and 14 must not be changed, because register 11 contains the FCT pointer and register 14 contains the return address.

If you specify ASYNCH=YES, the appendage is entered asynchronously. Registers 2 through 13 are unchanged, but the values they point to might not be valid. Users of ASYNCH=YES should not rely on the contents of register 11 (the FCT pointer) and are responsible for validating the storage to which registers 2 through 10 and 13 point.

Restrictions

Do not use this macro in any routine that might run under the JES3 auxiliary task.

IATXCUE (Insert Reentrant Installation Exit Linkage)

The IATXCUE macro generates reentrant installation exit linkage into mainline code using only a single source statement. Using this macro is the recommended procedure for inserting reentrant installation exit linkage.

Syntax

```
IATXCUE

Called Routine

NO, PARM=(Rx)

ALOAD=YES, REGS

DCB=(R1)

LDERR=(Rx)

ACALL, LINKAGE=BALR

ERROR=(Rx)

NORMAL=(Rx)

Called Routine

EP=modname

EPLOC=UXL

RX= symbol

ADCON

VCON
```

270  z/OS V1R11.0 JES3 Customization
REGS:  
\[ \text{REGS} = (R_x, R_y, R_z) \]

Parameters

**EP=**  
Specifies the module or entry point (modname) or the routine name if the symbol is an internal or external symbol.

**EPLOC=**  
Specifies that the module entry point is to be loaded from the installation exit list which is mapped by the IATUXL macro, the specified register, the appropriate DC instruction (if ADCON or VCON is specified), or the location represented by the symbol. If EPLOC=UXL is coded, the IATYTVT, IATYREG, and IATUXL mapping macros are required.

**ALOAD=**  
Specifies whether linkage to the module is using the ALOAD/ADELETE macros.

If ALOAD=YES is specified, neither PARM=(R0) nor PARM=(R1) can be specified; REGS=(Rx) must be specified.

**DCB=**  
Specifies the location of the DCB parameter for the ALOAD macro.

**LDERR=**  
Specifies the error return address for the ALOAD macro.

**LINKAGE=**

- **ACALL**  
  Specifies that linkage to the module is using the ACALL macro. This linkage saves the contents of the registers across the linkage.

- **BALR**  
  Specifies that linkage to the module is using a BALR. The contents of the registers are not saved. If LINKAGE=BALR is specified, REGS=(R_x,R_y,R_z) must be specified.

**PARM=**  
Specifies the location where the parameters are passed to the called routine. If ALOAD=YES is specified, PARM=(R0) or PARM=(R1) cannot be specified.

**REGS=**  
Specifies the work register (Rx), return address register (Ry), and the entry point address register (Rz). The work register must be specified if the ALOAD=YES and ERROR= parameters are specified. The return address register and entry point address register are required if LINKAGE=BALR is specified.

**ERROR=**  
This parameter specifies the address to which the macro processing routine returns if an error occurs. When control is passed to the address, register 0 contains a dump code and register 15 contains a return code, unless otherwise specified.
IATXCUE

If an error occurs and ERROR was not specified, the dynamic support program (DSP) will be ended by the FAILDSP macro.

**NORMAL**

This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

**Linkage**

If the LINKAGE parameter specifies ACALL or is allowed to default, this macro uses ASAVE linkage. If the LINKAGE parameter specifies BALR, the macro issues a BALR 14,15 for linkage.

If you use the IATXCUE macro with BALR linkage, you must return to the ERROR return with a simple branch return (+0), and return to the NORMAL return with a displacement of 4 (+4).

**Restrictions**

Do not use this macro in any routine that might run under the JES3 auxiliary task.

**IATXCUE (Convert EBCDIC Number to Binary)**

The IATXCUE macro converts an EBCDIC decimal value (characters 0 to 9 only, and up to 8 bytes long), to a binary number, and returns it in register 1.

**Syntax**

```
/IATXCUE LENGTH= (reg) constant address , DATA= (reg) address , ERROR= (reg) address , WRKAREA= (reg) address , DATALOC= loctr_name
```

**Parameters**

**LENGTH**

Specifies the number of EBCDIC bytes to be converted. The length can be specified as contained in a register, as a constant, or as the address of a fullword containing the length. Only register 0, or registers 2-9 may be used. This parameter is required. The LENGTH parameter verifies that the length is no longer than 8 bytes. If it is, the error return is taken.

**DATA**

Specifies the address of the EBCDIC number to be converted to binary. The address can be provided in a register or as an address label. Only registers 1-9 may be used. This parameter is required.

If the DATA parameter does not point to an EBCDIC number, the error return is taken.
Note: The DATA= and LENGTH= keywords are checked only if the error return is specified. See the ERROR= keyword.

ERROR=
Specifies the address to return to in case the data is in the wrong format, or is more than 8 bytes long. The address can be provided in a register or as an address label. Only registers 2-9 may be used. If the subparameter R is specified with a label, a relative branch instruction is generated.

If you omit the ERROR= keyword, the length and data values are not verified before the data is converted. JES3 assumes that the length and data are correct.

WRKAREA=
Specifies the address of an 8-byte work area. The address can be provided in a register or as an address label. Only registers 2-9 may be used.

If the WRKAREA parameter is not specified, the macro generates its own work area and generates non-reentrant code. Therefore you must specify WRKAREA= when issuing this macro in reentrant code.

DATALOC=
Specifies the name of a location counter to place objects of EX instructions generated by the macro. This is needed if the invoker does not have a base register for the code.

Linkage
This macro is an inline expansion macro.

Restrictions
This macro converts up to 8 bytes of an EBCDIC number. Registers 0, 1, 14, 15 are destroyed.

Example of Use
An example of the IATXCVB macro is:

```
.(other code)
LA     R2,L'CHKDATA
LA     R6,CHKDATA
IATXCVB DATA=(R6),       DATA TO CONVERT
      LENGTH=(R2),       LENGTH IS IN A REGISTER
      WRKAREA=FCTWORK,   AREA USED IN CONVERSION
      ERROR=BADDATA     ADDRESS OF ERROR ROUTINE
.(other code)

* Processing for bad data
BADDATA   DS   0H
.(other code)

* Data to be checked
CHKDATA   DC   CL8'00001234'
```

IATXCVD (Convert Binary to EBCDIC Decimal)
The IATXCVD macro converts a binary number to an EBCDIC decimal value.
IATXCVD

Syntax

```
IATXCVD INPUT=(reg), OUTPUT=(reg),

FCTWORK, WORKAREA=label, OUTLEN=number,

LEADING_ZERO=KEEP
```

Parameters

**INPUT=**
- Specifies the address of the binary number to be converted. If a register is specified, the register must contain the binary number to be converted. If a label is specified, the label must be the name of a 1, 2, or 4 byte field.

**OUTPUT=**
- Specifies the address of the area to contain the converted number. The converted number can be any size from 1– to 8–bytes.

**WORKAREA=**
- Specifies address of a work area to be used in the conversion. This area must be a double word.

**OUTLEN=**
- Specifies the length of the output area. If OUTPUT= is specified as a register, the default of OUTLEN is 8. If OUTPUT is specified as a label, the default of OUTLEN is the declared length of the label.

**LEADING_ZERO=**
- Specifies whether leading zeros in the OUTPUT area should be left zero or changed to blank.

Linkage

This macro expands in-line.

Restrictions

None.

Examples of Use

An example of the IATXCVD macro is:
```
  . (other code)
  IATXCVD INPUT=BINVALUE,
    OUTPUT=PRTVALUE,
    LEADING_ZERO=KEEP
  . (other code)
  . (other code)

* Data to be used
  BINVALUE DS H
  PRTVALUE DS CL5
```

274  z/OS V1R11.0 JES3 Customization
IATXDATX (Process DATCC with/without extension)

The IATXDATX macro is used to address user data following a DATCC with or without extension. It also provides the data length and logical record length of the current record. The logical record length returned is zero if DATCC extension (DATCCX) does not exist.

Syntax

```
IATXDATX(DATCC(address),DATLEN(number),DATL(NO=YES),DATP(NO=YES),LRECL(NO=YES))
```

Parameters

**DATCC=**

Specifies the address of the beginning of the record, including the DATCC prefix area.

**DATLEN=**

Specifies the data length explicitly. If you specify this parameter, IATXDATX will use this value for the length of data calculation instead of the DATLEN field in the DATCC. You might do this for one of the following reasons:

1. To get the length of the first part of a split record.
2. To get the maximum user buffer length while reading the record from spool.

SVTMUBLN and TVTMUBLN cannot be used when reading records from spool because records may or may not contain a DATCCX extension, which would reduce the amount of available room whenever the extension is present.

If the data length is greater than 32768, you cannot specify this parameter as a label; you must use a register.

**DATP=**

Specifies whether you want IATXDATX to return a pointer to the user data following the DATCC, and possibly the DATCCX. If you specify DATP=YES, the pointer is returned in register 0.

**DATL=**

Specifies whether you want IATXDATX to return the length of the user data. If you specify DATL=YES, the length is returned in register 14.

**LRECL=**

Specifies whether you want IATXDATX to return the original logical record length of the record, before truncation of trailing blanks. If you specify LRECL=YES, the original logical record length is returned in register 15.

Linkage

This macro expands in-line.
Restrictions

If DATCC= specifies a register, it cannot be register 0, 14 or 15.

If DATLEN= specifies a register, it cannot be register 0, 1, or 15.

Register 1 is destroyed by this macro.

Examples of Use

None.

IATXDEL (Dequeue a Slot from the Active DJST Chain)

The IATXDEL macro dequeues the slot containing the specified step and DD pair from the active dynamic job summary table (DJST) chain. The macro then queues the slot to the beginning of the available DJST chain.

Syntax

```
/sm590000/ sm590000
iatxdel djst=label(address), step=(r1) address, dd=(r2) address, eod=address, error=address, navail=address
```

Parameters

**DJST=**

Specifies the address of the dynamic job summary table (DJST).

**STEP=**

Specifies the address of the step number to be dequeued.

**DD=**

Specifies the address of the ddname to be dequeued.

**EOD=**

Specifies the address of the exit to be taken if issuing this macro causes depletion of the active chain.

**ERROR=**

This parameter specifies the address to which the macro processing routine returns if an error occurs. When control is passed to the address, register 0 contains a dump code and register 15 contains a return code, unless otherwise specified.

If an error occurs and ERROR was not specified, the dynamic support program (DSP) will be ended by the FAILDSP macro.

**NAVAIL=**

Specifies the address of the exit to be taken if the specified step and DD pair are not found.

Linkage

This macro issues the IATZCALL macro without using ASAVE linkage.
Restrictions
Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task.

IATXDEQ (Dequeue a Block of Information)
This macro dequeues a block of information from the end of a single-threaded, first-in-first-out (FIFO) queue.

Syntax

```
IATXDEQ HDR=(reg) address, BLK=(reg), PTR=name, REF=name, EMPTY=(reg) address
```

Parameters

- **HDR=**
  Specifies the address of a fullword queue header.

- **BLK=**
  Specifies the register in which the address of the dequeued block is returned after the macro completes processing. If the queue is empty, zero is returned.

- **PTR=**
  Specifies the field name in the DSECT which represents the chain field for blocks on this queue.

- **REF=**
  Specifies the name of the DSECT used to describe blocks on this queue.

- **EMPTY=**
  Specifies the address to return to when the queue is empty.

Linkage
This macro expands as inline code. It uses registers 0 and 1 as work registers.

Restrictions
Do not use this macro in any routine that might run under the JES3 auxiliary task. Registers 0 and 1 cannot be specified in the macro.

IATXDEV (Convert a Device Number to 4-Digit Format)
The IATXDEV macro converts a device number to 4-digit format. Input to the macro is a device number. Output from the macro is a device number in 4-digit format.

The input device number can be in one of the following formats:
- A device number without delimiters:
  For example:
  DDD
  /DDDD
- A device number delimited by parentheses:
IATXDEV

For example:

  DDD)
  (DDDD
  (/DDD)

- A device number delimited by a comma:
  For example:
  DDDD,
  /DDD,

- A device number delimited by some combination:
  For example:
  (DDD,
  (/DDDD),

Syntax

```
IATXDEV—INDEV — (reg) ,OUTDEV — (reg)
```

```
,DEVLEN — (reg) ,ERROR — (reg)
```

```
,NORMAL — (reg)
```

Parameters

**INDEV**

Specifies the name of a field containing the input device number, or a register containing the address of a field containing the device number.

**OUTDEV**

Specifies a four byte field in which the converted device number is placed. Can be specified as a label or as a register containing the address of the output area.

**DEVLEN**

Specifies the length of the input device number. Can be specified as the label of a halfword field containing the length, or as a register containing the length.

If INDEV indicates a field that is only four characters long, DEVLEN must be specified.

If DEVLEN is not specified, then INDEV must be a field or a register that points to a field that contains the device number followed by at least one blank. The field can be up to eight bytes long. The IATXDEV macro determines the length of the device number if DEVLEN is omitted.

**ERROR**

Specifies the address to which control is passed if an error occurs during processing. Can be specified as a label or a register containing the address of where processing should continue.
NORMAL
The address to which control is passed after normal completion. Can be specified as a label or a register containing the address of where processing should continue.

Linkage
This macro expands inline code.

Restrictions
None.

IATXDPL (Delete Cell Pool/Extent)
The IATXDPL macro deletes a cell pool, a secondary extent, or all secondary extents of a cell pool. The cell pool or extent(s) must have been created using the IATXBPL macro.

Syntax

```
IATXDPL(label),CPADDR=(R1)(reg),CPB=(R0)(reg),NORMAL=address,ERROR=address
```

Parameters

**CPADDR=**
Specifies the primary cell pool control block (CPB) from which the requested cell pool or extent(s) is to be deleted.

**CPB=**
Specifies the address of the extent to be deleted. This parameter is valid only when EXTENT=SINGLE is specified.

**EXTENT=**

- **SINGLE**
  Specifies that the secondary cell pool extent specified in the CPB parameter is to be deleted.

- **ALL**
  Specifies that all secondary cell pool extents for the specified primary CPB are to be deleted.

**NORMAL=**
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.
IATXDPL

**ERROR=**

This parameter specifies the address to which the macro processing routine returns if an error occurs. When control is passed to the address, register 0 contains a dump code and register 15 contains a return code, unless otherwise specified. The return codes are described below.

If an error occurs and ERROR was not specified, the dynamic support program (DSP) will be ended by the FAILDSP macro.

**Return Codes**

Following are the error return codes returned in register 15:

<table>
<thead>
<tr>
<th>Return Code</th>
<th>In Hex</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>04</td>
<td>04</td>
<td>The address specified in the CPADDR parameter does not point to a primary CPB.</td>
</tr>
<tr>
<td>08</td>
<td>08</td>
<td>The address specified in the CPB parameter does not point to a secondary CPB.</td>
</tr>
<tr>
<td>0C</td>
<td>0C</td>
<td>The cell pool or extent still contains active cells. No changes are made to the chain structure if the extent contains active cells.</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>The CPB parameter specified the primary CPB address, yet EXTENT=SINGLE was specified. EXTENT=SINGLE symbol specifies a secondary extent address.</td>
</tr>
</tbody>
</table>

**Linkage**

This macro uses ASAVE linkage.

**Restrictions**

Do not use this macro in any routine that might run under the JES3 auxiliary address space.

**IATXDST (Validate a Destination)**

The IATXDST macro checks whether a destination is defined to JES3 as a writer queue destination when processing inbound NJE sysout data. To be defined as such, the destination must not be defined as a USERID on any DESTDEF initialization statement, and it must be one of the following:

- LOCAL or ANYLOCAL
- A destination defined as a DEVICE on a DESTDEF initialization statement
- A printer or punch device name, device type, group name, or device number
- A remote workstation name, group name, or device name

**Note:** If the destination matches a destination defined on a DESTDEF initialization statement as a USERID, IATXDST will treat it as if it were not a known JES3 destination regardless of whether the name matches a print or punch device or a remote workstation.

**Syntax**

```
IATXDST DEST address, NAVAIL = address (Rx)
```


280  z/OS V1R11.0 JES3 Customization
Parameters

DEST=
Specifies the address of an eight-byte field containing the name of the destination to be checked. The destination must be left justified and padded with blanks.

NAVAIL=
Specifies the address to branch to when the destination is not a known JES3 device.

NORMAL=
Specifies the address to branch to when the destination is a known JES3 device. The default is to continue processing at the next sequential instruction after the macro.

Linkage

This macro uses ASAVE linkage.

Restrictions

Do not use this macro in any routine that might run in a C/I FSS address space or under a generalized subtask or the JES3 auxiliary task.

Note: IATXDST will not check whether the destination is a networking node name. Use the IATXNTS macro to validate node names.

IATXDYH (Performance Measurement Hook)

The IATXDYH macro provides the capability for a performance measurement in module IATSICA, which runs in the user address space. The sampling routine must be supplied by the user. The user must also place the sampling routine address in field SVTSAMPA in the IATYSVT data area.

Syntax

```
label IATXDYH
   TIME= N
   TYPE= ALL
   SVT= (reg)
   address
```

Parameters

TIME=
IATXDYH

\[\text{TIME=\textbf{Y}}\] Indicates that the sampling routine should measure the elapsed time because the preceding use of the IATXDYT macro. TIME=\textbf{Y} is valid only for TYPE=JES3.

\[\text{TIME=}\textbf{N}\] Indicates that the sampling routine should not measure the elapsed time because the preceding use of the IATXDYT macro.

\[\text{TYPE=}\]

\[\text{\textbf{ALL}}\] Indicates that all types of hooks (other than JES3 and NOJST) should be taken.

\[\text{JES3}\] Indicates that the hook should be taken upon return from the JES3 global processor.

\[\text{NOJST}\] Indicates that the hook should be taken in the common allocation routine when no matching job summary table (JST) entry is found.

\[\text{SVT=}\] Specifies the address of the subsystem vector.

**Note:** The IATXDYH macro should be issued with \text{TIME=}\textbf{Y} and \text{TYPE=}\textbf{JES3} following an SSISERV call to the JES3 global processor to measure elapsed time.

**Linkage**

This macro issues a BALR R14,R15 for linkage.

**Restrictions**

Do not use this macro in any routine that might run under the JES3 auxiliary task.

**IATXDYT (Performance Measurement Timing)**

This macro provides the capability for performance measurement timing in module IATSICA by taking a time stamp. The time stamp is saved in field AWAOTIM of the IATYAWA data area.

**Syntax**

```
IATXDYT
    label (R1)
    SEL= (reg)
        address
```

**Parameters**

\[\text{SEL=}\]

Specifies the address of the service entrance.

**Linkage**

This macro expands as inline code.
Restrictions

The calling routine must establish addressability to the IATYAWA data area. Do not use this macro in any routine that might run under the JES3 auxiliary task.

IATXELA (Add ECF to the ECF List)

The IATXELA macro adds an ECF mask and address to the ECF list control block. The ECF list control block, referred to as the ELB, is mapped by the IATYELB macro.

The ECF entry address is returned in register 0 and the relative ECF position is returned in register 1.

Syntax

```
IATXELA
label
ELB = (R2)
(reg) address

ECFMASK = (R1)
(reg) hex
symbol
hex

ECF = (R0)
(reg) address

ERROR = address

NAVAIL = address
NORMAL = address
```

Parameters

**ELB=**

Specifies the address of the ELB for the ECF list to which the ECF mask and address is to be added.

**ECFMASK=**

Specifies a one-byte mask indicating which bits in the ECF are to be waited on. This parameter can be specified as a decimal value, a hex value, a symbol, or as a register. If a decimal (shown by nnn) is specified, it must be a value less than 256. If a hex expression is used, the mask must be specified in the form X'hexexp'. If a symbol is used, it must be equated to a one-byte ECF mask value. The symbol is not the address of the ECF mask. If a register is specified, the register must contain the one-byte mask value in the low-order byte. The remainder of the register must be zero. Any register except R0 or R2 can be used.

If ECFMASK is coded, ECFADD must also be coded.

**ECFADD=**

Specifies the address of the ECF that is to be added to the ELB. If coded, the ECFMASK parameter must also be coded. Register 2 cannot be used for ECFADD.

**ECF=**

Specifies the address of a doubleword containing the ECF address in word 1,
and the ECF mask in the low-order byte of word 2. The first three bytes of word 2 must be zeros. The macro will load the ECF address into register 0 and the ECF mask into register 1.

Register 2 cannot be specified for ECF=. If ECF= is coded, ECFADD and ECFMASK cannot be coded.

**ERROR**=
This parameter specifies the address to which the macro processing routine returns if an error occurs. When control is passed to the address, register 0 contains a dump code and register 15 contains a return code, unless otherwise specified. The return codes are described below.

If an error occurs and ERROR was not specified, the dynamic support program (DSP) will be ended using the FAILDSP macro.

**NAVAIL**=
Specifies the address to which the routine returns if no ECF entries are available.

**NORMAL**=
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

**Return Codes**
Following are the error return codes returned in register 0:

<table>
<thead>
<tr>
<th>Return Code</th>
<th>In Hex</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>04</td>
<td>04</td>
<td>Incorrect ELB identifier</td>
</tr>
<tr>
<td>08</td>
<td>08</td>
<td>Allowed ECF count exceeds maximum</td>
</tr>
</tbody>
</table>

**Linkage**
This macro issues the IATZCALL macro without using ASAVE linkage.

**Restrictions**
Do not use this macro in any routine that might run under the JES3 auxiliary task.

**IATXELD (Delete ECF from ECF List)**
The IATXELD macro deletes an ECF mask and address from an ECF list.

**Syntax**

```
IATXELD label (R1) address, ECFPOS = (R0) nn
```

```
-, ERROR = address, NORMAL = address
```
Parameters

ELB=
Specifies the address of the ECF list control block (ELB) for the ECF list from which the ECF is to be deleted.

ECFPOS=
Specifies the relative position of the ECF mask and address to be deleted from the ELB.

ERROR=
This parameter specifies the address to which the macro processing routine returns if an error occurs. When control is passed to the address, register 0 contains a dump code and register 15 contains a return code, unless otherwise specified. The return codes are described below.

If an error occurs and ERROR was not specified, the dynamic support program (DSP) will be ended using the FAILDSP macro.

NORMAL=
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

Return Codes
Following are the error return codes returned in register 0:

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>04</td>
<td>Incorrect ELB identifier</td>
</tr>
<tr>
<td>08</td>
<td>Relative position exceeds maximum</td>
</tr>
<tr>
<td>0C</td>
<td>Relative position negative</td>
</tr>
<tr>
<td>10</td>
<td>Relative position not in active ECF list</td>
</tr>
</tbody>
</table>

Linkage
This macro issues the IATZCALL macro without using ASAVE linkage.

Restrictions
Do not use this macro in any routine that might run under the JES3 auxiliary task.

IATXELS (Scan ECF List for Satisfied ECF)
The IATXELS macro scans an ECF list for a satisfied ECF mask and address.

The ECF entry address is returned in register 0 and the relative ECF position is returned in register 1.

Syntax

```
IATXELS label ELB=(R1) address ECFPOS=(R0) m
```
Parameters

**ELB**
Specifies the address of the ECF list control block (ELB) for the ECF list to be scanned for a satisfied ECF.

**ECFPOS**
Specifies the relative position of the ECF mask and address from which the scan of the ELB will begin; the default is zero.

**TYPE**
- **ON** Specifies that the caller is using AWAIT TYPE=ON macro logic.
- **OFF** Specifies that the caller is using AWAIT TYPE=OFF macro logic.

**SCAN**
- **ALL** Specifies that the caller is testing all entries starting with the entry specified by the ECFPOS parameter for a satisfied ECF.
- **ONE** Specifies that the caller is testing just the entry specified by the ECFPOS parameter for a satisfied ECF.

**NAVAIL**
Specifies the address to which the routine returns if no satisfied ECF entry is found.

**ERROR**
This parameter specifies the address to which the macro processing routine returns if an error occurs. When control is passed to the address, register 0 contains a dump code and register 15 contains a return code, unless otherwise specified. The return codes are described below.

If an error occurs and ERROR was not specified, the dynamic support program (DSP) will be ended using the FAILDSP macro.

**NORMAL**
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

Return Codes
Following are the error return codes returned in register 0:

<table>
<thead>
<tr>
<th>Return Code in Hex</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>04</td>
<td>Incorrect ELB identifier</td>
</tr>
</tbody>
</table>
### Return Code In Hex

<table>
<thead>
<tr>
<th>Hex</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>08</td>
<td>Relative position exceeds maximum</td>
</tr>
<tr>
<td>0C</td>
<td>Relative position negative</td>
</tr>
</tbody>
</table>

#### Linkage

This macro issues the IATZCALL macro without using ASAVE linkage.

#### Restrictions

Do not use this macro in any routine that might run under the JES3 auxiliary task.

### IATXENQ (Enqueue a Block of Information)

The IATXENQ macro enqueues a block of information to the front of an IATYSRT queue in SNA RJP. The IATYSRT macro maps the resident SNA RJP table.

#### Syntax

```plaintext
IATXENQ(label, HDR=(reg), address, BLK=(reg), PTR=, name=, REF=, POST=(flag, mask), JPOST, ERROR=(reg), address)
```

#### Parameters

- **HDR=** Specifies the address of a fullword queue header.
- **BLK=** Specifies the address of the block to be enqueued.
- **PTR=** Specifies the field name in the DSECT which represents the chain field for blocks on this queue.
- **REF=** Specifies the name of the IATYSRT DSECT used to describe blocks on this queue.
- **POST=** Specifies that the specified ECF be posted. The flag subparameter is an A-type address of an ECF. The mask subparameter specifies a symbol describing an absolute expression to be ORed with the ECF.
- **JPOST** Specifies that the JES3 task is to be posted.
- **ERROR=** This parameter specifies the address to which the macro processing routine
returns if an error occurs. When control is passed to the address, register 0 contains a dump code and register 15 contains a return code, unless otherwise specified.

If an error occurs and ERROR was not specified, the dynamic support program (DSP) will be ended using the FAILDSP macro.

Linkage

This macro expands as inline code. It uses registers 0 and 1 as work registers. If the POST parameter is specified, the macro also uses registers 3, 14, and 15.

Restrictions

If this macro is issued by a routine running in a C/I FSS address space, the macro cannot specify POST with the JPOST option. Registers 0, 1, 3, 14, or 15 must not be used in the macro.

IATXENT (Build an Entry in the Entry Table)

The IATXENT macro identifies SNA RJP module entry points which the driver module stores in the location specified in the address field. Registers cannot be used to pass the name or address field.

Syntax

IATXENT name, address, TYPE=

Parameters

name
Specifies the label at the beginning of the routine to be entered.

address
Specifies the label where the address of the routine can be found.

TYPE=

EOD
Specifies that this is the last entry and that the table is to be ended.

DSECT
Specifies that a map of the entry table is to be produced.

Linkage

This macro expands as inline code.

Restrictions

Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task.
The IATXERCK macro checks the return code from VTAM after issuance of any VTAM macro. If CHECK=YES, the IATXERCK macro results in issuing a VTAM CHECK macro on completion of asynchronous requests generated by the VTAM macros. The IATXERCK macro is valid for synchronous and asynchronous requests.

Syntax

```
IATXERCK RPL= (R1) (reg) address, CHECK= YES NO

MODE= RB DSP, RETRY= (reg) address, PERR= (reg) address

TERR= (reg) address, NORMAL= (reg) address
```

Parameters

**RPL=**
Specifies the address of the request parameter list (RPL) used for this request.

**CHECK=**

**YES**
Specifies that the VTAM CHECK macro is to be issued. Registers 0, 1, 14, and 15 are destroyed. CHECK=YES should only be specified to check the final completion of an asynchronous request.

**NO**
Specifies that the VTAM CHECK macro is not to be issued. Registers 15 and 0 must contain the return codes supplied by VTAM. The contents of registers 0, 1, 2, 14, and 15 are then destroyed.

**MODE=**

**RB**
Specifies that the caller is under interruption request block (IRB) or service request block (SRB) control.

**DSP**
Specifies that the caller is a DSP running under FCT control.

**RETRY=**
Specifies the address branched to if a retry of the VTAM request was successfully completed. For the OPNDST VTAM macro request, this address is branched to if a condition which can be tried again exists, but the error routine does not try again.

**PERR=**
Specifies the address branched to if a permanent error occurred. If a session exists, a message is issued and the session is ended. If a session does not exist, only a message is issued.
IATXERCK

TERR=
Specifies the address branched to if a temporary error condition is detected.
This parameter is valid only for VTAM RECEIVE macro requests.

NORMAL=
This parameter specifies that after the macro processing routine executes, the
issuing program can resume execution at an instruction other than the
instruction that immediately follows the macro.

Linkage
This macro uses ASAVE linkage.

Restrictions
Do not use this macro in any routine that might run in a C/I FSS address space.

IATXERCV (Correct Chaining FDBs)
The IATXERCV macro corrects the chaining of the file description blocks (FDBs) of
a chained single-record file (SRF) after a write error. Register 15 must contain the
error code from the AWRITE macro.

Syntax
```
/SM590000/SM590000
IATXERCV
label
FDB=(R1) address,
ORGFDB=(R0) address
/SM590000
12
,DISP=number,NORMAL=address,ERROR=address
/SM590000
```

Parameters

FDB=
Specifies the address of a work area containing the FDB for which the write
error occurred. The FDB work area must be 22 bytes long: the first 12 bytes are
the FDB itself; the next 4 bytes are the chain FDB sequence number; the next 4
bytes are the file ID; the last 2 bytes are the displacement from the start of
each record to the four-byte sequence number, if applicable.

If the record does not contain a four-byte sequence number, the last 2 bytes
must be zero.

ORGFDB=
Specifies the address of the first FDB for the file.

DISP=
Specifies the displacement from the start of each record to the FDB that points
to the next buffer (the “chain FDB”).

ERROR=
This parameter specifies the address to which the macro processing routine
returns if an error occurs. When control is passed to the address, register 0
contains a dump code and register 15 contains a return code, unless otherwise
specified. The return codes and dump codes are described below. For a
description of every dump code, see z/OS JES3 Diagnosis.
If an error occurs and ERROR was not specified, the dynamic support program (DSP) will be ended using the FAILDSP macro.

**NORMAL=**
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

**Return Codes**
Following are the return codes and dump codes returned in registers 15 and 0, respectively:

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Dump Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>04</td>
<td>nnn</td>
<td>FDB address is incorrect.</td>
</tr>
<tr>
<td>08</td>
<td>nnn</td>
<td>Bad buffer address in FDB. FDB was refreshed and reread.</td>
</tr>
<tr>
<td>0C</td>
<td>nnn</td>
<td>Bad buffer, FDB was refreshed and reread.</td>
</tr>
<tr>
<td>10</td>
<td>nnn</td>
<td>Corrected I/O error, ORGFDB was updated.</td>
</tr>
<tr>
<td>14</td>
<td>nnn</td>
<td>Permanent I/O error, chaining FDBs were truncated.</td>
</tr>
</tbody>
</table>

**Linkage**
This macro uses ASAVE linkage.

**Restrictions**
Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task. This macro uses register 2.

**IATXFIFO (First In/First Out Queue Manipulation)**
The IATXFIFO macro performs one of the following functions on a doubly threaded, double ended queue:
- Add an element to the end of the queue. Optionally, a task or address space can also be posted.
- Remove an element from the head of the queue.
- Remove a specific element from its present position in the queue.

All these functions are performed with serialization.

**Syntax**

```
/label--IATXFIFO--MF=L

/label--IATXFIFO--FUNC=ENQUEUE

/label--IATXFIFO--FUNC=DEQUEUE

/label--IATXFIFO--FUNC=ENQUEUE

/label--IATXFIFO--FUNC=DEQUEUE
```

Chapter 2. JES3 Macros 291
Parameters

`FUNC=`
The function to be performed.

**ENQUEUE**
Add an element to the end of a queue.

**DEQUEUE**
Remove an element from a queue. The ELEMADDR and DQELEM parameters determine whether the first element is to be removed or a specific element is to be removed.

**ELEMADDR=**
For the ENQUEUE function, ELEMADDR is an input parameter pointing to the element to be added.

For the DEQUEUE function, if the first element is to be removed, ELEMADDR is an output parameter containing either a register or the address of an area to receive the address of the first element in the queue.

ELEMADDR is mutually exclusive with DQELEM.

**DQELEM**
For the DEQUEUE function, DQELEM is an input parameter containing a pointer to a specific element to be deleted from the queue. The element is not returned in an output parameter. ELEMADDR is mutually exclusive with DQELEM.

ELEMADDR is not valid with FUNC=ENQUEUE.

**HEADER=**
The pointer to the head (first element) of the queue.

**TAIL=**
The pointer to the tail (last element) of the queue.

**ELNEXT=**
The offset within the queue element of a field that contains a fullword pointer to the next element in the queue.

**ELPREV=**
The offset within the queue element of a field that contains a fullword pointer to the previous element in the queue.

**ECB=**
An optional pointer to an Event Control Block to post (for example, to notify a task of work represented by the element that is being added).

**ASCB=**
An optional pointer to an Address Space Control Block if a post is to be cross memory. If you specify ASCB=, you must also specify ECB=.

**ERROR=**
A label to branch to for one of the following conditions:
- FUNC=DEQUEUE is specified with ELEMADDR and the queue is empty.
• FUNC=DEQUEUE is specified with DQELEM and the element is not on the queue.

• FUNC=ENQUEUE is specified and the element pointed to by ELEMADDR is not valid. To be valid, the next and previous pointers of the element must both be zero and the element must not be the head or tail of the queue. This prevents an element from being added to the same queue more than once, or being added to more than one queue at a time with the same next and previous chain offsets.

As a convenience to callers, the DEQUEUE function sets the next and previous chain pointers to zero whenever an element is removed either from the head or by specifying DQELEM. Then if the element is subsequently added to another queue and the chain pointers do not get changed, the ENQUEUE function will not encounter this error.

ERROR= is optional for FUNC=ENQUEUE. If this parameter is omitted and the requested element is determined not to be valid, control continues at the next sequential instruction after the call to IATXFIFO but the requested element is not added. The IATXFIFO operation is ignored in effect. ERROR= is required for FUNC=DEQUEUE.

MF=
  Specifies the list or execution form of the macro. If MF=L is used, label is required. MF is required on all invocations, so every caller requires at least one MF=L invocation.

Note: Registers 0, 1, 10, 11, 12, 14, and 15 must not be used for any of the above parameters.

Linkage

This macro results in a call to a service routine using BALR linkage.

Restrictions

The caller must have addressability to IATYSVT.

IATXFLCB (Find an LCB)

The IATXFLCB macro finds a logical unit control block (LCB) for the specified session communication identifier (CID). Registers 0 and 1 must not be used for input.

Syntax

```
label IATXFLCB CID address,FOUND address

,NOFOUND address

address

(reg)

address
```

Chapter 2. JES3 Macros
IATXFLCB

Parameters

**CID=**
Specifies the register or address containing the CID of a logical unit (LU).

**FOUND=**
Specifies the return address when the required LCB is found.

**NOFOUND=**
Specifies the return address when the required LCB is not found.

Linkage

This macro expands as inline code.

Restrictions

Do not use this macro in any routine that might run in a C/I FSS address space.

IATXFRQ (Free a RESQUEUE Entry)

The IATXFRQ macro returns a resident job queue (RESQUEUE) entry to the correct RESQUEUE cell pool. The RESQUEUE entry must have been obtained previously using the IATXGRQ macro.

Syntax

```
/IATXFRQ label RQ=(reg) address,ERROR=address
```

Parameters

**RQ=**
Specifies the address of the RESQUEUE to be returned.

**ERROR=**
Specifies the address to which control returns if an error occurs. The macro service routine in module IATGRRQ returns the dump code in register 0 and the return code in register 15. If this parameter is not specified and an error occurs, the macro service routine saves the return code in register 2 and issues the FAILDSP macro with a dump code of DM103.
NORMAL=

This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

Linkage

This macro uses ASAVE linkage.

Restrictions

Do not use this macro in any routine that might run under the JES3 auxiliary task.

IATXFSS (Functional Subsystem Services)

The IATXFSS macro can:

- Start a functional subsystem (FSS) address space (TYPE=START)
- Abnormally end a FSS (TYPE=ABEND)
- Perform clean-up services for a FSS (TYPE=CLEANUP)
- Checkpoint one or all FSS table entries (TYPE=CHKPT)
- Assign a device number to a device supported by an output writer FSS and check for ambiguous device numbers (TYPE=AMBCHECK)
- Establish an FSS environment (TYPE=FSSSTART) by starting the FSS controller DSP (named FSSCONT)
- Notify all active FSAs in the specified FSS of an event (TYPE=FSAPost)

The service routines for this macro are in module IATGRFS.

If you write your own DSP to create an FSS, this macro will be helpful in establishing the FSS environment.

Syntax

```
/SM590000/SM590000
label: IATXFSS
  TYPE=start
  ABEND
  CLEANUP
  CHKPT
  AMBCHECK
  FSSSTART
  FSAPost
/SM590000/SM590000

START:

  | START | FSS Keyword | PARM=(Rx) | address | AWAIT=NO |
```

Chapter 2. JES3 Macros 295
Parameters

**TYPE=**  
Specifies the type of service to be performed.

- **START**  
  Specifies that the FSS address space identified by the FSS= parameter is to be started.

- **ABEND**  
  Specifies that the FSS address space identified by the FSS= parameter is to be abnormally ended. All active FSAs for that FSS only will be posted to end immediately.

- **CLEANUP**  
  Specifies a request to reinitialize and checkpoint a specific FSA when used with the FSA= parameter. When used with the FSS= parameter, the request is for the FSS and all of its FSAs.

- **CHKPT**  
  Specifies that the FSS table entry identified by the FSS parameter is to be checkpointed; or, if the FSS=ALL parameter is specified, all FSS table entries are to be checkpointed.

- **AMBCHK**  
  Specifies that a device number is to be assigned to a device supported by an output writer FSS. Ambiguous device number processing should also be performed. If an ambiguous device number is created or removed by this request, the service routine issues informational messages.

  You specify the device number by using the SUP parameter. If the SUP parameter is omitted, the macro service routine performs device number assignment and device number ambiguity processing for all SUPUNITS devices that can use the FSS table entry specified by the FSS parameter.

- **FSSSTART**  
  Specifies a request to establish an FSS environment. It starts the FSS controller DSP. An FSA DSP (such as a writer DSP) should use this option before activating its FSA. If you specify TYPE=FSSSTART, you must also specify the FSS= and ERROR= parameters.

- **FSAPOST**  
  Specifies a request to post all active FSAs in the specified FSS. If you specify TYPE=FSAPOST, you must also specify the FSS=, ERROR=, and ECFMASK= parameters.

  As a result of TYPE=FSAPOST, all active FSAs (FSAALLOC) will have FSASTAT3 updated with the ECFMASK value, and the value in FSAECFP'T will point to the area that will be updated with FSAECFMK. If the post request includes FSATERM, then the high-order byte of register 0 will contain the reason for the termination request. This value will be placed in FSATMRSN for use by the FSA DSPs in determining which error messages to issue.

**FSS=**  
address
Specifies the address of the FSS table entry for the required FSS address space, or the register containing that address. Register 1 is the default.

**ALL**

Specifies all entries in the FSS table. This parameter is valid only with the TYPE=CHKPT parameter.

**FSA=**

Specifies the address of the FSA table entry. Either this parameter or FSS=, but not both, must be specified with a TYPE=CLEANUP request. Register 1 will be updated with the specified value.

**PARM=**

Specifies the address of a parameter string to be included as part of the MVS START command that JES3 uses to start the FSS address space. The parameter string consists of a byte specifying the length of the list, followed by a maximum of 20 alphameric characters; the only special character allowed is the comma. This parameter is valid only with the TYPE=START parameter.

**SUP=**

Specifies the address of a SUPUNITS table entry for the device to be assigned a device number. This parameter is valid only with TYPE=AMBCHK.

**AWAIT=**

Specifies whether an AWAIT macro may be issued while a request to start an FSS address space is being processed. If a routine specifies AWAIT=NO, that routine is responsible for monitoring the progress of the start request. That routine is also responsible for reissuing the IATXFSS TYPE=START macro, if necessary. Field FSSSTAT1 in the FSS table entry for the FSS address space provides the information necessary to monitor the start request.

This parameter is valid only with the TYPE=START parameter.

**DUMP=**

Specifies whether a dump of the FSS address space is to be taken when the address space abnormally ends. This parameter is valid only with the TYPE=ABEND parameter.

**ERROR=**

Specifies the address to which the macro processing routine returns if an error occurs. When control is passed to the address, register 0 contains a dump code and register 15 contains a return code, unless otherwise specified.

If an error occurs and ERROR was not specified, the FAILDSP macro ends the dynamic support program (DSP).

**Note:** The ERROR= parameter is required on all IATXFSS requests except the TYPE=AMBCHK request, for which it must not be used.

**ECFMASK=**

Specifies the mask, or a register containing the mask, to be used to post each active FSA. Register 0 will be updated with the specified value. ECFMASK= is allowed only with a TYPE=FSAPOST request. The acceptable registers are 0, 2-9, and 13.

The value is a constant or a register with the value in the low-order byte. Acceptable constants are 1-byte hexadecimal expressions or decimal numbers between 0 and 255, or a label equated to the same.
For the termination reason, a value is expected in the high-order byte of the register used for the ECFMASK value. Acceptable values are constants equivalent to the equated values in the IATYFSS table for the start FSS error processing.

**Linkage**

This macro uses ASAVE linkage.

**Restrictions**

This macro may be used only by routines that run in the JES3 global address space.

---

### IATXFSV (Free a Save Area)

The IATXFSV macro returns a save area to the save area pool. This save area must have been previously obtained using the IATXGSV macro.

**Syntax**

```
IATXFSV label CHAIN= YES NO
```

**Parameters**

**CHAIN=**

- **YES**
  
  Specifies that the save area being returned to the save area pool is chained in the conventional MVS manner. Upon completion, register 13 points to the next higher save area from the save area returned. Register 15 is not restored.
  
- **NO**
  
  Specifies that the save area being returned to the save area pool is not chained in a higher save area. No registers are restored.

**Linkage**

This macro expands as inline code.

**Restrictions**

Do not use this macro in any routine that might run in a C/I FSS address space.

---

### IATXFWSB (Find Work Station Block)

The IATXFWSB macro finds the corresponding workstation block (WSB) or resident RJP line/terminal table (RJPTAB) for the specified SNA RJP workstation.

**Syntax**

```
IATXFWSB label NAME=(reg) address NOEXIST=(reg) address
```
IATXFWSB

Parameters

NAME=
Specifies the address of the workstation name.

NOEXIST=
Specifies the return address if the workstation name is not found.

BLOCK=

WSB
Specifies that the workstation block for the specified work station be found.

RLT
Specifies that the resident RJP line/terminal table (RJPTAB) for the specified workstation is to be found.

FOUND=
Specifies the return address for BLOCK=WSB when the WSB is found and is core resident. Register 1 contains the address of the WSB. For BLOCK=RLT, this is the return address. If the RJPTAB is found, register 1 contains the address of the RJPTAB.

NOFOUND=
Specifies the return address when the workstation name is known to JES3 but the WSB is not core resident. Register 1 contains the address of the associated RJPTAB. This parameter is required when the BLOCK=WSB parameter is specified. It is ignored when the BLOCK=RLT parameter is specified.

Restrictions

Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task.

IATXGCL (Allocate Cells From Cell Pool)

The IATXGCL macro allocates cells from a cell pool.

Syntax
Parameters

**CPADDR**=
Specifies the primary cell pool control block (CPB) address for the cell pool from which the requested cell is to be allocated.

**NAVAIL**=
Specifies the address to which control is passed if the cell pool cannot be expanded. This macro returns an ECF mask and address which may be used to wait until the cell pool can be expanded. The 31-bit ECF address is returned in register 0. The ECF mask needed is contained in the low-order byte of register 1, with the other three bytes containing zeros.

**RSVD**=
Specifies that the cell pool manager module, IATGRQC, is to allocate a reserved cell from the target cell pool’s primary extent when no primary extent non-reserved cells are available. NO is the default value.

**NORMAL**=
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

**ERROR**=
Specifies the address to which the macro processing routine returns if an error occurs. When control is passed to the address, register 0 contains a dump code and register 15 contains a return code, unless otherwise specified.

If an error occurs and ERROR was not specified, the dynamic support program (DSP) will be ended using the FAILDSP macro.

**Error Return Codes**
Following are the error return codes returned in register 15:

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>04</td>
<td>The address specified in the CPADDR parameter does not point to a primary CPB.</td>
</tr>
<tr>
<td>10</td>
<td>No cells are available but the total counter indicates that cells are available.</td>
</tr>
<tr>
<td>14</td>
<td>The bit may indicate that all cells are in use but the counter indicates that cells are available.</td>
</tr>
<tr>
<td>1XX</td>
<td>Return code from an internal call to macro IATXBPL.</td>
</tr>
</tbody>
</table>

**Return Codes**
Following are the return codes returned in register 15 if the cell pool cannot be expanded:

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>08</td>
<td>The cell pool cannot be expanded without exceeding the maximum extents defined for the cell pool. An ECF mask and ECF address are not returned when IATXGCL returns to the NOAVAIL exit with a return code of 8.</td>
</tr>
</tbody>
</table>
Return Code In Hex  Meaning
0C  The cell pool cannot be expanded because storage is not available in the user-specified subpool.

Linkage
This macro issues the IATZCALL macro without using ASAVE linkage.

Restrictions
Do not use this macro in any routine that might run under the JES3 auxiliary task.

IATXGENF (General Purpose FCT Functions)
This macro is used to perform a function under a General Purpose FCT. This is similar to the IATXCSF facility which allows functions to be performed under a generalized subtask.

Syntax
```
/SM590000/SM590000
label
IATXGENF
START
FUNC=
/SM590000/SM590000

Start Parameters:

,ROUTINE=(reg)
,DESC=(reg)
,label
,string
,MF=
,address

,CALLER=label
,NORMAL_ECF=(reg)
,label

,ERROR_ECF=(reg)
,REG0=(reg)
,label
,REG1=(reg)
,label
```

Parameters

**FUNC=**
Specifies the function to be performed.

**START**
This function is used to start a function under a general purpose FCT. When control returns, R0 contains a token to represent the IATXGENF request, and R1 contains the address of the general purpose FCT.

**ROUTINE=**
Specifies the address of the appendage routine to get control under the General Purpose FCT. A label or register can be specified. If the label form of this parameter is used, a load instruction is used to get the
address if the field associated with the label is an address constant. Otherwise, a load address instruction is used.

The appendage routine is called under the General Purpose FCT through the ACALL macro. The registers on entry to the appendage routine are as follows:

**R0**
Value specified for the REG0 parameter

**Note:** If the REG0 parameter was not specified, then the contents of R0 are unpredictable.

**R1**
Value specified for the REG1 parameter

**Note:** If the REG1 parameter was not specified, then the contents of R1 are unpredictable.

**R2-10**
Same as when the IATXGENF macro was issued

**R11**
General Purpose FCT address

**R12**
TVT address

**R13**
Same as when the IATXGENF macro was issued

**R14**
Return address

**R15**
Appendage entry point

The appendage should return to the caller by issuing an ARETURN macro, optionally specifying a return code in R15. If R15 is zero and the NORMAL_ECF parameter was specified, the normal ECF is posted. If R15 is zero and the NORMAL_ECF parameter is not specified, no posting is done. If R15 is non-zero and the ERROR_ECF parameter was specified, the error ECF is posted. If R15 is non-zero and the ERROR_ECF parameter is not specified, no posting is done.

**Note:** When the general purpose FCT gets control it sets up a JESTAE in case an abend occurs under the general purpose FCT. This does not preclude the appendage routine from setting up its own JESTAE to free resources acquired while running under the appendage routine. If an abend occurs under the general purpose FCT and the JESTAE set up by the general purpose FCT gets control, the ECF specified by the ERROR_ECF parameter will get posted, if it was specified.

**DESC=**
Specifies a 16 byte description of the function that is to be performed by the general purpose FCT.

**MF=**
Specifies whether this is the list or execute form of the macro.

**E,address**
The execute form is used and an existing parameter list specified by “address” is passed to the routine.

**L**
The list form of the parameter list is being generated.

**PRIORITY=**
Specifies the priority to be assigned to the general purpose FCT. This is used to determine where the FCT will be put on the FCT dispatching queue. If PRIORITY=0 is specified, the caller’s priority will be used.

**NORMAL_ECF=**
An optional parameter that specifies the ECF to be posted if processing
is successful. The first subparameter must be an ECF address, and the second subparameter must be an ECF mask.

**ERROR_ECF=**
An optional parameter that specifies the ECF to be posted if processing is unsuccessful. The first subparameter must be an ECF address, and the second subparameter must be an ECF mask.

**REG0=**
An optional parameter that specifies the value to be passed to the General Purpose FCT in R0. A label or register can be specified. If the label form of this parameter is used, a load instruction is used to get the address if the field associated with the label is an address constant. Otherwise, a load address instruction is used. If a register is used, R0 and R1 cannot be specified.

**REG1=**
An optional parameter that specifies the value to be passed to the General Purpose FCT in R1. A label or register can be specified. If the label form of this parameter is used, a load instruction is used to get the address if the field associated with the label is an address constant. Otherwise, a load address instruction is used. If a register is used, R0 and R1 cannot be specified.

**QUERY_ACTIVE**
This function is used to query the number of active general purpose FCT’s for the specified function that were started by the calling FCT. When control returns, register 15 contains the number of active general purpose FCT’s.

**DESC=**
Specifies a 16 byte description of the function that is associated with the query request.

**Linkage**
This macro uses ASAVE linkage.

**Restrictions**
Do not use this macro in any routine that might run under the JES3 auxiliary task.

**IATXGET (Get a DJST Slot)**
The IATXGET macro builds a step entry in an available slot in the dynamic job summary table (DJST), queues that slot to the active chain, and passes a pointer, in register 1, to the selected slot.

**Syntax**
```
IATXGET(DJST=(R0)label,STEP=(R1)address,NAVAIL=address,ERROR=address)
```

**Parameters**

**DJST=**
Specifies the address of the DJST.
STEP=
    Specifies the address of the step entry to be built into the available DJST slot.

NAVAIL=
    Specifies the return address to be taken if a slot is not available.

ERROR=
    This parameter specifies the address to which the macro processing routine
    returns if an error occurs. When control is passed to the address, register 0
    contains a dump code and register 15 contains a return code, unless otherwise
    specified.

    If an error occurs and ERROR was not specified, the dynamic support program
    (DSP) will be ended using the FAILDSP macro.

Linkage

    This macro issues the IATZCALL macro without using ASAVE linkage.

Restrictions

    Do not use this macro in any routine that might run in a C/I FSS address space or
    under the JES3 auxiliary task.

IATXGFC (Obtain an FCT for a DSP)

    The IATXGFC macro obtains an FCT on behalf of a DSP. The address of the FCT
    is returned in register 1. If the macro call specifies a DSP dictionary address, fields
    in the FCT are filled in with DSP-related information.

Syntax

```
IATXGFC DSP=address label
                string USECNT=NO

                BUSY=address
                NORMAL=address
                ERROR=address

                NAVAIL=address
                PERM=YES
                PREALLOC=NO

                R10=(reg)address
                R13=(reg)address

                DISPATCH=(reg)address
                PARM=(reg)address
```

Parameters

DSP= 
address
Specifies the address of a DSP dictionary entry to be used to initialize fields in the FCT.

'string'
Specifies a DSP name. The DSP entries will be searched for the specified name

USECNT=

YES
Specifies that the DSP use count is to be compared with the DSP maximum use count to determine whether a DSP is available. If no DSPs are available, the macro service routine issues an AWAIT macro or returns to the address specified by the BUSY parameter, if any. If a DSP is available, or when one becomes available, the macro service routine increases the DSP use count by one.

NO
Specifies that the DSP use count and the DSP maximum use count are not to be compared. The routine issuing this macro must check whether a DSP is available and increase the DSP use count itself. This option can be used when more than one DSP use count applies to a DSP (for example, the CI DSP counts batch DSPs and demand select DSPs separately).

If the USECNT parameter is specified (as either YES or NO), the DSP parameter must also be specified and the BUSY parameter cannot be specified.

NAVAIL=
Specifies the address to which control returns if no storage space is available for the FCT. If the caller wants to wait for storage to be available, see the BUSY= parameter on the AGETMAIN macro for the correct procedure.

NORMAL=
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

ERROR=
Specifies the address to which control returns if the specified DSP is not found. Only allowed if DISP='string' is specified.

If DSP name specified in DISP='string' is not found and ERROR= is not specified a DM100 abend is issued.

BUSY=
Specifies the address to which control returns if the use count of the specified DSP is at its maximum. The caller can wait for the count to be reduced by issuing the AWAIT TYPE=ON macro. The ECF address needed is returned in register 0. The ECF mask is contained in the low-order byte of register 1. The other three bytes of register 1 contain zeros.

If the BUSY parameter is omitted, the execution of the calling routine is suspended until the DSP’s use count is reduced. If the BUSY parameter is specified, the DSP parameter must also be specified and USECNT=NO cannot be specified.

R10=
Specifies the value that is to be put into register 10 (FCTBASE) when the FCT is first dispatched.
R13=
  Specifies the value that is to be put into register 13 (FCTSAVE) when the FCT is first dispatched.

DISPATCH=
  Specifies the address that the FCT will be dispatched when MFM gives it control. This value will be put into the register 14 field in the FCT (FCTRETN).

PARM=
  Specifies the parameter that is to be passed to the FCT in FCTATFPF.

PERM=
  Specifies whether the FCT is permanent. If PERM=YES is specified, the FCTPERM flag is set in the FCT entry.

PREALLOC=
  Specifies whether or not a preallocated FCT can be used to create this FCT.

Linkage
  This macro uses ASAVE linkage.

Restrictions
  Do not use this macro in any routine that might run under the JES3 auxiliary task.

IATXGFM (Execute Form: Issue an Error Message from a FSS Address Space)

The execute form of the IA4XGFM macro issues an error message from a functional subsystem (FSS) address space. This form of the macro refers to, and can modify, the list form of the macro.

Note: This macro has no standard form.

Syntax

```
IATXGFM MSGID=69nn, FSCBAD=address, FSDBAD=address
```

```
CALLER=list, ERRID=PARMERR, LISTEN, FSIORD, FSIPOST, FSISEND, FSACON, FSADCON, FSGOS, FSIROS, FSIGDS, FSIGREC, FSIFREC, FSIOMP, FSAQCELL, FSAWB, FSAQX, CIORDER
```

```
Parameters

**MSGID=**
Specifies the 4-digit message number. The first two digits must be 69.

**FSCBAD=**
Specifies the address of the FSS/FSA common control block, IATYFSCB, for the issuing FSS address space.

**FSDBAD=**
Specifies the address of the FSA data set control block, IATYFSDB, for the data set (if any) involved in the error.

**CALLER=**
Specifies the calling routine.

- **LISTEN**
  FSS or FSA listen task
- **FSIORD**
  ORDER FSI routine
- **FSIPOST**
  POST FSI routine
- **FSISEND**
  SEND FSI routine
- **FSACON**
  Writer FSA connect routine
- **FSADCON**
  Writer FSA disconnect routine
- **FSIGDS**
  GETDS FSI routine
- **FSIRDS**
  RELDS FSI routine
- **FSIGREC**
  GETREC FSI routine
- **FSIFREC**
  FREEREC FSI routine
- **FSICKPT**
  CHKPT FSI routine
- **FSAQCELL**
  Quick-cell service routine
- **FSASWB**
  SWB processing routine
- **FSARDA**
  Read-ahead task
- **FSACKW**
  Checkpoint writer task
- **CIORDER**
  C/I FSS order routine

**ERRID=**
Specifies the type of error.

- **PARMERR**
  Parameter error
- **BLKERR**
  Block spooler (module IATDMBS) error
BADRC
Non-zero return code

ERROR
Error (to be explained by the error text pointed to by the ERRTEXT parameter)

ERRTEXT=
Specifies the address of the optional error text. The first byte of the optional text must contain the length of the text (excluding the length byte).

MF=
Specifies that this is the execute form of the macro and specifies the address of the parameter list generated by the list form of the macro.

Linkage
This macro issues a BALR R14,R15 for linkage.

Restrictions
Issue this macro only from routines that run in a FSS address space.

**IATXGFM (List Form: Issue an Error Message from a FSS Address Space)**

The list form of the IATXGFM macro constructs a parameter list for issuing an error message from a functional subsystem (FSS) address space. The execute form of this macro can modify the values set by the list form.

The description of the execute form provides the explanation of each parameter and subparameter that is not described below.

**Note:** This macro has no standard form.

**Syntax**

```
(label) IATXGFM(MF=L, MSGID=69, nn)
```

```
"CALLER","LISTEN
"FSIORD
"FSIPOST
"FSISEND
"FSACON
"FSADCON
"FSIGDS
"FSIRODS
"FSIGREC
"FSIFREC
"FSAQCELL
"FSASWB
"FSARDA
"FSACKW
"CIORDER"
```
Parameters

MF=
  Specifies that this is the list form of the macro. This generates a parameter list required by the message module, IATFCMS. The format of the parameter list, with the fields in the following order, is:
  Two-byte message id (from the MSGID parameter) or zero
  One-byte caller id (equated from the CALLER parameter) or zero
  One-byte error id (equated from the ERRID parameter) or zero
  Fullword FSCB address (from the FSCBAD parameter) or zero
  Fullword FSDB address (from the FSDBAD parameter) or zero
  Fullword error text address (from the ERRTEXT parameter) or zero

Linkage

This macro issues a BALR R14,R15 for linkage.

Restrictions

Issue this macro only from routines that run in a FSS address space.

IATXGOSE (Add an Output Service Summary Entry)

The IATXGOSE macro adds an output service summary entry (OSS) and, if necessary, a storage resident output scheduling element (OSE) to the OSE scheduling matrix.

Syntax

Parameters

OSE=
  Specifies the address of a parameter OSE for which an MOSE/OSS is to be built.

TOKEN=
  Specifies the address of the security token to be placed in the IATYOSS. TOKEN= should not be specified if the OSE address is zero. Otherwise, TOKEN= is required.
TOKEN=NAVAIL is specified if the security token is not immediately available. In this case, the caller is responsible for filling OSSTOKEN with the security token before a writer scheduling a pass which attempts to use the OSSTOKEN field.

**RQ=**
Specifies the resident job queue (RESQUEUE) address of the job with which the OSE parameter is associated.

**QUE=**
Specifies the RESQUEUE status of the job. YES indicates that the job is on the writer queue and NO indicates that it is not. The RESQUEUE provided will not be associated with the OSS if NO is specified.

**NORMAL=**
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

**BUFNUM=**
Specifies the address of the sequence number (OSECNT4) of the OSE buffer that contains the OSE variable entry.

### Linkage

This macro uses ASAVE linkage.

### Restrictions

Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task.

### IATXGRQ (Get RESQUEUE Entry)

The IATXGRQ macro gets a resident job queue (RESQUEUE) entry and returns its address in register 1. JES3 quick cell services obtain the RESQUEUE entry from the proper cell pool on behalf of this macro. If this macro specifies a job control table (JCT) address, the macro service routine in module IATGRRQ initializes the RESQUEUE entry with data from the JCT.

### Syntax

```
IATXGRQ label, DSP=(reg), address, JCT=(reg)
```

### Parameters

**DSP=**
Specifies the address of the DSP dictionary entry for the DSP on whose behalf the RESQUEUE entry is being obtained. Different types of DSPs need different types of RESQUEUE entries, as defined below.
### JCT=
Specifies the address of the job’s JCT. The macro service routine will initialize the RESQUEUE entry with data from the specified JCT. If this parameter is not specified, the macro service routine does not initialize the RESQUEUE entry.

### NAVAIL=
Specifies the address to which control returns if no cells are available and either:
- Not enough storage remains to extend the cell pool
- The cell pool has reached the maximum number of extents

On return to the NAVAIL address, the registers contain:

- **R1**: The cell pool block address (data area IATYCPB) of the RESQUEUE cell pool
- **R15**: Return code from an internal call to the IATXGCL macro service routine. See the following return code descriptions.

If this parameter is not specified, the macro service routine assumes that the caller wants to obtain a RESQUEUE entry unconditionally. If no RESQUEUE is available, the macro service routine saves the return code in register 2 and issues a FAILDSP with a DM102 dump code.

### ERROR=
Specifies the address to which control returns if an error occurs. The macro service routine returns a dump code in register 0 and a return code in register 15. If this parameter is not specified and an error occurs, the macro service routine saves the return code in register 2 and issues the FAILDSP macro with a DM102 dump code.

### NORMAL=
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

### Error Return Codes
Following are the return codes returned in register 15 if the macro service routine takes the ERROR return:

<table>
<thead>
<tr>
<th>Return Code In Hex</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>04</td>
<td>The address specified in the CPADDR parameter of the IATXGCL macro does not point to a primary CPB.</td>
</tr>
<tr>
<td>10</td>
<td>No cells are available but the total counter indicates that cells are available.</td>
</tr>
<tr>
<td>14</td>
<td>The bit may indicate that all cells are in use but the counter indicates that cells are available.</td>
</tr>
<tr>
<td>1XX</td>
<td>Return code from an internal call to build pool.</td>
</tr>
</tbody>
</table>
**NAVAIL Return Codes**

Following are the return codes returned in register 15 if the macro service routine takes the NAVAIL return:

<table>
<thead>
<tr>
<th>Return Code</th>
<th>In Hex</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>08</td>
<td>08</td>
<td>The cell pool cannot be expanded without exceeding the maximum extents defined for the cell pool. (See below for possible actions to take.)</td>
</tr>
<tr>
<td>0C</td>
<td>0C</td>
<td>The cell pool cannot be expanded because storage is not available in the user-specified subpool. (See below for possible actions to take.)</td>
</tr>
</tbody>
</table>

If the return code is X'08', the issuer of the IATXGRQ macro can wait for an available RESQUEUE entry. The ECF on which to wait is field CPBFLAG2 in data area IATYCPB, whose address is returned in register 1. The ECF mask is bit CPBF2CAV in field CPBFLAG2.

If the return code is X'0C', the issuer of the IATXGRQ macro can wait either for an available RESQUEUE entry or for storage. In this case, the caller must set up 2 ECFs. The first ECF address and mask are the same as described above to wait for an available RESQUEUE entry. The second ECF is field FCTFLAG1 in the FCT, and the ECF mask is bit FCTAGET in field FCTFLAG1.

**Linkage**

This macro uses ASAVE linkage.

**Restrictions**

Do not use this macro in any routine that might run under the JES3 auxiliary task.

**IATXGSV (Get a Save Area)**

The IATXGSV macro obtains a save area from the save area pool.

The routine using this macro must set register 11 according to the SNA RJP convention. In this convention, the high-order bit of register 11 is used to indicate the type of address (in the low-order 31 bits) as follows:

- If the bit is off (0), it is an FCT address.
- If the bit is on (1), it is an IRB address or an SRB address.

To distinguish an IRB from an SRB, test the PSATOLD word in the PSA. If this word is zero, the SRB is active. If the word is not zero, the IRB is active.

**Syntax**

```
label IATXGSV CHAIN= YES NO
```

**Parameters**

CHAIN=
YES
The newly obtained save area is to be chained, in the conventional MVS manner, to the save area pointed to by register 13.

NO
Specifies that no chaining is to take place.

Linkage
This macro issues a BALR R14,R15 for linkage.

Restrictions
Do not use this macro in any routine that might run in a C/I FSS address space.

IATXHOME (Check Node Names)
This macro determines if a networking destination represents the home node, whether explicitly or by an alias definition. It also optionally changes a destination that is an alias to the home node.

Syntax
```
IATXHOME,label,IN=’name’,OUT=’name’,YES=’address’,NO=’address’
```

Parameters

IN=
Specifies the name of the input field to be checked for either the home node or an alias.

OUT=
Specifies the name of the output field to be changed if IN is found to be an alias. OUT can be the same as IN, but need not be. OUT=* indicates that the names of the input and output fields are the same. OUT is optional and if it is omitted, nothing gets changed. If IN is not an alias, but literally the home node, OUT does not get changed.

YES=
Specifies the address to branch to if the node is either home or alias. If the subparameter R is specified, a relative branch instruction is generated.

NO=
Specifies the address to branch to if the node is neither home or alias. If the subparameter R is specified, a relative branch instruction is generated.
UNDEF=
Specifies the address to branch to if the node to be checked for home or alias is not defined.

Linkage
This macro expands in-line.

Restrictions
Do not use this macro in any routine that might run in a C/I FSS address space, under the JES3 auxiliary task, or any address space other than JES3.

IATXICA (Access Method Control Block Processing)
The IATXICA macro updates the converter/interpreter’s access method control block (ACB).

Syntax

```
IATXICA
```

Parameters

ACB=
Specifies the address of the ACB to be updated.

SIZE=
Specifies the size in bytes required to build the ACB. Do not use the SIZE parameter when USE=I or USE=C.

ENTER=
Specifies the address of the routine to update the ACB.

USE=
Specifies whether to open (O), convert from output to input (I), or close the ACB(C). When USE=I or USE=C is specified, do not specify the SIZE parameter.

Linkage
This macro issues a BALR R14,R15 for linkage. Register 13 must point to the save area.

Restrictions
Do not use this macro in any routine that might run under the JES3 auxiliary task.
IATXIOE

IATXIOE (Recover From I/O CHECK Macro Error)

The IATXIOE macro attempts to recover from recoverable I/O errors by updating the anchor FDB. If the error is unrecoverable, a FAILDSP macro is issued.

Syntax

```
label IATXIOE DMPCD=(R0) address, DYQ=(R1) address
      , NORMAL=address
```

Parameters

- **DMPCD=**
  Specifies the dump code received from the I/O CHECK macro error.

- **DYQ=**
  Specifies the address of the dynamic allocation queue entries (DYQ) associated with the I/O CHECK macro error.

- **NORMAL=**
  This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

Linkage

This macro uses ASAVE linkage.

Restrictions

Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task.

IATXIOX (Check an SRF I/O Completion)

The IATXIOX macro checks the I/O completion status of a single-record file (SRF).

Syntax

```
label IATXIOX FDB=(R1) address, ID=name
      , ID=NONE, NAVAIL=address, ERROR=address
      , NORMAL=address
```

Parameters

- **FDB=**
  Specifies the address of the FDB for the SRF to be checked for I/O completion status.
ID=
  Specifies the name to be associated with the SRF. The ID length is limited to 4 characters. This applies to single-record reads only.

NAVAIL=
  Specifies the address to which the routine returns if the I/O for the FDB has not yet completed.

  The NAVAIL= parameter returns the ECF address in register 0 and the ECF mask in register 1 for AWAIT TYPE=ON requests.

ERROR=
  This parameter specifies the address to which the macro processing routine returns if an error occurs. When control is passed to the address, register 0 contains a dump code and register 15 contains a return code, unless otherwise specified. The return codes and dump codes are described below. For a description of every dump code, see z/OS JES3 Diagnosis.

  If an error occurs and ERROR was not specified, the dynamic support program (DSP) will be ended using the FAILDSP macro.

NORMAL=
  This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

Error and Dump Return Codes
  Following are the error return codes and dump codes returned in register 15 and 0, respectively:

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Dump Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>04</td>
<td>721</td>
<td>Incorrect track address</td>
</tr>
<tr>
<td>08</td>
<td>729</td>
<td>Spool space could not be allocated to a new single record file (SRF) because of a job or data set TAT error.</td>
</tr>
<tr>
<td>0C</td>
<td>724</td>
<td>Recovered I/O error, should checkpoint FDB</td>
</tr>
<tr>
<td>10</td>
<td>725</td>
<td>Unrecoverable I/O error</td>
</tr>
<tr>
<td>14</td>
<td>725</td>
<td>Error during ROOT I/O error recovery</td>
</tr>
<tr>
<td>18</td>
<td>704</td>
<td>IDs did not match</td>
</tr>
</tbody>
</table>

Linkage
  This macro issues the IATZCALL macro without using ASAVE linkage.

Restrictions
  Do not use this macro in any routine that might run under the JES3 auxiliary task.

IATXISMG (Write a Message to the JESMSG Data Set During Input Service)
  The IATXISMG macro enables you to write messages to the JESMSG data set for JCL and JES3 control card errors detected by input service installation exits. This ability is important because it allows you to capture error messages for jobs that are flushed because they violate installation restrictions. The IATXISMG macro does not require the job’s RESQUEUE in order to issue a message to the JESMSG data set. Instead it uses a pointer found in data CSECT IATYISD to locate the JESMSG data set.
IATXISMG

For a summary of JES3 macros involved in message processing, see the section “Choosing the Appropriate Macro to Issue a Message” on page 42.

Syntax

```
IATXISMG TEXT=address, ERROR=address
```

Parameters

**TEXT=**
Specifies the address of the message. The first byte of the message must contain the length of the message text, not to exceed 120 bytes. If the text length is greater than 120, it will be set to 120 and the message will be truncated. This parameter is required.

**ERROR=**
Specifies the address to which control is returned if an error is encountered while processing the request. This parameter is required.

**NORMAL=**
Specifies the address to which control is returned if the request completes normally. If NORMAL= is omitted, control returns to the next sequential instruction.

Linkage

None. This macro expands inline.

Restrictions

Do not use this macro in any routine that might run under the JES3 auxiliary task. This macro uses registers 0, 1, 2, and 15.

This macro works correctly only if the following conditions are true:
1. Addressability to the input service data set CSECT (IATYISD) has been established.
2. The JESMSG data set is open.

IATXIWT (Generate Specified Message)

The IATXIWT macro writes a message from the CI DSPs to either the system operator or the JESMSG data set. If an error occurs during JESMSG processing and the DSP is not in recovery mode, a FAILDSP macro is issued with dump code DM048, with the JESMSG return code in register 2.

For a summary of JES3 macros involved in message processing, see the section “Choosing the Appropriate Macro to Issue a Message” on page 42.

Syntax
Parameters

**TYPE=**

**OPER**
Specifies that the message will go to the console receiving messages with a destination class of JES. By default, this means going to the JES3 operator’s console.

**MOPER**
Specifies that the message is to go to both the JES3 operator and the JESMSG data set. In this case, its length may not exceed 120 characters.

**TITLE**
Generates the message with rows of asterisks above and below the message to draw attention to the message, and sends it to the JESMSG data set.

**SPACE**
Generates a line of blanks. To close the JESMSG data set when using this parameter, set field IDDETRC2 in the CI DSP data area (IATYIDD) equal to IDDJSML. The MSG and MSG# parameters are incorrect if this operand is coded.

**EQU**
Specifies that the message subroutine equates are to be generated. These equates are used by the IATXIWT service routine and are meant for IBM use only. The MSG and MSG# parameters are incorrect if this operand is coded.

**NONE**
Specifies that the message is to be added to the JESMSG data set. If NONE is coded or assumed, then the MSG or MSG# parameters must also be coded.

**MSG=**
Specifies the address of a nonstandard message that has been created by the user. The message must be in the form of a length byte followed by text. Messages cannot be longer than 132 characters, or they will be truncated. Variable user information can be inserted using embedded X’FA’s in the message text and the VARPTR and VARNO parameters in this macro. If this parameter is coded, then the MSG# parameter must not be coded. Also, TYPE=SPACE and TYPE=EQU are incorrect when this parameter is coded.

**MSG#=**
Specifies the number of the standard message within the message table. Standard messages are contained in module IATIIMS and are constructed to
allow variable user information to be inserted using the VARPTR and VARNO parameters. If this parameter is coded, then the MSG parameter may not be coded. Also, TYPE=SPACE and TYPE=EQU are incorrect when this parameter is coded. The message identified by this parameter must start with a 1-byte length field.

If the MSG# parameter value is greater than the largest predefined message, module IATIIMS calls installation exit IATUX10 to provide a message or to ignore the request for a message.

VARPTR=

Specifies the address of the list of variables to be inserted into the message. These variables allow you to insert user information into the message. The variables replace the X’FA’s embedded in the preconstructed standard messages or user-created nonstandard messages. Each parameter list entry is in the form of a length byte (with the length in hexadecimal) followed by the variable.

VARPTR= cannot be specified if TYPE=OPER is specified.

VARNO=

Specifies the number of variables to be inserted in the message or user-created nonstandard message. The maximum value that may be specified is 255. If this parameter is coded, the VARPTR parameter must be coded.

VARNO= cannot be specified if TYPE=OPER is specified.

Linkage

This macro uses ASAVE linkage.

Restrictions

Do not use this macro in any routines that might run under the JES3 auxiliary task. This macro uses register 2.

IATXJBN0 (Perform a job number operation)

The IATXJBN0 macro performs a load or store operation on 4–byte (extended) and 2–byte (compatible) job numbers. The IATXJBN0 macro can also be used on any value that represents the total of jobs that have some attribute in common (for example, the total number of jobs using a particular resource). A compatible job number or value refers to a value in a control block that is saved on spool or communicated between processors and therefore might be manipulated by a JES3 release that does not support job numbers greater than 65534.

Syntax

```
-IATXJBN0- FUNC=LOAD Func=STORE,REG=Rnn,J0BNO=(Rnn)label

,VER=,VERNO=n,FLAG_BYTE=,MASK=n,COMPAT_JOBNO=(Rnn)label

,VER=,VERNO=n,FLAG_BYTE=,MASK=n

,COMPAT_JOBNO=(Rnn)label

IATXJBN0 label

```

FUNC=LOAD:
Parameters

**FUNC**
Specifies the operation to be performed. This is a required parameter.

**LOAD**
The job number is to be loaded.

**STORE**
The job number is to be stored.

**REG**
Specifies the target register of a LOAD operation, or a source register for a STORE operation. This is a required parameter.

**JOBNO**
The address of the extended (4-byte) job number field that is the source for a LOAD operation or a target for a STORE operation. This is a required parameter.

**VER**
Specifies the one byte version number in the data area containing JOBNO and COMPAT_JOBNO in which JOBNO is defined only at a minimum version level. The parameter pairs VER/VERNO and FLAG_BYTE/MASK are mutually exclusive.

**VERNO**
Specifies the minimum version required in order for JOBNO to be defined. Verno and COMPAT_JOBNO must be specified whenever VER is specified.

**FLAG_BYTE**
Specifies the byte containing the flag or flags required to be set in order for JOBNO to be defined. The parameter pairs VER/VERNO and FLAG_BYTE/MASK are mutually exclusive.

**MASK**
Specifies the bits that must be on in FLAG_BYTE in order for JOBNO to be defined. MASK and COMPAT_JOBNO must be specified whenever FLAG_BYTE is specified.

**COMPAT_JOBNO**
Specifies the address of the compatible (2-byte) job number field that is:
- The source for a LOAD operation when the job number field specified by JOBNO does not exist or contains zero.
- The target for a STORE operation in conjunction with the field specified by JOBNO, or by itself if the JOBNO field does not exist.

The purpose of this parameter is so that data areas containing job numbers can be shared between releases of JES3 that support greater than 65,634 job
numbers and those that do not when the OPTIONS initialization statement defines a job number range whose upper limit does not exceed 65534.

COMPAT_JOBNO must be specified if FLAG_BYTE, VER, or COMPAT_FORMAT is specified. The way that JOBNO and COMPAT_JOBNO are used depends on whether FUNC is LOAD or STORE, and also depends on the presence of VER/VERNO or FLAG_BYTE/MASK.

If MOD is specified, the low two bytes of JOBNO are placed into COMPAT_JOBNO.

For a LOAD operation:
- If JOBNO is specified and COMPAT_JOBNO is not, the contents of JOBNO are loaded into REG.
- If JOBNO and COMPAT_JOBNO are specified and VER and FLAG_BYTE are omitted, the contents of COMPAT_JOBNO are loaded into REG if the value at the address specified by the JOBNO parameter is zero. Furthermore, the contents of COMPAT_JOBNO are copied to JOBNO.
- If either VER or FLAG_BYTE is specified and the JOBNO field is present because the conditions specified by VER/VERNO or FLAG_BYTE/MASK are met, the contents of JOBNO are loaded into REG.
- If either VER or FLAG_BYTE is specified and the JOBNO field is not present because the conditions specified by VER/VERNO or FLAG_BYTE/MASK are not met, no attempt is made to load from JOBNO. The contents of COMPAT_JOBNO are loaded into REG.

For a STORE operation:
- If JOBNO is specified and COMPAT_JOBNO is not, the contents of REG are stored in JOBNO.
- If JOBNO and COMPAT_JOBNO are specified and VER and FLAG_BYTE are omitted, the contents of REG are stored in JOBNO. If the job number is 65534 or less, it is copied to COMPAT_JOBNO. Otherwise, COMPAT_JOBNO is set to a special value of 65535.
- If either VER or FLAG_BYTE is specified and the JOBNO field is present because the conditions specified by VER/VERNO or FLAG_BYTE/MASK are met, the contents of REG are stored in JOBNO. If the job number is 65534 or less it is copied to COMPAT_JOBNO. Otherwise, COMPAT_JOBNO is set to a value that depends on the COMPAT_FORMAT parameter. If STANDARD is specified or allowed to default, a special value of 65535 is placed into COMPAT_JOBNO. If MOD is specified, the low two bytes of JOBNO are placed into COMPAT_JOBNO.
- If either VER or FLAG_BYTE is specified and the JOBNO field is not present because the conditions specified by VER/VERNO or FLAG_BYTE/MASK are not met, no attempt is made to store into JOBNO. The contents of REG are stored in COMPAT_JOBNO. The caller is responsible for making sure that the job number in REG does not exceed 65534 if this is the case.

COMPAT_FORMAT=
Specifies how a compatible job number is to be created for a STORE operation. If STANDARD is specified or allowed to default, a special value of 65535 is placed into COMPAT_JOBNO. COMPAT_FORMAT can only be specified when FUNC=STORE is specified. COMPAT_JOBNO must be specified if COMPAT_FORMAT is specified.

ZERO=
Specifies the address to branch to if the job number loaded is zero. ZERO can only be specified when FUNC=LOAD is specified.
NONZERO=
    Specifies the address to branch to if the job number loaded is not zero.
    NONZERO can only be specified when FUNC=LOAD is specified. ZERO and
    NONZERO can be specified together but need not be.

Linkage

This macro expands in-line.

Restrictions

Do not specify register 0 for JOBNO or COMPAT_JOBNO.

If you reference any of the fields that are defined as part of the programming
interface in JES3 and are either job numbers or job counts, you must use the
IATXJBNO macro to load or store them. Furthermore, if you reference any of the
fields in the below table, you must use additional parameters as shown.

<table>
<thead>
<tr>
<th>JOBNO</th>
<th>COMPAT_JOBNO</th>
<th>Other Required Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>BSIDJNUM</td>
<td>BSIDJNMC</td>
<td></td>
</tr>
<tr>
<td>DJSTJOBN</td>
<td>DJSTJOBMC</td>
<td></td>
</tr>
<tr>
<td>DLFJNUM</td>
<td>DLFJNUMC</td>
<td></td>
</tr>
<tr>
<td>FSSJOBNO</td>
<td>FSSJOBNC</td>
<td></td>
</tr>
<tr>
<td>INTJBLIM</td>
<td>INTJBLMC</td>
<td></td>
</tr>
<tr>
<td>JBTJNUM</td>
<td>JBTJNUMC</td>
<td></td>
</tr>
<tr>
<td>JDABJOBS</td>
<td>JDABJOBMC</td>
<td></td>
</tr>
<tr>
<td>JDSJOBNO</td>
<td>JDSJOBMC</td>
<td>FLAG_BYTE=JDSFLG8,MASK=JDSXJBNO</td>
</tr>
<tr>
<td>JSTFNCJB</td>
<td>JSTFNCJC</td>
<td></td>
</tr>
<tr>
<td>JVJTNUM</td>
<td>JVJTNUMC</td>
<td></td>
</tr>
<tr>
<td>NJHGJNO</td>
<td>NJHGJID</td>
<td>COMPAT_FORMAT=MOD (when FUNC=STORE is specified)</td>
</tr>
<tr>
<td>OSEJOBN</td>
<td>OSEJOBMC</td>
<td></td>
</tr>
<tr>
<td>OSESJBID</td>
<td>OSESJBC</td>
<td></td>
</tr>
<tr>
<td>OSRAPJBI</td>
<td>OSRAPJBC</td>
<td></td>
</tr>
<tr>
<td>WSPTEJBI</td>
<td>WSPTEJBC</td>
<td>VER=WSPVER,VERNO=WSPVER01</td>
</tr>
<tr>
<td>WSPCKJBI</td>
<td>WSPCKJBC</td>
<td>VER=WSPVER,VERNO=WSPVER01</td>
</tr>
</tbody>
</table>

IATXJBTS (Search a Job’s TATs)

The IATXJBTS macro scans all the track allocation tables (TATs) for a job or
spin-off data set to identify track groups allocated to particular spool data sets or
partitions. Before issuing this macro, the caller must set the IATYJTS data area to
zeros and also initialize the JTSID field with the control block identifier ‘JTS’.

Syntax

```
label IATXJBTS address
```
Parameters

**JTS**
Specifies the address of the job TAT search parameter list mapped by macro IATYJTS.

**EXT**
Specifies which track groups should be counted for each job TAT searched. The HELD, UNAVAIL, or DRAIN subparameters may be coded together by enclosing them in parentheses and separating them with commas.

**ALL**
All track groups for the job are to be counted.

**LIST**
Only track groups allocated to the job which are on spool data sets described by the EXTLIST parameter or the SPBLIST parameter are to be counted. If you specify this parameter, you must also specify the EXTLIST or SPBLIST parameter to identify the relevant spool data sets.

**HELD**
Only track groups allocated to the job that are on held spool data sets are to be counted.

**DRAIN**
Only track groups allocated to the job that are on drained spool data sets are to be counted.

**UNAVAIL**
Only track groups allocated to the job that are on unavailable spool data sets are to be counted.

**JCT**
Specifies the address of the job’s job control table (JCT).

**GRPS**
Specifies when the search should be ended.
End the search when all track groups that meet the search criteria are counted.

**FIRST**
End the search when the first track group that meets the search criteria is counted.

**TAT**
End the search after processing the job TAT chain or each spin data set TAT chain.

**EXTLIST=**
Specifies the address of a table representing each spool data set with a fullword entry. If you specify the EXT=LIST parameter, set to a nonzero value the entries representing spool data sets that you want counted. For all other EXT parameter options, the table entry for a spool data set will be increased by one for each logical track group that meets the specified criteria.

**SPBLIST=**
Specifies the address of a table representing each spool partition with a fullword entry. If you specify the EXT=LIST parameter, set to a nonzero value the entries representing spool partitions that you want counted. For all other EXT parameter options, the table entry for a spool partition will be incremented for each logical track group that meets the specified criteria.

**EOD=**
Specifies the address to which control returns when all of the job’s TATs are processed.

**ERROR=**
Specifies the address to which control returns when an error occurs.

**NAVAIL=**
Specifies the address to which control returns if the spool data set is unavailable or in hold status.

**NORMAL=**
Specifies the address to which control returns under normal conditions.

**Return Codes**
The following return codes are returned in register 15:

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>04</td>
<td>JCT address not supplied, or JCT supplied is incorrect.</td>
</tr>
<tr>
<td>08</td>
<td>You did not specify the JTS parameter, or the JTS supplied is incorrect.</td>
</tr>
<tr>
<td>0C</td>
<td>You specified EXT=LIST and did not specify the EXTLIST or SPBLIST parameter.</td>
</tr>
<tr>
<td>10</td>
<td>JBT or JDS FDB is incorrect.</td>
</tr>
<tr>
<td>14</td>
<td>JESREAD error for JDS or JBT.</td>
</tr>
<tr>
<td>18</td>
<td>Job has data on an unavailable spool data set.</td>
</tr>
<tr>
<td>1C</td>
<td>An incorrect X.G was encountered.</td>
</tr>
</tbody>
</table>

**Linkage**
This macro uses ASAVE linkage.
IATXJCT (Add, Alter, Examine, Delete, or Release a JCT)

The IATXJCT macro adds, alters, examines, deletes, or releases a JCT by invoking the JCT access method. The JCT access method accesses the JCT in the JCT data space whenever the data space is enabled. JES3 automatically enables the JCT data space during JES3 initialization. If the JCT data space is not enabled, the access method accesses the JCT from the JCT data set.

Syntax

```
IATXJCT TYPE=ADD Parms1
   CHKPT SOURCE=Parms2
   RW, SOURCE=JCTDS
   RO, SOURCE=JCTDS
   DEL, SOURCE=JCTDS
   REL, SOURCE=JCTDS
ERROR=address
BUSY=address
NORMAL=address
```

Parms1:

```
,JCT= (R1) (reg)
```

Parms2:

```
,JOBNUM= (R1) (reg)
,JQE= (R1) (reg)
```

SOURCE=:

```
,SOURCE= DSPACE
```

Parameters

**TYPE=**

**ADD**

Add a new JCT to the JCT data set. On return, register 0 contains the address of the JQE.
Access an existing JCT to allow the JCT to be read or written to (read/write access). On return, register 1 contains the address of the JCT, register 0 contains the address of the JQE.

Examine an existing JCT (read-only access). On return, register 1 contains the address of the JCT, register 0 contains the address of the JQE.

Delete an existing JCT (previous RW access must be obtained).

Relinquish access on a specific JCT (previous RW or RO access must be obtained).

Write the JCT to the JCT data set but do not relinquish control of the JCT. The buffer containing the JCT will not be freed, and the FCT will still be enqueued on the JCT. Previous read/write access must have been obtained to invoke this checkpoint function. JES3 does not update the JCT data space for TYPE=CHKPNT requests until after it writes the JCT to the JCT data set.

Specifies the binary job number for the JCT. A register, a 4 byte field, or a 2 byte field may be specified. If a 2 byte field is specified, it must be defined in the assembly before the macro invocation. This parameter is not valid with the TYPE=ADD parameter.

Specifies the register containing the address of the job queue element (JQE) for the job whose JCT is being manipulated. This parameter is not valid with the TYPE=ADD parameter.

Specifies the register containing the address of the JCT to be added to the JCT chain. This parameter is valid only with the TYPE=ADD parameter.

Specifies that the JCT should be obtained from the JCT data space. If the JCT data space is not enabled, JES3 obtains the JCT from the JCT data set.

Specifies that the JCT should be obtained from the JCT data set. This is the default if the JCT data space is not enabled. SOURCE=JCTDS should only be used for doing problem analysis when you need to access the copy of the JCT in the JCT data set.

This parameter can only be used with TYPE=RO or TYPE=RW requests. The default is SOURCE=DSPACE.

Specifies the address of a routine to be given control when errors are detected. On entry, register 15 contains an error return code, register 0 contains the address within module IATGRJX where the error was detected, and register 1 may contain the return code of a JSAM function which has returned in error (for example, JESREAD). A common error routine may be used for IATXJCT and IATXJQE macro calls.
BUSY=
Specifies the address of a routine to be given control if the JCT, or other resource needed, is not available. If BUSY is not specified, the routine will wait until the needed resource becomes available.

NORMAL=
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

Return Codes

<table>
<thead>
<tr>
<th>Return Code In Hex</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>04</td>
<td>An unrecoverable JSAM error has occurred (JESREAD, AWRITE, ARELEASE, or APURGE). The JSAM return code is in register 1 on return from the IATXJCT macro.</td>
</tr>
<tr>
<td>08</td>
<td>The JCTID field of the JCT record is not “JCT”.</td>
</tr>
<tr>
<td>0C</td>
<td>An attempt to release (TYPE=REL) or delete (TYPE=DEL) a JCT was made without previous read/write or read-only access having been obtained, or an attempt to obtain read-only or read/write access to a JCT was made when the caller already had read/write access.</td>
</tr>
<tr>
<td>10</td>
<td>An attempt was made to ADD a JCT for which the JQE is already in use. This may be an indication of duplicate job numbers.</td>
</tr>
<tr>
<td>14</td>
<td>Reserved.</td>
</tr>
<tr>
<td>18</td>
<td>Same as IATXJQE.</td>
</tr>
<tr>
<td>1C</td>
<td>Same as IATXJQE.</td>
</tr>
<tr>
<td>20</td>
<td>Same as IATXJQE.</td>
</tr>
</tbody>
</table>

Linkage

This macro uses ASAVE linkage.

Restrictions

Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task.

Note: If you need to access the scheduler elements (SEs) from the job control table (JCT), do not use the values JCTFEND or JCTFSIZE. Instead, add the value JCTFIXL to the base of the JCT.

IATXJDS (Job Data Set Control Block Access Routines)

The IATXJDS macro accesses a job data set (JDS) associated with a specific job. This macro should be used along with other macros that process entries within the JDS: JDSADD, JDSGET, JDSPPOINT, JDSPUT, JDSHOLD, and JDSREL macros.

The IATXJDS macro searches for the requested JDS and does one of the following:
- Creates a new JDS
- Writes an updated JDS to spool
- Reads a specific JDS from spool
- Releases a specific JDS

On normal return, register 1 will contain the address of the file description block specified with the FDB= parameter.
If an error return is specified (with the ERROR= parameter), then in the event of an error, the DM codes and error return codes are returned in register 0 and register 15 from the services used by the IATXJDS service routine:

- For TYPE=CREATE and TYPE=WRITE, the return codes are passed back from the AWRITE macro.
  - If CHAIN=YES is specified with TYPE=WRITE, the return codes are passed back from the WRTCHAIN macro.
- For TYPE=READ, the return codes are passed back from the JESREAD macro.
- For TYPE=RELEASE, the return codes are passed back from the ARELEASE macro.
  - If CHAIN=YES is specified with TYPE=RELEASE, the return codes are passed back from the IATXRELC macro.
  - See these macros for explanations of the return codes.

If you do not specify an error return on IATXJDS, and an error occurs, the IATXJDS service routine issues the FAILDSP macro using the DM code returned from the service routines that it calls.

**Syntax**

```
IATXJDS

TYPE=

- CREATE  CKPT  TATPTR
- READ  DISP  PUTBUF
- WRITE  DISP  PUTBUF  CHAIN
- RELEASE  ROOT  CHAIN

-FDB=(R1)  ,RQ=(reg)  ,NORMAL=(reg)

-ERROR=(reg)

ROOT:

,ROOT=(RQ)

DISP:

,DISP=(number)
```
Parameters

**TYPE=**

Specifies the type of service to be performed. Register 2 is used to pass the function code to the service routine.

**CREATE**

Specifies that a new buffer is to be obtained for a JDS. If CKPT=NO is specified, the buffer will not be written to spool until the next TYPE=WRITE request.

**READ**

Specifies that the specified JDS or JDS chain is to be read into storage.

**WRITE**

Specifies that the specified JDS or JDS chain is to be written to spool.

**RELEASE**

Specifies that the specified JDS record is to be released (that is, the spool address is restored in the FDB to allow subsequent reads, and the buffer will be released).

**FDB=**

Specifies the address of the single record file FDB that points to the JDS for the job. Only register 1 may be used to pass the address of the FDB to the service routine.

**TATPTR=**

Specifies the address of the track allocation table (JBT) FDB used for allocating space to a new JDS record. This parameter is required for TYPE=CREATE. Only register 0 may be used to pass the JBT FDB address to the service routine.
RQ=
  Specifies the address of the resident job queue element (RQ) for the job. The
  FCTWORK field in the caller’s FCT is used to pass the address of the RQ to
  the service routine (if necessary).

ROOT=
  Specifies the origin of the FDB specified on the FDB= parameter. If ROOT is
  specified as an FDB address, this record is updated in the event of a recovered
  I/O error in the JDS chain. If ROOT=RQ, then the job’s resident job queue
  element (RQ) contains the FDB specified on the FDB= parameter. For
  ROOT=RQ, the DISP= parameter cannot be specified.

DISP=
  Specifies the displacement, within the master buffer, of the FDB that points to
  the next record in the chain (the "chain FDB"). (This parameter cannot be
  specified with ROOT=RQ, and is only valid for TYPE=CREATE and
  TYPE=WRITE).

CHAIN=
  Specifies whether the request is for one record or for the entire JDS chain.
  (This parameter is only valid for TYPE=CREATE and TYPE=WRITE.)

    YES
      Specifies that the entire JDS chain will be written to spool (using the
      WRTCHAIN macro). For TYPE=RELEASE, the entire JDS chain will be
      released (using the IATXRELC macro).

    NO
      For TYPE=WRITE, specifies that only the first JDS record is to be written to
      spool (using the AWRITE macro). For TYPE=RELEASE, specifies that only
      the first JDS record is to be released (using the ARELEASE macro).

PUTBUF=
  Specifies whether the buffer containing the JDS is retained after the requested
  I/O has completed. (This parameter is only valid for TYPE=CREATE and
  TYPE=WRITE).

    YES
      Specifies that the buffer containing the JDS record is released after the I/O
      has completed. On return, the FDB contains the spool address of the JDS
      record.

    NO
      Specifies that the buffer containing the JDS record will be retained after the
      I/O has completed. On return, the FDB contains the address of the buffer
      containing the JDS record.

CKPT=
  Specifies whether the buffer containing the JDS is to be updated on spool. (This
  parameter is only valid for TYPE=CREATE.)

    YES
      Specifies that the buffer containing the JDS record is to be written to spool.
      On return, the FDB contains the spool address of the JDS record.

    NO
      Specifies that the buffer containing the JDS record is not to be written to
      spool. On return, the FDB contains the address of the buffer containing the
      JDS record.
**IATXJDS**

**NORMAL=**
Specifies the address of the routine that will receive control if the request is processed successfully.

**ERROR=**
Specifies the address of the routine that will receive control if the request encounters an error.

**Linkage**
This macro uses ASAVE linkage. The IATXJDS service routine is located in module IATGRJA and is pointed to by the TVT.

**Register Conventions to Follow**
Register conventions are as follows:
- Registers 0, 1, 2, 14, and 15 are destroyed by this macro.
- Register 0 can only be specified for the TATPTR= parameter.
- Register 1 can only be specified for the FDB= parameter.
- Registers 2, 14, and 15 cannot be used for the return parameters (ERROR=, and NORMAL=).
- All other parameters may specify registers 3-10.

**Example of Use**
The following is an example of code issuing the IATXJDS macro to create a JDS for a job.

```
*----------------------------------------------------------------*
* CREATE A JOB DATA SET CONTROL BLOCK FOR THE JOB                 *
*----------------------------------------------------------------*
IATXJDS TYPE=CREATE,     BUILD A NEW JDS       *
      FDB=USERJDS,        *
      TATPTR=JCTTATFD,    *
      CKPT=NO
*----------------------------------------------------------------*
* BUILD THE FIRST ENTRY IN THE JDS                               *
*----------------------------------------------------------------*
* CHECKPOINT THE JDS ON SPOOL                                   *
*----------------------------------------------------------------*
IATXJDS FDB=USERJDS,     WRITE THE NEW           *
      TATPTR=JCTTATFD,    JDS TO SPOOL       *
      TYPE=WRITE
*----------------------------------------------------------------*
MVC JCTJDSFD,USERJDS    SET UP THE JDS ANCHOR
```

**IATXJLOK (JES3 BSC Lock Service)**
The IATXJLOK macro obtains or releases ownership of a BSC RJP line lock in order to serialize access to a line device characteristics table (DCT) and other resources associated with the line. There is one doubleword lock area defined in each BSC line DCT for this purpose. When obtaining or releasing a lock for a particular BSC line, the calling program must specify the address of the doubleword lock area associated with that line. In the case of a request to obtain a lock, the lock area will be used to set up serialized access to the line and to identify ownership. In the case of a request to release a lock, the lock area will be cleared to free up ownership of the line resource.
### Syntax

```
label IATXJLOK, OBTAIN, LOCK=(Rx), TYPE=BSCLINE
```

### Parameters

**OBTAIN**
- A positional parameter specifying that the lock is to be obtained on behalf of the caller. This parameter is required unless the RLSE parameter is specified, in which case it should not be specified.

**RLSE**
- A positional parameter specifying that the lock is to be released on behalf of the caller. This parameter is required unless the OBTAIN parameter is specified, in which case it should not be specified.

**LOCK=**
- Specifies that general register n contains the address of the doubleword lock.
- Specifies the address of the doubleword lock field.

**TYPE=**
- Specifies the type of lock being obtained or released.
  - **BSCLINE**
    - Specifies that a BSC line lock is required.

**MODE=**
- Determines the type of return from the lock service routine in the case where another FCT holds the lock at the same time the caller requests it.
  - **UNCOND**
    - If another FCT holds the lock at the time of the request, MODE=UNCOND specifies that the caller waits until the lock is freed and subsequently acquired in its behalf.
  - **COND**
    - If the lock is currently held by another FCT, MODE=COND specifies that the lock service is to return to the caller without waiting for the lock. The caller is responsible for determining using the return code whether the lock was successfully obtained.

**NOTIFY=**
- This is an optional interface for the lock services whereby the caller receives a POST after the release of a lock. This option may only be specified when MODE=COND is also specified.
  - **BSCLM**
    - This option is intended only for the BSC line manager. It specifies that the BSC RJP line manager is to be posted when the requested lock is released.
**IATXJLOK**

### Return Codes

When the OBTAINT parameter is specified:

<table>
<thead>
<tr>
<th>Return Code (Hex)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>OBTAIN request successful; the caller now owns the lock.</td>
</tr>
<tr>
<td>04</td>
<td>Lock busy; the resource is currently controlled by another FCT (only if MODE=COND).</td>
</tr>
<tr>
<td>08</td>
<td>The lock is already held by this FCT (lock was not reacquired).</td>
</tr>
</tbody>
</table>

When the RLSE parameter is specified:

<table>
<thead>
<tr>
<th>Return Code (Hex)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>RELEASE request successful; the lock was freed.</td>
</tr>
<tr>
<td>04</td>
<td>The lock was not held by this FCT. This return code is effectively a NOP, as no action is taken with respect to the lock.</td>
</tr>
</tbody>
</table>

### Linkage

This macro does not use ASAVE linkage. The lock service routine is invoked using a BALR R14,R15 and registers 0 and 1 are used to pass parameters. All other registers remain unchanged.

### Restrictions

Do not use this macro in any routine that might run in a C/I FSS address space.

### USAGE NOTES

- The caller is assumed to have addressability to the TVT using register 12 and to the caller’s FCT using register 11.
- If a DSP currently owns a lock obtained with this service and requests a second lock, the DSP is failed by the lock service routine. The FAILDSP issued in the lock service routine has a dump code of DM054. Before the FAILDSP is issued, the caller’s registers are restored from FCTWKSAV and the caller’s return address is loaded into register 2, so that it will not be destroyed by the FAILDSP call.
- A DSP should not wait while holding a JES3 lock because this could keep other FCTs from using the locked resource, and could result in a deadlock.

### IATXJMR (Read or Write the JMR)

Use the IATXJMR macro to modify accounting information contained in the job management record (JMR). The IATXJMR macro retrieves the JMR for a job or a transaction program data set, and writes or releases the JMR in the JES3 global address space.

### Syntax
Parameters

**TYPE=**
Specifies the function to be performed.

**GET**
Reads the JMR. On return, register 0 will contain the address of a buffer containing a copy of the JMR, and register 1 will contain the address of the JMR.

**PUT**
Writes the JMR in the JES3 global address space. IATXJMR releases the JMR after writing it.

**Notes:**
1. A TYPE=PUT request should be preceded by a TYPE=GET request.
2. The size of the JMR may not be changed between the TYPE=GET and TYPE=PUT requests.

**REL**
Releases the JMR.

**OSEDS=**
Indicates the address of the OSE data set section.

**TATPTR=**
Specifies the address of the track allocation table FDB. This parameter is only valid with TYPE=PUT.

**NORMAL=**
Specifies the return address for the macro processing routine when no error occurs. If NORMAL= is not specified, the routine returns to the instruction immediately following the macro.

**ERROR=**
Specifies the return address for the macro processing if an error occurs. When control is passed to the address, register 15 contains a return code. The possible return codes are as follows:

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>X'04'</td>
<td>The JMR was not found in the DOI.</td>
</tr>
<tr>
<td>X'08'</td>
<td>An error occurred while reading the DOI.</td>
</tr>
<tr>
<td>X'0C'</td>
<td>No DOI exists.</td>
</tr>
</tbody>
</table>

For TYPE=PUT:
X'04' Reason Code X'10'

Error occurred during IATXSIO
Return Code | Meaning
--- | ---
For TYPE=REL:
X'04' | JES3 could not return the JSAM buffer.

If ERROR= is not specified with TYPE=PUT, the dynamic support program will be ended by using the FAILDSP macro. If ERROR= is specified and an error occurs, control will be returned with a dump code X'107' in register 0.

**Linkage**

This macro uses ASAVE linkage.

**Restrictions**

Do not use this macro in any routine that might run under the JES3 auxiliary task.

This macro may only be used in output service modules.

---

**IATXJOB (Convert Job Numbers and Job Identifiers)**

The IATXJOB macro converts a job number to a job id, a job id to a job number, or a job number in one format (such as binary) to the other format (EBCDIC).

You may need to change the format of a job number in certain situations, depending on how much you’ve customized your system. Some of these situations are:

- In messages, jobs should always be specified with the job identifier rather than a job number.
- In some control blocks, JES3 uses different lengths for the job number field and a different format (sometimes EBCDIC and sometimes binary) for the job number. If you write exit routines that refer to control blocks containing job numbers, you will need to convert the job numbers to a consistent form before displaying the job numbers to the system operator or the end user. For example, you would not want to display a job number in binary form to the operator.
- If you have a DSP that parses your own installation’s commands, the DSP may need to convert job numbers to job identifiers.

JES3 uses the term “mixed format field” to describe the job number field in some JES3 control blocks that reside on spool. The field is called mixed format not because it is part binary and part EBCDIC, but because the value in the field can be either an EBCDIC value or a binary value, depending on the size of the job number. For job numbers less than or equal to 9,999, JES3 uses EBCDIC values for the job number. JES3 uses binary values for job numbers 10,000 and higher.

Mixed-format fields remain mixed-format all the time. Other fields contain only an EBCDIC job number or a binary job number.

When JES3 converts a job number to a job identifier, JES3 adds the letters “JOB” in front of the number, inserting zeros where needed. For example, JES3 converts the job number 4444 to job identifier JOB04444 before referring to the job in a message. Any code you write for an installation exit or DSP should use this same convention.
Syntax

Parameters

**JOBID=**
Specifies that the input is the address of an 8-character EBCDIC job id. The address of the job id must be provided by the caller either in the specified register, or as an address label. Only registers 1-9 may be used.

**JOBNUM=**
Specifies that the input is a 2 byte or 4 byte job number in the specified register or at the specified address label. The 4 byte job number is EBCDIC if less than or equal to 9999 and binary if 10000 or higher. Only registers 1-9 may be used when specified as a register.

**OUTJBID=**
Specifies that the output should be an 8-character job id. The address at which the job id will be returned must be provided by the caller in the specified register or as an address label. (See note below). Only registers 2-9 may be used.

**OUTEBCD=**
Specifies that the output should be a 7-byte EBCDIC job number, right-justified
with character zeros. The address at which the job number will be returned must be provided by the caller in the specified register or as an address label. (See note below). Only registers 2-9 may be used.

The WRKAREA parameter should not be specified with this parameter if JOBID= is also specified.

**OUTNUM=BINARY**

Specifies that the output will be a 4-byte binary job number. It will be returned in register 0. (See the following note).

**PREFIX=**

Specifies the three characters to precede the job number when the job id is constructed. The default is JOB. If the job number is greater than 99999, only the first character of the prefix is used. For example, the constructed job id for job 31415 is JOB31415, and the constructed job id for job 926536 is J0926536.

**Note:** JOB is the prefix under normal circumstances. Therefore, this parameter is typically allowed to default. Other prefixes are used by JES3 internally.

**WRKAREA=**

Specifies the address of an 8-byte work area provided by the caller. The address may be provided in the specified register or as an address label. Only registers 2-9 may be used.

WRKAREA is a required parameter, except in one instance: When IATXJOB is being used to generate a 7-byte EBCDIC job number from an 8-byte EBCDIC job ID (item number 2 below), then WRKAREA may be omitted.

**Note:** If OUTJBID=, OUTNUM=, or OUTEBCD= is not specified, the output will be a 4-byte job number in either binary or EBCDIC form (EBCDIC if 9999 or less, binary if greater than 9999) in register 0.

The table presented in Figure 10 on page 339 shows a summary of the seven combinations of IATXJOB. The numbers in the boxes correspond to one of the seven combinations that appear after the table.
The seven valid combinations of this macro are:

1. To convert a job id to a 4-byte binary job number (for example, to use in macro AJOBNUM which only accepts binary job numbers):

\[
\text{IATXJOB JOBID}=(R1), \text{OUTNUM} = \text{BINARY}, \text{WRKAREA}=(R2)
\]

2. To convert a job id to a 7-byte EBCDIC job number (for example, to use in the INTERCOM macro):

\[
\text{IATXJOB JOBID}=(R3), \text{OUTEBCD}=(R5)
\]

3. To convert a job id (using default values) to a mixed-format job number (that is, either a binary job number or an EBCDIC job number):

\[
\text{IATXJOB JOBID} \Rightarrow \text{RQJNUM}, \text{WRKAREA}=(R2)
\]

4. To convert a 4-byte binary or EBCDIC job number to an 8-byte job id (for example, to insert a job id into a message):

\[
\text{IATXJOB JOBNUM}=(R1), \text{OUTJBID}=(R2), \text{WRKAREA} = \text{FCTWORK}
\]

5. To convert a 4-byte binary or EBCDIC job number to a 4-byte binary job number:

\[
\text{IATXJOB JOBNUM}=(JDABJNUM), \text{OUTNUM} = \text{BINARY}, \text{WRKAREA}=(R5)
\]

6. To convert a 4-byte binary or EBCDIC job number to a 7-byte EBCDIC job number:

\[
\text{IATXJOB JOBNUM}=(R1), \text{OUTEBCD}=(R3), \text{WRKAREA}=(R9)
\]

7. To convert a 4-byte binary or EBCDIC job number (using default values) to a mixed-format job number (that is, to either a 4-byte EBCDIC or binary job number):

\[
\text{IATXJOB JOBNUM}=(R1), \text{WRKAREA}=(R9)
\]

**Linkage**

This macro is an inline expansion macro.

**Restrictions**

Registers 0 and 1 are destroyed in most cases.
IATXJQE (Obtain the Address of a JQE)

The IATXJQE macro obtains the address of the job queue element (JQE) when the job name or job number is known. The individual priority levels may be accessed if the priority level is known.

Syntax

IATXJQE

Syntax:

```
label IATXJQE JOBNUM=(R1) (reg)
      JOBNAME=(R1) L=(R2)
      JOBOWN=(R1) (reg)
      PRTY=(R1) (reg) Priority Syntax
      DEQ=ONE ALL
      FUNC=SEARCH Execute Syntax
      Modify Syntax
      List Syntax
      FUNC=SEARCH_END
      TOKEN=address
```

```
ERROR=address
NORMAL=address
```

Priority Syntax:

```
BUSY=address
ACCESS=EXCL SHR
```

Execute Syntax:

```
MF=(E, address) TOKEN=address
SEARCH_TYPE=AND OR
```

```
EOD=address
ACCESS=SHR EXCL
```

Modify Syntax:

```
MF=(M, address) KEYS= (field, value, cond)
```
Parameters

The valid functions for the IATXJQE macro are as follows:

- **JOBNUM** — Access JQE’s by job number.
- **JOBNAME** — Access JQE’s by job name.
- **JOBOWN** — Access JQE’s by job owner.
- **PRTY** — Access JQE’s by priority.
- **DEQ** — Dequeue a JQE priority.
- **FUNC=SEARCH** — Search for JQE’s matching the specified selection criteria.
- **FUNC=SEARCH_END** — Complete a JQE search.

The parameters will be listed under the individual functions.

**JOBNUM=**

Specifies the binary job number. A register, a 4 byte field, or a 2 byte field may be specified. If a 2 byte field is specified, it must be defined in the assembly before the macro invocation. On return, register 0 contains the 31-bit address of the job name. The length of the job name is eight bytes. Also on return, register 1 contains the JQE address (IATYJQE). Either a fullword or halfword field may be specified.

If this parameter is specified, the BUSY and DEQ parameters may not be specified.

**JOBNAME=**

Specifies the register containing the address of the job name. The job name table is searched until the first entry matching the specified job name is found or all entries have been examined. If the job name contains an * in character position two or higher, the * is treated as a delimiter. The comparand length and value for the job name search may thus be a subset of the full job name. If the job name address remains from a previous IATXJQE request, the job name search is continued, beginning with the next entry in the table.

This procedure allows generic searches to locate all jobs with the same name or subset name. Iterative searches use the original comparand value and length. The register 0 value returned by IATXJQE is required as input for generic job name searches.

If you specify JOBNAME, both the input to this macro and the output from this macro use register 2 to contain the job name length.

If this parameter is specified, the BUSY and DEQ parameters may not be specified.

**L=**

Specifies the length of the job name provided as input. The length value should be in the low-order byte of the register, with the rest of the register containing zeros. For the register, you can specify any register from 2 through 9 or register 13.

For output, the length is returned in the low-order byte of register 2 with the rest of the register containing zeros.

The L= parameter is valid only when JOBNAME= is specified.
JOBOWN=
  Specifies the register containing the job owner. On return, register 0 contains the 31-bit address of the job name. The length of the job name is eight bytes. Also on return, register 1 contains the JQE address (IATYJQE).

If this parameter is specified, the BUSY and DEQ parameters may not be specified.

PRTY=
  Specifies the binary priority level to be searched or the binary job number if the priority level is to be scanned beginning with a specific job (R1 or reg). If the DEQ parameter is specified, this parameter specifies the priority level to be dequeued.

On return, register 0 contains the address of the job name and register 1 contains the JQE address (IATYJQE).

The initial call will provide the first job in the specified priority level. Subsequent calls will provide the successor jobs. The priority level is enqueued for the life of the request. Dequeue occurs automatically when end-of-data (EOD) is encountered. If the PRTY level scan is to be ended before EOD, the IATXJQE DEQ option may be used to force an EOD and dequeue.

ACCESS=
  Specifies whether the JQE priority should be enqueued exclusively or shared. If ACCESS=SHR is specified, multiple FCT’s can access the JQE’s in the priority at the same time. If ACCESS=EXCL is specified, only one FCT at a time can access the JQE’s in the priority. The default is EXCL.

BUSY=
  Specifies the address of a routine to be given control if the PRTY level specified is already in use. If this parameter is not specified, the routine will wait until the PRTY level specified becomes available. If the JOBNUM or JOBNAME parameter is specified, this parameter may not be specified.

On a BUSY return, this macro passes back an ECF mask and address. The 31-bit ECF address is returned in register 0. The ECF mask needed is contained in the low-order byte of register 1, with the other three bytes containing zeros.

DEQ=
  Requests that ONE or ALL priority levels are to be dequeued. This operand may be specified at any time to release control of priority level(s) that are enqueued during the IATXJQE priority scan(s). The active FCT is used in the ENQ/DEQ operations. It is not necessary to dequeue if the priority scan is ended using EOD as this causes automatic release of the priority level. Specification of DEQ forces an EOD. The PRTY value is used when ONE is specified. If ALL is specified, the specified PRTY value is ignored. If the JOBNUM or JOBNAME parameter is specified, this parameter may not be specified.

FUNC=SEARCH
  Specifies a search for JQE’s matching the specified selection criteria.

MF=
  (E,address)
    Specifies an execute request.

TOKEN=
  An eight byte input/output value which contains a token returned by
the IATXJQE service. This token value serves as a reference point to indicate where you left off in the search.

**SEARCH_TYPE=**
Specifies whether a match should occur for multiple keys when one key matches (OR) or when all keys match (AND). The default is AND.

**ACCESS=**
Specifies whether the JQE priority should be enqueued exclusively or shared. If ACCESS=SHR is specified, multiple FCT’s can access the JQE’s in the priority at the same time. If ACCESS=EXCL is specified, only one FCT at a time can access the JQE’s in the priority. The default is SHR.

**EOD=**
Specifies the address to which the routine returns if end-of-data has been reached.

(M,address)
Specifies a modify request. The modify request sets the arguments used to search the JQE’s.

**KEYS=**

- **field**
  Can be one of the following:
  - JQExxxxx — The name of a field in the JQE to be examined.
  - RQxxxxxx — The name of a field in the RQ pointed to by the JQE to be examined.
  
  **Note:** If a particular job’s JQE does not have a RQ, the JQE will not be returned, even if NE is specified.
  
  - JOBNAME — The format for values associated with this keyword are a specific job name, a generic job name, a job name prefix, or a job name containing wildcard characters.
  
  **Note:** A question mark, “?”, represents a single character and an asterisk, “*”, represents zero or more characters.
  
  - OWNER — The format for values associated with this keyword are a specific job owner, a generic job owner, a job owner prefix, or a job owner containing wildcard characters.
  
  **Note:** A question mark, “?”, represents a single character and an asterisk, “*”, represents zero or more characters.

- **value**
  The value in the specified field. Can be one of the following:
  - A decimal value, such as 0, 1, or 32764.
  - A hexadecimal value, such as X’10’ or X’FFFF’.
  - A character string, such as ‘ABCD’ or ‘1234’.
  - An equate that represents a flag bit name or constant, such as JQEDSEL or RQONMAIN.
• A field name that is defined locally or in a control block that is addressable when the IATXJQE SEARCH request is issued. Valid field names must be preceded by ”FIELD-”. For example
  FIELD-FCTRQAD
  FIELD-MCNAME

  are valid values.

cond
The condition that must occur in order for the key to match the value, such as EQ, NE, LE, GT, ALLON, or ALLOFF.

Note: See macro IATYEQU for the explanations for these conditional arguments.

L  Specifies a list request

MAXKEYS
Specifies the maximum number of search keys to be used.
MAXKEYS defines how large the list form of the parameter list must be.

FUNC=SEARCH_END
Specifies that the caller is finished scanning the JQEs so that the last priority that was enqueued will be dequeued.

TOKEN
The eight byte token that was returned from the last IATXJQE SEARCH request.

ERROR=
Specifies the address of a routine to be given control when errors or end-of-data are detected. On entry, register 15 contains the error return code; register 0 contains the address within module IATGRJX where the error was detected. A common error routine may be used for IATXJCT and IATXJQE macro calls.

NORMAL=
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

Return Codes

<table>
<thead>
<tr>
<th>Return Code In Hex</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td>This return code may indicate that:</td>
</tr>
<tr>
<td></td>
<td>• The job name specified is not in the system.</td>
</tr>
<tr>
<td></td>
<td>• The job number specified is not in the system.</td>
</tr>
<tr>
<td></td>
<td>• There are no more jobs in this priority level.</td>
</tr>
<tr>
<td></td>
<td>• This JQE/JCT is no longer accessible because of unrecoverable error while processing the current or previous IATXJQE or IATXJCT request for this job (JOEFLG1-JQECAT).</td>
</tr>
<tr>
<td></td>
<td>• End-of-data has been reached if the DEQ parameter was specified. (This is not an error.)</td>
</tr>
</tbody>
</table>
### IATXJQE

<table>
<thead>
<tr>
<th>Return Code in Hex</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1C</td>
<td>Incorrect input parameters. This return code may indicate that:</td>
</tr>
<tr>
<td></td>
<td>• The job number specified is not within the valid job number range as specified by the JOBNO parameter of the OPTIONS statement.</td>
</tr>
<tr>
<td></td>
<td>• The length supplied in the high-order byte of register 1 for a generic job name search is greater than 8</td>
</tr>
<tr>
<td></td>
<td>• The priority level specified is greater than 15.</td>
</tr>
<tr>
<td></td>
<td>• Parameter register 0 contents are incorrect.</td>
</tr>
<tr>
<td></td>
<td>• The specified JQE address is incorrect.</td>
</tr>
<tr>
<td>20</td>
<td>Reserved.</td>
</tr>
</tbody>
</table>

### Linkage

This macro uses ASAVE linkage.

### Restrictions

Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task. This macro uses register 2 if the JOBNAME= parameter is specified.

### IATXLOC (Locate a Slot in the DJST)

The IATXLOC macro locates the specified slot in the dynamic job summary table (DJST) and returns to the caller with a pointer in register 1 to the slot.

### Syntax

```
IATXLOC(DJST=label,STEP=address,NAVAIL=address,DD=address,ERROR=address)
```

### Parameters

**DJST=**

Specifies the address of the DJST.

**STEP=**

Specifies the address of the step number. If the STEP parameter is not specified, the search is made for the first active slot.

**NAVAIL=**

Specifies the return address if the specified slot is not found.

**DD=**

Specifies the address of the DD name. If this parameter is not specified, then the search for the slot is based entirely on the address specified in the STEP parameter.

**ERROR=**

Specifies the return address if the DJST is incorrect.
IATXLOC

Linkage

This macro issues the IATZCALL macro specifying SAVE=NO, which generates a BALR R14,R15 for linkage.

Restrictions

Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task.

IATXLP (BDT Language Processor Interface Macro)

The IATXLP macro invokes the services of the BDT language processor to process BDT translation text and constructs the appropriate control block that internally defines a command or transaction to the BDT subsystem.

The IATXLP macro requires that the calling routine be in 24-bit mode. This macro generates a BALR to MVS/BDT code that does not support 31-bit AMODE callers. The calling module must reside below the 16 megabytes, and all data areas passed as parameters must reside below 16 megabytes too.

Syntax

```
<table>
<thead>
<tr>
<th>label</th>
<th>IATXLP-BSID-=(R1)</th>
<th>ERRMSG-=(R15)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>,EP</td>
<td>(R14)</td>
<td>ERROR-=(reg)</td>
</tr>
<tr>
<td></td>
<td>address</td>
<td></td>
</tr>
<tr>
<td>,LIST</td>
<td>address</td>
<td>TIME-=(reg)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TEXT=:

<table>
<thead>
<tr>
<th>TEXT-=(R0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(reg)</td>
</tr>
<tr>
<td>address</td>
</tr>
</tbody>
</table>
```

Parameters

**label**

Optional label associated with the first instruction generated by the IATXLP macro.

**BSID=**

Specifies the address of a BDT subsystem interface area that is used in processing the request. If register notation is specified, the register must contain the address of the BDT subsystem interface data area. If (R1) is specified, register loading is suppressed.

**TEXT=**

Specifies the address of the transaction text record that contains the English-language BDT transaction text being analyzed. The transaction text record consists of a two-byte length field followed by the transaction text being
processed. If register notation is used, the specified register must contain the address of the transaction text record. If (R0) is specified, register loading is suppressed.

**Note:** If the LIST parameter is specified, this parameter must not be specified.

**ERRMSG=**
Specifies the address of a 130-byte area used by the language processor to return an error message, should an error be encountered. If register notation is used, the specified register must contain the address of the error message buffer area. If (R15) is specified, register loading is suppressed.

**EP=**
Specifies the entry point address of the BDT language processor load module. If (R14) is specified, register loading is suppressed.

**ERROR=**
Specifies the address of an exit taken if an error is encountered by the language processor. If register notation is used, the specified register must contain the appropriate exit address.

**NORMAL=**
If specified, contains the address of an exit taken when the language processing function is complete. If register notation is used, the specified register must contain the appropriate exit address.

**LIST=**
Specifies the address of a parameter list.

**Note:** If the TEXT parameter is specified, this parameter must not be specified.

### IATXLPJ3 (JES3 Local Post)

The IATXLPJ3 macro posts either the JES3 master event control block (ECB), which is used by the JES3 primary task, the JES3 auxiliary task ECB, or both. The posting is performed synchronously using either compare and swap logic or branch entry to the MVS POST service. The calling program must be running in supervisor state with key 0 or the JES3 key.

#### Syntax

```
<table>
<thead>
<tr>
<th>Label</th>
<th>IATXLPJ3</th>
<th>TYPE=</th>
<th>JPOST</th>
<th>APOST</th>
<th>LOCKED=</th>
<th>NO</th>
<th>YES</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADDR=</td>
<td>TVT</td>
<td>SSVT</td>
<td>MPC</td>
<td>NONE</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

#### Parameters

**TYPE=**
Specifies the type of post to be done.

- **JPOST**
  Specifies that the JES3 master ECB is to be posted.
APOST
Specifies that the auxiliary task ECB is to be posted.

JAPOST
Specifies that both the JES3 master ECB and the auxiliary task ECB are to be posted.

LOCKED=
Specifies whether the calling module already holds the local lock. If LOCKED=YES is specified, the caller must be in key 0, supervisor state. If LOCKED=NO is specified, the caller may be in either key 0 or in the JES3 key.

ADDR=
Specifies the way the IATXLPJ3 post routine address, which is located in the TVT, is to be obtained. The subparameter that you use depends upon the control block to which you have addressability. The macro produces the necessary expansion to obtain the address.

TVT
Specifies that the address is to be obtained directly from the TVT. The TVT DSECT (IATYTVT) must be defined by a USING.

SSVT
Specifies that the address is to be obtained using the subsystem vector table (SSVT). The SSVT DSECT (IATYSVT) and the TVT DSECT (IATYTVT) must be included in the module and the SSVT must be defined by a USING instruction.

MPC
Specifies that the address is to be obtained using the main processor control table (MPC). The MPC DSECT (IATYMPC) and the SSVT DSECT (IATYSVT) and the TVT DSECT (IATYTVT) must be included in the module and the MPC must be defined by a USING instruction.

NONE
Specifies that the address must be obtained using the CVT. If ADDR=NONE, the following must be included in the module:
- The CVT DSECT with DSECT= YES specified on the CVT macro call.
- The JSECT DSECT (IEFJESECT)
- The SSCT DSECT (IEFJSCVT)
- The SSVT DSECT (IATYSVT)
- The TVT DSECT (IATYTVT)

Note: In order to obtain the entry point of the MVS POST routine, you must specify one of the ADDR= subparameters, or take the TVT default. Which of these subparameters is specified depends on what control block your routine has access to at the time that a post is needed.

Linkage
This macro does not use ASAVE linkage. The post service routine is invoked using a BALR R14,R15 with register 0 pointing to the input parameter list. If the calling program does not hold the local lock, registers 0-3 and 13-15 will be unpredictable upon return. registers 0-3 and 13-15 will be unpredictable upon return.

Restrictions
If this macro is issued by any routine that might run in a C/I FSS address space, the only valid parameters are TYPE=JPOST and ADDR=TVT.
IATXLRGT (SNA Logical Record Get)

The IATXLRGT macro receives statement records or console messages transmitted from a SNA work station logical unit.

Syntax

```
IATXLRGT SUPUNIT = (reg), AREA = (reg)
```

```
, EOD = (reg), AREALEN = n, ERROR = (reg)
```

```
, NORMAL = (reg)
```

Parameters

**SUPUNIT=**
Specifies the address of the SUPUNITS table entry for the device at the work station from which data is to be received.

**AREA=**
Specifies the address where the data is to be placed. For console data, register 0 is set to the byte count of data. For card reader data, register 0 is set to the number of statement logical records returned. The default is 1. This value cannot be greater than 255.

**EOD=**
The address to which control is returned when end-of-data (EOD) is reached.

**AREALEN=**
Specifies the length of the card reader or console message buffer at the address specified by the AREA parameter. If register notation is used, the value must be right-justified in the register.

**ERROR=**
This parameter specifies the address to which the macro processing routine returns if an error occurs. When control is passed to the address, register 0 contains a dump code and register 15 contains a return code, unless otherwise specified. The return codes are described below.

If an error occurs and ERROR was not specified, the dynamic support program (DSP) will be ended using the FAILDSP macro.

**NORMAL=**
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

Return Codes

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>04</td>
<td>SCB/SCS data error</td>
</tr>
<tr>
<td>08</td>
<td>Temporary device error</td>
</tr>
</tbody>
</table>
IATXLRPT (SNA Logical Record Put)

The IATXLRPT macro sends print and punch logical records or console messages to a SNA work station logical unit.

Syntax

```
label IATXLRPT SUPUNIT=(reg) address ,FLAG=(reg) address ,LIST=(reg) address ,ERROR=(reg) address ,NORMAL=(reg) address
```

Parameters

**SUPUNIT=**
Specifies the address of the SUPUNITS table entry for the device at the work station to which the data is sent.

**FLAG=**
Specifies the address of a 1-byte flag which is set in the following manner:

<table>
<thead>
<tr>
<th>FLAG= Settings</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0... ...</td>
<td>The carriage control character is in machine code.</td>
</tr>
<tr>
<td>1... ...</td>
<td>The carriage control character is in ASA code.</td>
</tr>
<tr>
<td>.1... ...</td>
<td>The data input is a forms control buffer (FCB) image to be converted into an SVF sequence.</td>
</tr>
<tr>
<td>..1. ...</td>
<td>End of chain. This indicator is valid with or without input data.</td>
</tr>
<tr>
<td>...1 ...</td>
<td>Send print records with transparency.</td>
</tr>
<tr>
<td>.... 1...</td>
<td>Forms feed (eject) console device.</td>
</tr>
</tbody>
</table>

Linkage

This macro uses ASAVE linkage.

Restrictions

Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task.
LIST= Specifies the length and address for the data being PUT. For console, you can specify one length and one address. For card or print media, you can specify two length-address pairs. All fields are fullwords and are specified in the order LEN1, ADDR1, LEN2, ADDR2.

CCC= Specifies the address of a 1-byte field that contains the carriage control character for the print logical record. If this parameter is omitted or its value is 0, the carriage control character is assumed to be the first byte of the print record.

ERROR= This parameter specifies the address to which the macro processing routine returns if an error occurs. When control is passed to the address, register 0 contains a dump code and register 15 contains a return code, unless otherwise specified. The return codes are described below.

If an error occurs and ERROR was not specified, the dynamic support program (DSP) will be ended using the FAILDSP macro.

NORMAL= This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

Return Codes
Following are the error return codes which are returned in register 15:

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>04</td>
<td>Data conversion error occurred</td>
</tr>
<tr>
<td>0C</td>
<td>Temporary device error occurred</td>
</tr>
<tr>
<td>10</td>
<td>Permanent device error occurred</td>
</tr>
<tr>
<td>14</td>
<td>Session error occurred</td>
</tr>
<tr>
<td>18</td>
<td>Terminate</td>
</tr>
<tr>
<td>1C</td>
<td>Incorrect FCB</td>
</tr>
</tbody>
</table>

Note: When register notation is used, the register must be 4, 5, 6, 7, 8, or 9. Registers 0, 1, 2, 3, 14, and 15 must not be used to pass parameters because their contents are destroyed.

Linkage
This macro issues a BALR R14,R15 for linkage.

Restrictions
The issuer of this macro must run under the control of an FCT. Do not use this macro in any routine that might run in a C/I FSS address space.

IATXMBFE (Map the SEND Macro Options)
The IATXMBFE macro maps the options used on the VTAM SEND macro to the RU buffer entry (BFE) control block. This information is used by data flow control (DFC) in managing session protocols.
IATXMBFE

Syntax

```
IATXMBFE BFE
  ^---label---(R1)
  ^---(reg)---address
  ^---BB---EB---FOC---MOC---EOC---OC---FMH---CAN---CD---ALT
```

Parameters

BFE= Specifies the address of the BFE to be mapped.

BB Indicates that begin bracket is set.

EB Indicates that end bracket is set.

FOC Indicates that first of chain is set.

MOC Indicates that middle of chain is set.

EOC Indicates that end of chain is set.

OC Indicates that only in chain is set.

FMH Indicates that set function management header is present.

CAN Indicates that cancel RU is sent.

CD Indicates that change direction is set.

ALT Indicates that alternate code is set.

Note: When this macro executes, the contents of register 1 are destroyed.

Linkage

This macro expands as inline code.

Restrictions

Do not use this macro in any routine that might run in a C/I FSS address space.

IATXMGET (Obtain Storage Address of a Specified Area)

The IATXMGET macro obtains the address of an area in the JES3 address space, the JES3 auxiliary address space (JES3AUX), or a data space — as specified by the DSPNAME parameter — during JES3 abend formatting. The routine that services the request is in module IATABN0 for JES3 abend formatting and in module IATABPR for IPCS (interactive problem control system) dump formatting.
On return, register 1 contains the address of the area.

**Syntax**

```
.label IATXMGET AREA= Parameters

,ASID= (R0) address

,LENGTH= n (reg)

,VERIFY= NO YES

,ERROR= (reg) address

,DSPNAME= name

,NORMAL= (reg) address

```

**AREA= Parameters:**

```
AREA= (R1) (reg) address

,BASE= (reg) ,REF= label

,WREG= (reg)
```

**Parameters**

**AREA=**

Specifies the address of the area in the dump to be obtained. If you specify AREA=address, then BASE= and REF= must be specified if this is the first expansion of IATXMGET in this module.

**BASE=**

Specifies the base register to be used in performing the calculation of the address specified for AREA.

**REF=**

Specifies the label in the control block which the BASE register points to. If on an invocation of IATXMGET, BASE or REF is not specified, then it defaults to its last specified value.

**WREG=**

Specifies a work register to be used during the address calculation. If not specified, it defaults to R1. The address calculation is done as follows:

```
LAWREG,AREA-REF
ALRWREG,BASE
```

where the values of WREG, AREA, REF, and BASE are symbolically substituted in the above instructions.

The values specified for BASE and WREG must be different.

Registers 0, 1, 14, and 15 are changed by the macro and should not be specified for BASE.

**ASID=**

Specifies the ASID of the area. If R0 is used, the address space identifier
IATXMGET

(ASID) must be in the two low-order bytes. If this parameter is omitted or a value of zeros is coded, the current JES3 address space identifier will be used.

LENGTH=
  Specifies the length of the area, in bytes. The maximum value that can be coded is 255. A register can be specified.

  Note: R0 and R1 cannot be specified.

VERIFY=
  Specifies whether the area address is to be checked for a fullword boundary. When YES is coded, the ERROR parameter must also be coded.

DSPNAME=
  Specifies the address of a data space name where the area is located.

ERROR=
  Specifies the address of a routine to which the macro processing routine is to return when: (1) IPCS cannot retrieve the requested address from a dump; (2) the specified area was not obtained or the area does not have the JES3 storage protection key during JES3 abend formatting; or (3) the VERIFY=YES parameter is specified and the area address is not on a fullword boundary.

NORMAL=
  This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

Linkage

  This macro issues a BALR R14,R15 for linkage.

Restrictions

  This macro may be used only by the abend formatting routines.

IATXMID (Generate Message Identifier)

The IATXMID macro generates a message identifier that can be used to prefix any messages which are to be written to the operator. Each message identifier consists of 7 characters, the first three being IAT and the last four being the message number.

Syntax

```
IATXMID-xxxx
```

Parameters

xxxx
  Specifies 4 decimal digits to be used as the message identifier.

Linkage

  This macro expands as inline code.
Restrictions

Do not use this macro in any routine that might run under the JES3 auxiliary task.

IATXMLWO (Multi-line WTO Message)

This macro creates a token which represents one line of a multi-line message. It has two functions for multi-line WTO message processing:

- Building a single line of a multi-line WTO message
  This macro creates a IATYMLWO token which represents one line of a multi-line WTO message. These tokens will be chained together and sent to the MESSAGE macro to issue the multi-line WTO message. On exit from this macro, register 1 (R1) will contain the address of the first IATYMLWO token for the multi-line WTO message. Each IATYMLWO token will be chained to the next IATYMLWO token in the chain for the multi-line WTO message. This is done by passing in the IATYMLWO token address of the first line of the multi-line WTO message through the TOKEN parameter. The TOKEN parameter must be zero for the first line of the multi-line WTO message. The address of the first IATYMLWO in the chain must be passed to the MESSAGE macro through the MLWOLST parameter to issue the multi-line WTO message. The supported limit on number of lines in a multi-line WTO message is 999. The following example illustrates how a multi-line WTO message can be created:

- Cleaning up a multi-line WTO message.
  This macro frees the storage of all the IATYMLWO tokens in a chain for a multi-line WTO message. The IATYMLWO token address of the first line of the multi-line WTO message is passed in through the TOKEN parameter.

Syntax

```
<table>
<thead>
<tr>
<th>Label</th>
<th>IATXMLWO</th>
<th>REQUEST=</th>
<th>BUILD Request</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>CLEANUP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TOKEN=</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(reg)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>address</td>
<td></td>
</tr>
</tbody>
</table>

BUILD Request:

```

Multi-Line WTO Message

```
IATXMLWO ...,TOKEN=0
ST R1,MSG
IATXMLWO ...,TOKEN=MSG
IATXMLWO ...,TOKEN=MSG
IATXMLWO ...,TOKEN=MSG
MESSAGE MLWOLST=MSG,...
```
Parameters

REQUEST= Specifies the type of request to process.

BUILD Indicates a request to build a new message line token.

CLEANUP Indicates a request to clean up a multi-line WTO message chain of tokens.

TEXT= Specifies the name or address in a register of a required character input that contains the single line of a multi-line WTO message for which a IATYMLWO token is to be created. The first byte of this message line must contain the length of the line (excluding this length byte), and the actual text of the line must begin in the second byte. The maximum allowed length of the line is dependent on the type of line specified by the LINETYPE keyword as follows: maximum of 34 bytes for a control line; maximum of 70 bytes for a label line or data line. If REQUEST=BUILD is specified, this keyword is required. If REQUEST=CLEANUP is specified, this keyword is invalid.

LINETYPE= Indicates the type of line within a multi-line WTO message for which a IATYMLWO token is to be created. If REQUEST=BUILD is specified, this keyword is required. If REQUEST=CLEANUP is specified, this keyword is invalid.

C Indicates this line is a control line of a multi-line WTO message.

L Indicates this line is a label line of a multi-line WTO message.

D Indicates this line is a data line of a multi-line WTO message.

TOKEN= Specifies the name or address in register (2)-(12) of a required 4 byte input that contains the address of the IATYMLWO token of the first line in the multi-line WTO message.

When REQUEST=BUILD is specified, the new message line being built will be chained to the end of the message that this parameter represents. This parameter must be zero for the first line of a multi-line WTO message.

When REQUEST=CLEANUP is specified, all lines of the message that this parameter represents will be cleaned up (i.e., all storage for the message will be freed).

MF= Specifies the macro form.

S Specifies the standard form of the macro. The standard form generates code to put the parameters in a parameter list and invoke the specified operation. It is the default.

L Specifies the list form of the macro. The list form defines an area to be used for the parameter list.

xmfcrit Specifies the name of a storage area for the parameter list. It is required input.

xmfatatr Specifies an optional character string that is used to force boundary
alignment of the parameter list. The default is 0D which forces the parameter list to a doubleword boundary.

E  Specifies the execute form of the macro. The execute form generates code to put the parameters into the parameter list specified by xmfctrl.

xmfctrl  Specifies the name or address in register (1)-(12), of a storage area for the parameter list. It is required input.

COMPLETE  Specifies the macro parameter syntax checking. A check is done for required macro keys.

Linkage

This macro uses ASAVE linkage.

Restrictions

Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task.

IATXMLWO (Determine Minimum and Maximum Values, and Accumulate Total Values)

The IATXMLWO macro tabulates the value of a field by maintaining a table showing the minimum and maximum values of the field, and also showing the accumulated value of all numbers each time a new value is examined.

The macro service compares an incoming number (or value) to the old minimum and old maximum. If the new value is lower than the minimum or higher than the maximum, the new value becomes the new minimum or new maximum, respectively. Also, the new value is added to the running total to give a new running total. The example on the next page shows how the macro works.

This macro can be useful if you write an installation exit that monitors performance indicators.

Syntax

```
IATXMLWO

NEWVALUE= (R1) (Rx)

OLDMIN= label

OLDMAX= label

OLDTOTAL= label

HALFWORD= NO YES
```

Parameters

**NEWVALUE=**

Specifies the address of the current value for which the minimum, maximum, and running total are being maintained.

**OLDMIN=**

Specifies the address of the previous minimum value.
IATXMNMX

OLDMAX=
   Specifies the address of the previous maximum value.

OLDTOTAL=
   Specifies the address of the previous cumulative total.

HALFWORD=
   Specifies the type of instruction to generate.

   YES
      Specifies that the fields defined for OLDMIN= and OLDMAX= are halfword fields. This parameter causes the Compare-Halfword (CH Rx,label) instruction instead of the Compare-Fullword (C Rx,label) to be generated. It also causes the Store-Halfword (STH Rx,label) instruction instead of the Store-Fullword (ST Rx,label) instruction to be generated.

   NO
      Specifies that the fields defined for OLDMIN= and OLDMAX= are fullword fields. This parameter causes the Compare (C Rx,label) and Store (ST Rx,label) instructions to be generated.

Linkage
   This macro expands inline. This macro is used by modules of the JMF function.

Restrictions
   None.

Example of Use
   An example of the IATXMNMX macro is:
   IATXMNMX NEWVALUE=(R7), X
   OLDMIN=PTXMIN, X
   OLDMAX=PTXMAX, X
   OLDTOTAL=PTXAVG, X
   HALFWORD=YES

   In the macro in this example, register 7 points to a location containing a number. The macro service routine compares this value to the old minimum and old maximum. If the new value is lower than the minimum or higher than the maximum, the new value becomes the new minimum or new maximum, respectively. Also, the new value is added to the value located at PTXAVG to give a new running total. The fields pointed by OLDMIN= and OLDMAX= are halfword fields.

IATXMP (Main Processor Control Table Services)

   This macro is used to perform different kinds of functions related to the Main Processor Control Tables (MPC’s). All functions expand code in-line.

   If the caller is running in the user’s address space, the address of the first MPC is obtained from the SSVT. Therefore, the caller must have addressability to the SSVT. Otherwise, the first MPC is obtained from the TVT.
Syntax

FIND Syntax:

\[ \text{FIND} \quad \text{NAME}=(\text{label}) \quad \text{SEQNO}=(\text{label}) \quad \text{ANCHOR}=(\text{label}) \quad \text{SYSID}=(\text{label}) \quad \text{MAIN_MASK}=(\text{label}) \quad \text{MPC_REG}=(\text{reg}) \quad \text{NOTFOUND}=(\text{label}) \quad \text{NORMAL}=(\text{label}) \]

UPDATE_FLAG Syntax:

\[ \text{UPDATE_FLAG} \quad \text{FLAG}=(\text{label},\text{label},\text{constant}) \quad \text{MAIN_MASK}=(\text{label}) \quad \text{MPC_REG}=(\text{reg}) \]

CREATE_MASK Syntax:

\[ \text{CREATE_MASK} \quad \text{SYSTEM_LIST}=(\text{label}) \quad \text{MAIN_MASK}=(\text{label}) \quad \text{SYSTEM_COUNT}=(\text{label}) \]

CREATE_SYSTEM_LIST Syntax:

\[ \text{CREATE_SYSTEM_LIST} \quad \text{SYSTEM_LIST}=(\text{label}) \quad \text{MAIN_MASK}=(\text{label}) \quad \text{SYSTEM_COUNT}=(\text{label}) \]

Parameters

FUNC=

Specifies the function to be performed.

FIND=

This function is used to find a particular MPC by one of the following:

- System name.
- Main processor sequence number.
- Main mask.

Note: It is assumed that the main mask has only one bit on. Otherwise, the first system whose bit is on in the main mask will be returned.

The caller can specify the register that will contain the MPC address if it is found. If a register is not specified, the MPC address will be returned in R1.

ANCHOR=

Specifies the first pointer to the MPC chain which should be searched according to the search parameters. The default is to use the MAINPROC chain built the last time JES3 was initialized. The pointer to the MAINPROC chain used when this parameter is allowed to default depends on the environment in which the IATXMPC macro is called.
**IATXMP**

**Note:** JES3 uses this parameter internally. In most cases this parameter should be omitted and allowed to default.

**UPDATE_FLAG**=
This function is used to set or reset a flag in all MPC’s, or set or reset a flag in MPC’s that appear in the main mask that was specified.

**CREATE_MASK**=
This function is used to create a main mask from a list of system names that were specified.

**Note:** The main mask should be cleared by the caller before invoking this function.

**CREATE_SYSTEM_LIST**=
This function is used to create a list of systems from a main mask. The caller must provide a work area that is large enough to contain all of the systems in the main mask.

When processing is complete, the field pointed to by the SYSTEM_COUNT parameter or the register specified by the SYSTEM_COUNT parameter will contain the number of systems in the list.

**NAME**=
Specifies the system name associated with the MPC.

**SEQNO**=
Specifies the one byte main processor sequence number associated with the MPC.

**SYSID**=
Specifies the sequence number assigned to the spooled MPC record during JES3 initialization.

**Note:** JES3 uses this parameter internally. In most cases the SEQNO parameter should be used instead.

**FLAG**=
Specifies the flag to be set or reset in the MPC’s associated with the request. This parameter takes the following form: FLAG=label(The flag field name), label(The flag bit name),constant(Indicates whether the flag should be set (SET) or reset (RESET))

**MAIN_MASK**=
Specifies the main mask associated with the request. The specified function will be performed on those MPC’s that appear in the main mask.

**SYSTEM_LIST**=
Specifies the address of a list of eight byte system names associated with the request. The number of systems in the list is specified by the SYSTEM_COUNT parameter.

**SYSTEM_COUNT**=
Specifies the number of systems in the system list specified by the SYSTEM_LIST parameter. If the label form of this parameter is used, it must be a fullword field that contains the number of systems in the list.

**MPC_REG**=
Specifies the register to contain the MPC address. If not specified, R1 is used.

**NOTFOUND**=
Specifies the address to return control to if the MPC is not found.
NORMAL=
Specifies the address to return control to if processing is successful.

Linkage
This macro expands in-line.

Restrictions
None.

IATXMVDA (Access or Release a JES3 Table in Common Storage)
This macro is used to control access to a JES3 table in Common Storage that gets rebuilt during a JES3 restart. The purpose of this controlled access is to ensure that if a JES3 restart occurs, the new version of the table will be created without impacting any address space that may be accessing an old version. The old version is freed as soon as the last address space that is accessing the table releases its access.

All functions expand code in-line.

Syntax

```
IATXMVDA
  ^label
  |    ^-FUNC=RELEASE
  |       ^USE, TABLE=SETNAMES
  |          ^DYNALDSN
  |             ^SETUNITS
  |                ^MAIN_FIELD=MPSETTRE
  |                    ^TOKEN=address(reg)
```

Parameters

**FUNC=**
Specifies the function that you want to perform on the table.

**USE**
Indicates that you want to access the table. The address of the table is returned to you in register 1.

**RELEASE**
Indicates that you are no longer using the table.

**TABLE=**
Specifies whether you are accessing the SETUNITS, SETNAMES, or DYNALDSN table. This parameter is valid only with FUNC=USE.

**TOKEN=**
Specifies the address of an area containing a token representing your access to the table. If FUNC=USE and TABLE= are both specified, IATXMVDA returns a TOKEN to you in the address you specify for TOKEN=. If FUNC=USE is
specified and TABLE= is not specified, or if FUNC=RELEASE is specified, 
TOKEN= must point to a token returned by a previous IATXMVDA call with 
FUNC=USE and TABLE specified.

**MAIN_FIELD=MPSETTRE**

Identifies that the SETUNITs field to be returned should be MPSETTRE. If this 
parameter is not specified, the field to be returned is MPUNITs. This parameter 
can be specified only when FUNC=USE. It also requires that either 
TABLE=SETUNITs is specified, or the TOKEN= area points to a token returned 
by a previous IAYTXMVDA call with FUNC=USE,TABLE=SETUNITs specified. 
When MAIN_FIELD is specified and TABLE is not specified, the USE function 
does not count as an additional use by the same address space. Therefore, 
only one RELEASE for the table will be needed in order to release the table.

**Linkage**

This macro uses BALR R14,R15 linkage.

**Restrictions**

Do not use this macro in the JES3 address space. You may use this macro in a 
user address space. You must have addressability to IATYSVT.

**IATXMVXM (Move Data Among CSA, Private Area, and JES3AUX)**

The IATXMVXM macro moves data from/to the CSA, a private area, or the JES3 
 auxiliary address space (JES3AUX).

The routine issuing this macro must include the IATYSVT and IATYPCD mapping 
macros.

**Syntax**

```
IATXMVXM
  label
  ,ORG=JES3AUX,DEST=CSA
  ,PRIVATE
  ,DEST=JES3AUX
  ,FROM=(R1)address
  ,TO=(R15)address
  ,COUNT=(R0)count
  ,ERROR=address
```

**Parameters**

**ORG=**

Specifies the area from which data is to be moved.

**DEST=**

Specifies the area to which data is to be moved.

**FROM=**

Specifies the address from which data is to be moved.

**TO=**

Specifies the address to which data is to be moved.
COUNT=
   Specifies the number of bytes to be moved.

ERROR=
   This parameter specifies the address to which the macro processing routine
   returns if an error occurs. When control is passed to the address, register 0
   contains a dump code and register 15 contains a return code, unless otherwise
   specified. The return codes are described below.

   If an error occurs and ERROR was not specified, the dynamic support program
   (DSP) will be ended using the FAILDSP macro.

Note: Valid parameter combinations are:

<table>
<thead>
<tr>
<th>ORG</th>
<th>DEST</th>
</tr>
</thead>
<tbody>
<tr>
<td>JES3AUX</td>
<td>CSA</td>
</tr>
<tr>
<td>JES3AUX</td>
<td>PRIVATE</td>
</tr>
<tr>
<td>CSA</td>
<td>JES3AUX</td>
</tr>
<tr>
<td>PRIVATE</td>
<td>JES3AUX</td>
</tr>
</tbody>
</table>

Return Codes
Following are the return codes returned in register 15:

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Normal completion</td>
</tr>
<tr>
<td>Not 0</td>
<td>Abnormal completion</td>
</tr>
</tbody>
</table>

Linkage

This macro does not use ASAVE linkage. The routine issuing the macro must put
the address of a 72-byte save area into register 13. The macro saves registers 2
through 13, but registers 0, 1, 14, and 15 are destroyed.

Restrictions

Do not use this macro in any routine that might run in a C/I FSS address space or
under the JES3 auxiliary task.

IATXNGRS (Send a Negative Response)

The IATXNGRS macro sends a negative response when an error is found in an
inbound stream. The error can be found by presentation services (PS) or by a
VTAM RECEIVE macro request.

Syntax

```
label: IATXNGRS    LUCB= (R1) , CODE= (R0)  
       (reg) , X = 'nnnn'  
       (R2) , BFE= (reg)  
       , address  
```
Parameters

**LUCB=**
Specifies the address of a logical unit control block (LCB) that identifies a session.

**CODE=**
(R0) or (reg) specifies the address of the sense bytes which identify the type of error. X'n'nnn' specifies the actual VTAM sense code.

**BFE=**
Specifies the address of the buffer information field (BFE) of the request/response unit (RU) that is in error.

Linkage

This macro uses ASAVE linkage.

Restrictions

Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task.

---

**IATXNTS (Search the Node Table)**

The IATXNTS macro searches the networking node table to verify the validity of a destination.

On exit from this macro, register 1 points to the node table entry that satisfies the request. The node table entry is mapped by macro IATYNJY.

**Note:** If the “not available” return is taken by the IATXNTS macro, register 1 is the same as on entry.

Syntax

```
IATXNTS NODE=address,NAVAIL=address

,PATH=NO

,PATH=NO,NORMAL=address

,PATH=YES,TYPESNA=address,TYPETCP=address

```

Parameters

**NODE=**
Specifies the address of an 8-byte field containing the name of the node to be searched for. The node name must be left-justified and padded with blanks. This parameter is required.

**NAVAIL=**
Specifies the address to branch to when the required node name or a path node name (PATH=YES) is not found in the node table. This parameter is required.
PATH=
YES specifies that the search is to continue through the node table until an entry is found where the name and the path are the same (that is, the service routine locates the path to the given node). NO specifies that the pointer to the node table entry for the specified node is to be returned.

The PATH= parameter is optional. If you omit it, PATH=NO is the default.

TYPEBSC=
Specifies the address to return to if the node entry found represents a BSC connected node (TYPE=BSC on the NJERMT statement). This parameter is optional, and requires that TYPESNA and TYPETCP also be specified. TYPEBSC cannot be specified if the NORMAL= parameter is specified.

If none of TYPEBSC, TYPESNA, TYPETCP, or NORMAL= is specified, the default is NORMAL=*, which returns to the next sequential instruction after the macro.

TYPESNA=
Specifies the address to return to if the node entry found represents a SNA connected node (TYPE=SNA on the NJERMT statement). This parameter is optional, and requires that TYPEBSC and TYPETCP also be specified. TYPESNA cannot be specified if the NORMAL= parameter is specified.

If none of TYPEBSC, TYPESNA, TYPETCP, or NORMAL= is specified, the default is NORMAL=*, which returns to the next sequential instruction after the macro.

TYPETCP=
Specifies the address to return to if the node entry found represents a TCP/IP connected node (TYPE=TCPIP on the NJERMT statement). This parameter is optional, and requires that TYPEBSC and TYPESNA also be specified. TYPETCP cannot be specified if the NORMAL parameter is specified.

If none of TYPEBSC, TYPESNA, TYPETCP, or NORMAL= is specified, the default is NORMAL=*, which returns to the next sequential instruction after the macro.

NORMAL=
Specifies the address to return to if the node table entry is found. It does not matter if the type of connection to the node is BSC, SNA, or TCP/IP. When NORMAL= is specified, this macro masks the node connection type from the caller. This parameter is optional, and it cannot be specified if one of TYPEBSC, TYPESNA, and TYPETCP is specified. The default is NORMAL=*, which returns to the next sequential instruction after the macro.

Linkage

This macro uses ASAVE linkage.

Restrictions

Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task or any address space other than JES3.

Usage Notes

If you specify PATH=NO, you should also specify the NORMAL= parameter because if you are not looking for the path, you are not really interested in the connection. Basically, you’re mainly interested in determining if the node is defined.
Conversely, if you specify PATH=YES to determine the type of connection to the next node, you must also specify all of the TYPEBSC=, TYPESNA=, and TYPETCP= parameters.

Do not use PATH=NO along with TYPEBSC=, TYPESNA=, and TYPETCP=, because unpredictable results will occur.

**IATXOSBM (Scan the MOSE/OSS Chain for Work to Go to BDT)**

The IATXOSBM macro is used to scan the MOSE/OSS chain for output service work (such as data needed for BDT processing), and to update the OSS and OSE control blocks with current status data about the work for BDT. This status information is referred to as registry status.

### Syntax

```
IATXOSBM TYPE=GET label
   ,PARM=(R1),NORMAL=address
   ,ERROR=address

EOD:
   ,EOD=address

REC:
   ,REC=NORMAL ,ABNORMAL
```

### Parameters

**TYPE=**

The TYPE= parameter specifies the type of action required. This parameter is required. The input to and output from the service routine is a parameter list, which is described below. The service routine will either return +0 (no available work, use EOD=), +4 (use ERROR=), or +8 (use NORMAL=).

**GET**

Indicates to the IATOSBM service routine that the MOSE/OSS chain should be searched for the next available piece of SNA/NJE work.

**ASSIGN**

Indicates to the IATOSBM service routine that the MOSE/OSS chain should be searched and assigned the next available piece of SNA/NJE work.
INQUIRY

Finds the SNA/NJE stream according to the selection criteria set in the OSB. 368 shows the selection criteria. The job number and group ID to be checked is contained in the parameter list mapped by IATYOSB.

PUT

Indicates to the service routine that the registry status in the OSS and OSE is to be updated. The job to be updated and the new status are contained in the parameter list (mapped by IATYOSB).

RECOVERY

Indicates to the service routine that recovery is to occur. During recovery, the entire SNA/NJE work queue is updated. When TYPE=RECOVERY is specified, the service routine attaches an FCT to update the OSS/MOSE/OSE for all SNA/NJE jobs with the new status. This differs from TYPE=PUT, where only one SNA/NJE job is updated.

REC=

The REC= parameter is valid only when you also specify TYPE=RECOVERY. If you specify TYPE=RECOVERY, you must also specify REC=.

NORMAL

Normal recovery is to occur.

ABNORMAL

Abnormal recovery is to occur.

PARM=

The PARM= parameter specifies the address of a parameter work area (mapped by IATYOSB) that is passed to module IATOSBM. This parameter is required.

EOD=

Specifies the address to branch to if module IATOSBM finds no work. This parameter is required, but is valid only when the TYPE=GET, TYPE=ASSIGN, or TYPE=INQUIRY parameters are specified.

ERROR=

Specifies the address to branch to if module IATOSBM encounters an error in processing the request. This parameter is required, but is valid only when the TYPE=PUT or TYPE=ASSIGN parameters are specified.

NORMAL=

Specifies the address to branch to after normal completion of the specified request. This parameter is optional.

The parameter list required for the IATXOSBM macro is mapped by the IATYOSB macro. The parameter list is referred to as the OSB, which means “output service BDT” control block. Register 1 is set to the OSB by the PARM= keyword on the IATXOSBM macro. On return from module IATOSBM, register 1 points to the OSB. The OSB contains the following data:
For a GET request, the parameter list is zero on input to the service routine, and contains the information needed to build a BDT/NJE transaction upon exit. For PUT requests, the job name, job number, and group ID identify the job to be updated; and the status flags indicate what the new status is to be. For TYPE=RECOVERY requests, the REC= parameter indicates whether the recovery is to be normal or abnormal. For TYPE=ASSIGN requests, the job number identifies the job to be updated.

For TYPE=INQUIRY requests, the SNA/NJE stream is found according to the following selection criteria specified in the OSB. The list shows the field description followed by the field's label.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Job number</td>
<td>OSBJOBNO</td>
</tr>
<tr>
<td>Priority</td>
<td>OSBPRTY</td>
</tr>
<tr>
<td>Group ID</td>
<td>OSBGRPID (see note below)</td>
</tr>
<tr>
<td>Destination</td>
<td>OSBDEST</td>
</tr>
<tr>
<td>Destination path</td>
<td>OSBPATH</td>
</tr>
<tr>
<td>Pending work</td>
<td>OSBSPEND</td>
</tr>
</tbody>
</table>

Note: The “group id” is generated by the JES3 output service function and represents a group of work for BDT. For example, a job processed by JES3 could create three “spin-off” data sets, or perhaps, data sets going to different destinations. All of these data sets require BDT processing. A unique group id is assigned to each of the three data sets so that the system operator can inquire about them individually.

Linkage

This macro uses ASAVE linkage.

Restrictions

Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task.

IATXOSCI (Output Service Input Data Set Close)

The IATXOSCI macro closes the input data set currently being referenced for JSAM multirecord reads.
IATXOSCI

Syntax

\[ \text{IATXOSCI} \quad \text{TYPE=T} \quad \text{NORMAL=address} \]

Parameters

**TYPE=**
Specifies that the buffers obtained by the initial IATXOSCI macro call are not to be freed because the close is temporary. If omitted, the buffer will be freed.

**NORMAL=**
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

Linkage

This macro uses ASAVE linkage.

Restrictions

Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task.

IATXOSCO (End Processing on Output Device)

The IATXOSCO macro is used to end processing of a SYSOUT data set on an output device and clean up for possible termination.

Syntax

\[ \text{IATXOSCO} \quad \text{REJECT=address} \quad \text{LABEL=NONE}, \text{TRLR}, \text{FINAL}, \text{JCLOSE}, \text{SUSPEND} \]

Parameters

**REJECT=**
Specifies the address to which control returns if an error occurred while processing trailer records.

**LABEL=**

- **NONE**
  Specifies no trailer records are to be written to the device.

- **TRLR**
  Specifies trailer records are to be written.
IATXOSG

FINAL
Specifies that the device is to be released for subsequent use by another
writer or DSP.

JCLOSE
Specifies that normal output service close processing is bypassed; however,
the JESCLOSE macro is issued to close the device.

SUSPEND
Specifies that the sending of SYSOUT data to a remote device is to be
suspended, thus allowing operator messages to be written to the device.

PARM=
Specifies the address of the parameter list used by the IATXOSCO service
routine.

NORMAL=
This parameter specifies that after the macro processing routine executes, the
issuing program can resume execution at an instruction other than the
instruction that immediately follows the macro. If NORMAL= is not coded, the
issuing program resumes execution at the instruction immediately following the
macro.

Linkage

This macro uses ASAVE linkage.

Restrictions

Do not use this macro in any routine that might run in a C/I FSS address space or
under the JES3 auxiliary task.

IATXOSG (Obtain an Input Data Record)

The IATXOSG macro requests a data record from the output service spool input
routine. The request may be for the next sequential record in the input file, or for
any record in the file. A request for a nonsequential record which exceeds the limits
of the file will be positioned backward to the start-of-file or forward to end-of-file.

If the request is for a spanned record, one segment of the record is obtained at a
time. When a segment is obtained, control returns to the address specified using
the EOF parameter.

Upon return from the macro call, register 1 contains the address of the record. The
low-order byte of register 0 contains the length of the record. For records that are
not spanned, the high-order byte of register 0 contains the carriage control
information. This information is in the same format as mapped by the DATCC field
of the IATYDAT data area. For spanned records, the high-order byte of register 0
indicates whether the segment being obtained is the first or last segment or an
intermediate segment.

See the EOF parameter below for an explanation of the contents of registers 2 and
3 upon return.

Syntax

```
IATXOSG HOLD= (reg) address FREE= (reg) address

```
Parameters

HOLD=

(reg)

If (reg) is specified, the ‘HOLD’ byte is in the low-order byte of the register.

address

Specifies the address of a byte indicating which of a set of up to eight CCW areas are currently in use by a device-dependent routine.

FREE=

(reg)

If (reg) is specified, the ‘FREE’ byte is in the low-order byte of the register. If HOLD= and FREE= specify the same register, the HOLD byte is in the second byte of the register and the FREE byte is in the fourth byte of the register.

address

Specifies the address of a byte indicating which of the CCW areas are available. The HOLD and FREE parameters are used by the spool input routine to determine which data buffers are available for JSAM to read into.

COUNT=

Specifies the backward or forward record reposition count. Backward repositioning is specified by a negative number.

NAVAIL=

Specifies the address of the routine to receive control when the last request could not be processed. The reasons for this are that a read is required to obtain the required record and all buffers are in use, or backward/forward repositioning is requested and a buffer is still held.

EOF=

Specifies the address of a routine to receive control when the end of a buffer is reached. If registers 2 and 3 are zero, the record is wholly contained in the buffer; if registers 2 and 3 are not zero, the record is split across two buffers, and register 2 contains the length of the second part of the record with register 3 containing the address of the second part of the record.

EOD=

Specifies the address of a routine to receive control when the end-of-file or top-of-file condition is detected.

ERROR=

This parameter specifies the address to which the macro processing routine returns if an error occurs. When control is passed to the address, register 0 contains a dump code and register 15 contains a return code, unless otherwise specified. The return codes and dump codes are described below. For a description of every dump code, see \textit{z/OS JES3 Diagnosis}.

If an error occurs and ERROR was not specified, the dynamic support program (DSP) will be ended using the FAILDSP macro.
NORMAL=

This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

Return Codes

Following are the return codes and dump codes returned in registers 0 and 15, respectively:

<table>
<thead>
<tr>
<th>Return Code</th>
<th>In Hex</th>
<th>Dump Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>04</td>
<td>719</td>
<td></td>
<td>Recoverable validation failure</td>
</tr>
<tr>
<td>08</td>
<td>721</td>
<td></td>
<td>Incorrect buffer, track address</td>
</tr>
<tr>
<td>08</td>
<td>731</td>
<td></td>
<td>Bad character count</td>
</tr>
<tr>
<td>0C</td>
<td>724</td>
<td></td>
<td>Skip option I/O error recovery</td>
</tr>
<tr>
<td>10</td>
<td>725</td>
<td></td>
<td>Unrecoverable I/O error</td>
</tr>
<tr>
<td>14</td>
<td>702</td>
<td></td>
<td>Read past EOD</td>
</tr>
<tr>
<td>18</td>
<td>722</td>
<td></td>
<td>Unrecoverable validation failure</td>
</tr>
<tr>
<td>1C</td>
<td>701</td>
<td></td>
<td>File not open</td>
</tr>
<tr>
<td>20</td>
<td>724</td>
<td></td>
<td>Skip option I/O error recovery with EOD</td>
</tr>
<tr>
<td>40</td>
<td>731</td>
<td></td>
<td>Inadequate input/output buffers defined for processing stream mode data to a line mode device.</td>
</tr>
</tbody>
</table>

Linkage

This macro issues a BALR R14,R15 for linkage.

Restrictions

Do not use this macro in any routine that might run in a C/I FSS address space.

IATXOSOI (Output Service Input Data Set Open)

The IATXOSOI macro opens an input file from which JSAM multireads are to be done.

Syntax

```
/IATXOSOI

label IATXOSOI FDB=(R1) address

BUFFERS=number

,ERROR=address

,NORMAL=address
```

Parameters

FDB=

Specifies the address of the FDB from which the MRF will be read.

BUFFERS=

Specifies the number of JSAM buffers to attempt to fill with each JSAM multiread request.
ERROR=
  Specifies the address to which control is returned if the current open request
  cannot be satisfied.

NORMAL=
  This parameter specifies that after the macro processing routine executes, the
  issuing program can resume execution at an instruction other than the
  instruction that immediately follows the macro.

Return Codes
Following are the return codes and dump codes returned in registers 0 and 15,
respectively:

<table>
<thead>
<tr>
<th>Return Code In</th>
<th>Dump Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>08</td>
<td>721</td>
<td>Incorrect buffer, track address</td>
</tr>
<tr>
<td>0C</td>
<td>722</td>
<td>Validation failure</td>
</tr>
<tr>
<td>10</td>
<td>725</td>
<td>Unrecoverable I/O error</td>
</tr>
<tr>
<td>14</td>
<td>731</td>
<td>Bad character count</td>
</tr>
</tbody>
</table>

Linkage

This macro uses ASAVE linkage.

Restrictions

Do not use this macro in any routine that might run in a C/I FSS address space. If
this macro is used in a routine running under the auxiliary task, it can be used only
to note an input record.

IATXOSOO (Open and Initialize Output Device)

The IATXOSOO macro opens and initializes an output device.

Syntax

```
IATXOSOO(REJECT=address).PARM=(R1)address

YES,NEWPAGE=NO

LABEL=VOL

OS

SETUP

REAL
```

Parameters

REJECT=
  Specifies the address to which the open routine returns when it is unable to
  complete initialization. When the reject is the result of the RJPCAN return from
  the JESEXCP macro, the REJECT= parameter must specify the address of the
caller of the PUT.
**PARM=**
For LABEL=NONE, specifies the address of the JES3 header record describing the file which follows. For LABEL=VOL, DS, or SETUP, specifies the address of the output service element (OSE) associated with the data set which follows.

**NEWPAGE=**

**YES**
Specifies that a skip to channel 1 is to be inserted by JES3. On a 3800 printer, translate table O will be selected. NEWPAGE=YES is valid only if you specify LABEL=NONE or LABEL=REAL.

**NO**
Specifies that JES3 is not to insert a skip to channel 1 or to select a 3800 translate table.

**LABEL=**

**NONE**
Specifies that a new file follows and data attributes must be reinitialized.

**VOL**
Specifies that a new JOB has been selected.

**DS**
Specifies that a new data set follows.

**SETUP**
Specifies that the device is to be initialized but header pages are not to be generated.

**REAL**
Specifies that the device is to be operated at the STARTIO level.

**NORMAL=**
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

**Linkage**
This macro uses ASAVE linkage.

**Restrictions**
Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task.

**IATXOSP (Write Out Queued Output Requests)**
The IATXOSP macro passes data pointers and lengths to output service modules IATOSPR, IATOSP, IATOSSN, or IATOSWD causing any queued requests to be immediately written to the particular device associated with each module. For example, module IATOSSN’s request would be written to a SNA RJP device.

**Syntax**

```
IATXOSP label REJECT=address DATA=list COUNT=count
```

**Options**
DATA=

\[\text{DATA=} \rightarrow \text{address} \]

\(R1\)

COUNT=

\[\text{COUNT=} \rightarrow \text{number} \]

LIST=

\[\text{LIST=} \rightarrow \text{address} \]

Options:

\[\text{TRUNC=} \rightarrow \text{NO, YES} \]

\[\text{CC=} \rightarrow \text{A, C, M, OP, S} \]

\[\text{NORMAL=} \rightarrow \text{address} \]

Parameters

**REJECT=**

Specifies the address to which control returns if output cannot complete. When the reject is the result of the RJPCAN return from the JESEXCP macro, the REJECT parameter must specify the address of the caller of the PUT.

**DATA=**

Specifies the address of the record. The address points to a record below 16 megabytes in virtual storage. If you specify DATA=, you may not specify the LIST parameter.

**COUNT=**

Specifies the length of the record. If coded, the LIST parameter may not be coded.

**LIST=**

(Used only when the record to be written is split across two buffers.) Specifies the address of a 4-word parameter list. The first word specifies the length of the record segment contained in the first buffer; the second specifies the address of the first segment. The third word specifies the length of the record segment contained in the second buffer; the fourth word specifies the address of the second segment.

The parameter list pointed to by LIST= may reside above 16 megabytes in virtual storage. The buffers (whose addresses are in the second and fourth words of the parameter list) must be below 16 megabytes.

The LIST= parameter may not be specified if the DATA parameter, COUNT parameter, or both are specified.
TRUNC=

YES
Specifies that the CCW is to be truncated and queued for I/O. Control is to be returned immediately.

NO
Specifies that the CCW is to be added to the channel program and control is to be returned immediately.

CC=
Specifies the type of carriage control information in the first byte of the records.

A
Specifies ASA carriage control characters.

C
Specifies the carriage control specification is in the high-order byte of the length field (specified by the COUNT parameter). The meaning of this byte is:
- 00 – No carriage control
- 20 – ASA carriage control characters
- 40 – Machine carriage control characters

M
Specifies machine carriage control characters.

O
Specifies records do not contain carriage control character.

OP
Specifies that the record has a machine control character (a CCW operation code) that does not require validity checking.

S
Specifies information to initialize the 3800. Includes FLASH, COPIES, CHARS, and MODIFY information.

NORMAL=
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

Note: If you are both:
- Reading spool input records using the IATXOSG macro AND
- Using two IATXOSP macros without an intervening IATXOSG macro

you can avoid a failure in JES3 output services by saving the contents of the WTRIPARM field (returned in register 1 from IATXOSP) after each invocation of IATXOSP. The WTRIPARM field is subsequently used by IATXOSG to locate the next SYSOUT record in the sequence and will not be valid after the second IATXOSP macro.

One way to save the contents of the WTRIPARM field (after the first IATXOSP macro) is to OR the last byte of register 1 with the WTRIPARM field as follows:

```
Saving the WTRIPARM Field
N R1,=x'000000FF'
O R1,WTRIPARM
ST R1,WTRIPARM
```

Make sure to execute these instructions before issuing the second IATXOSP macro.
IATXOSP might cause a long AWAIT to be issued. Therefore, be aware of what resources you are holding because certain AWAITs are not posted without manual intervention; such as intervening on a device, or entering an operator command.

**Linkage**

This macro issues a BALR R14, R15 for linkage.

**Restrictions**

1. Do not use this macro in any routine that might run in a C/I FSS address space.
2. The data pointed to by the IATXOSP macro must reside below 16 megabytes in virtual storage. This means that if your code issues an AGETMAIN macro, the AGETMAIN must specify the LOC=(BELOW) parameter.
3. For output requests that will be handled at the STARTIO level (such as devices opened with IATXOSOO...LABEL=REAL...), when the device is operated at the STARTIO level, the user of the macro must page-fix the data pointed to in the IATXOSP macro and also obtain storage below 16 megabytes for the data.

**IATXOSPC (Access Output Service Elements for Process-SYSOUT Requests)**

The IATXOSPC macro accesses output service elements (OSEs) for process SYSOUT requests to set up the work queues for external (non-JES3) writers.

**Syntax**

```
/SM590000/IATXOSPC
  TYPE=GET,EOD=address
    RQ=
    PARM=(R1)address
    NORMAL=address

RQ=:
  RQ=(R0)address
```

**Parameters**

- **PARM=**
  Specifies the address of a work area (mapped by macro IATYWSP).

- **TYPE=**
  Specifies the action required.

  - **GET**
    Specifies a request for a data set because JES3 output service is looking for work to do. If RQ= is coded, the request is for a specific job. The EOD= parameter is required in this case.

  - **PUT**
    Specifies a request to change one or more data sets for a job. The ERROR= and RQ= parameters are required if TYPE=PUT is used.
RQ=
    Specifies the address of the resident job queue (RESQUEUE) entry. This
    parameter is optional for a TYPE=GET request but is required for a TYPE=PUT
    request.

EOD=
    Specifies the address to which the routine returns when the end of the output
    service queue is reached without satisfying the request. If the request was for
    an OSE in a particular job, then the job doesn’t contain an OSE satisfying the
    request. If the request was for any OSE, then there is no OSE in the output
    service queue that satisfies the request.

    EOD= is valid only when you specify TYPE=GET, in which case EOD= is
    required.

ERROR=
    Specifies the address to which the routine returns when the end of the output
    service queue is reached without satisfying the request, such as not finding a
    specific job.

    ERROR= is valid only when you specify TYPE=PUT, in which case ERROR= is
    required.

Error Return Codes
Following are the error reason codes returned in register 0:

<table>
<thead>
<tr>
<th>Reason Code In Hex</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>Either bad job name, job number, or RSQ</td>
</tr>
<tr>
<td>02</td>
<td>Invalid user of PSO with group ID</td>
</tr>
<tr>
<td>03</td>
<td>RSQ required but missing</td>
</tr>
<tr>
<td>04</td>
<td>Job is being dumped</td>
</tr>
<tr>
<td>05</td>
<td>No output</td>
</tr>
<tr>
<td>06</td>
<td>Invalid search argument</td>
</tr>
<tr>
<td>07</td>
<td>AWRITE error</td>
</tr>
<tr>
<td>08</td>
<td>Invalid data</td>
</tr>
<tr>
<td>FF</td>
<td>Severe error – dump already generated</td>
</tr>
</tbody>
</table>

NORMAL=
    Specifies the address to which the routine returns when the request has been
    satisfied.

Linkage
This macro uses ASAVE linkage.

Restrictions
Do not use this macro in any routine that might run in a C/I FSS address space or
under the JES3 auxiliary task.

IATXOSPM (Call the PPQ Manager or Writer Functions)

The IATXOSPM macro calls the writer-related and pending page queue
(PPQ)-related routines in module IATOSWP to perform a variety of functions that
are described under the TYPE parameter below.
Syntax

```
label IATXOSPM

<table>
<thead>
<tr>
<th>TYPE=</th>
<th>OSEUPDT, EOD=addr, BUSY=addr, PPQ=</th>
<th>OPT2</th>
<th>OPTIONS1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>OSESEL</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>OSESEL</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CKPT</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CKPT</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>OSSEL</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>OSSEL</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DSEND</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DSEND</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>JOBEND</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>JOBEND</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>OSUPDT</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>OSUPDT</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>COMPLETE, PPQ=</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>COMPLETE, PPQ=</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RSCD, PPQ=</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RSCD, PPQ=</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RSTR, PPQ=</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RSTR, PPQ=</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SMF6, PPQ=</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SMF6, PPQ=</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>WOSERD, PPQ=</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>WOSERD, PPQ=</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>WOSELT, PPQ=</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>WOSELT, PPQ=</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RSCSRC, PPQ=</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RSCSRC, PPQ=</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ENQUEUE, PARM=spid</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ENQUEUE, PARM=spid</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DEQUEUE, PARM=spid</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DEQUEUE, PARM=spid</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>JSTART, PARM=spid</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>JSTART, PARM=spid</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SYNC, PARM=(IDENT, dest)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SYNC, PARM=(IDENT, dest)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CHKPT, (reg)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CHKPT, (reg)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NUMBER, dest</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NUMBER, dest</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CANCEL, dest</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CANCEL, dest</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CLEAR</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CLEAR</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>INQUIRY, PARM=((reg x), (reg y))</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>INQUIRY, PARM=((reg x), (reg y))</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

OPTIONS1:

```
OPT2

<table>
<thead>
<tr>
<th>ERROR=address</th>
</tr>
</thead>
</table>
```

OPT2:

```
SAVE=YES

<table>
<thead>
<tr>
<th>NORMAL=address</th>
</tr>
</thead>
</table>
```

PPQ=:

```
<table>
<thead>
<tr>
<th>PPQ= (reg)</th>
</tr>
</thead>
</table>
```

Parameters

**TYPE=**

Specifies the type of function that is to be performed.

**JOBSEL**

Indicates that the request is for the writer-related job select function.
OSESEL
Indicates that the request is for the writer-related OSE select function.

DSSEL
Indicates that the request is for the writer-related data set select function.

CKPT
Indicates that the request is for the writer-related checkpoint function.

DSEND
Indicates that the request is for the writer-related data set end-at-channel function.

JOBEND
Indicates that the request is for the writer-related job end function.

DSUPDT
Indicates that the request is for the writer-related data set update function.

OSEUPDT
Indicates that the request is for the writer-related output service element (OSE) update function. This function will update all the data set entries in a work OSE. If OSEUPDT is specified, the ERROR= parameter cannot be specified. Also, EOD= and BUSY= are required if you specify OSEUPDT.

COMPLETE
Indicates that the request is for the writer-related event completion processing function.

RSCD
Indicates that the request is for the writer-related reschedule processing function.

RSTR
Indicates that the request is for the writer-related restore function.

SMF6
Indicates that the request is for the writer-related SMF6 create and writer function.

WOSERD
Indicates that the request is for the writer-related read and work output scheduling element (WOSE) function.

WOSEWT
Indicates that the request is for the writer-related write WOSE function.

WOSERL
Indicates that the request is for the writer-related release WOSE function.

ENQUEUE
Indicates that the request is for the PPQ-related enqueue function.

DEQUEUE
Indicates that the request is for the PPQ-related dequeue function.

JSTART
Indicates that the request is for the PPQ-related job on writer function.

SYNC
Indicates that the request is for the PPQ-related synchronization function. This causes the data sets to be repositioned backward and rescheduled based on a new reference point.
INQUIRY
   Indicates that the request is for the PPQ-related inquiry command function.

PPQ=
   Specifies the address of the PPQ entry that is to be used for the request.

PARM=
   spid
      Specifies the stacked page identifier.
   tpid
      Specifies the transfer page identifier.

(IDENT,dest)
   Specifies that data sets having page identifiers greater than the value specified at the dest are to be rescheduled. The dest subparameter is the name of the halfword that contains the unadjusted identifier of the page to which the writer is to page back.

(CHKPT,(reg))
   Specifies that synchronization is to occur based on the last stacked PPQ entry. The contents of the register specified by (reg) must be negative and represent the number of pages to go back from the checkpoint PPQ.

(NUMBER,dest)
   Specifies that synchronization is to be done based on the number of sheets of paper. The dest subparameter is the name of the word that contains the number of pages to page backward (negative) or forward (positive). The number can be supplied in register 0 or in register 10.

(CANCEL,dest)
   Specifies that synchronization is to occur but the deleted PPQ entries are not to be rescheduled. The dest subparameter can be one of the following:
   • ‘DS’ – Synchronize the writer to the start of the most recent data set.
   • ‘OSE’ – Synchronize to the start of the most recent OSE. Pass each incomplete PPQ entry to the COMPLETE routine.
   • Resqueue Address – Remove all traces of the particular job for the specified RQ.

(CLEAR)
   Indicates that all data represented in the PPQ is to be rescheduled. Note that the dest subparameter is not valid with the CLEAR subparameter.

SAVE=
   Indicates whether the ASAVE linkage routine is to be used.
   YES
      Indicates that ASAVE linkage is to be used and must be used with TYPE=SMF6 records.
   NO
      Indicates that ASAVE linkage is not to be used. For the writer-related functions, the contents of registers 0, 1, 6-10, and 14 are unpredictable upon return. For the PPQ-related functions, the contents of registers 0-5, 9, 10 and 14 are unpredictable. Most functions can call another function with SAVE=NO. In such a case, the contents of registers 0-10 and 14 are unpredictable.

EOD=
   Indicates that there are some data set entries pending update. EOD= is required if you specify OSEUPDT.
IATXOSPM

BUSY=
Indicates that there are some entries still unprocessed, but none are pending update completion. BUSY= is required if you specify OSEUPDT.

ERROR=
This parameter specifies the address to which the macro processing routine returns if an error occurs. When control is passed to the address, register 0 contains a dump code and register 15 contains a return code, unless otherwise specified.

If an error occurs and ERROR was not specified, the dynamic support program (DSP) will be ended using the FAILDSP macro.

NORMAL=
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

Linkage
The type of linkage used by this macro depends on what is specified for the SAVE parameter (see below).

Restrictions
Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task.

IATXOSSC (Access Output Service Elements for Subsystem Requests)
The IATXOSSC macro accesses output service elements (OSEs) for subsystem requests, such as those needed for directing output to normal printers like 1403s or 3211s. JES3 uses this macro to set up work queues for the output service function.

Syntax

Parameters

PARM=
Specifies the address of a work area (mapped by macro IATYWSP).
EOD=
    Specifies the address to which the routine returns when the end of the output
    service queue is found without satisfying the request. EOD= is valid only if you
    specify TYPE=SCHEDULE, in which case EOD= is required.

TYPE=
    Specifies the action required.
    SCHEDULE
        Specifies a request for those parameters that prevent a dynamic writer from
        scheduling.
    GET
        Specifies that a dynamic writer is to be queued for scheduling or a hot
        writer is to be scheduled for posting.
    PUT
        Specifies that any dynamic writers that are queued are to be started.
    REL
        Specifies that all nonprint OSEs for the job are to be canceled.
    DEL
        Specifies that all OSEs for the job are to be canceled.

OSE=
    Specifies the address of an output service element to be used for the call. This
    parameter is required and valid only if TYPE=GET or TYPE=SCHEDULE is
    specified.

NORMAL=
    Specifies the address to which the routine returns when the request has been
    satisfied.

Linkage
    This macro uses ASAVE linkage.

Restrictions
    Do not use this macro in any routine that might run in a C/I FSS address space or
    under the JES3 auxiliary task.

IATXOSUP (Set or Reset RQOSESUP)
    If you use an installation exit to access the output service element (OSE) chain
    from spool, you need to code the IATXOSUP macro in your installation exit.

    When reading an OSE from spool, the OSE FDB pointer contained in the job’s RQ
    is used as the spool record address. By using the FDB, your routine causes JES3
    to update control blocks associated with the OSE, and the spool record address
    (M.R.) is changed to a buffer address.

    The job’s RESQUEUE (RQ) contains an OSE update flag (bit RQOSESUP) which
    is used as a locking mechanism for the job’s OSE chain. JES3 uses this bit for
    exclusive access to the OSE chain on spool. Any routine reading or writing an OSE
    from or to spool must synchronize OSE usage with other routines.

    Note: Do not use the IATXOSUP macro if the installation exit processes the input
    OSE and does not initiate I/O for the spool OSE.
Use the IATXOSUP macro to gain exclusive use of the OSE chain when I/O is to be initiated as follows:

1. Test the OSE update flag (bit RQOSESUP) to ensure that no other routine has access to the same OSE.
2. If the bit is on (set to one), another routine has access to the OSE. Your routine should issue an AWAIT TYPE=OFF macro to wait for the other routine to reset (to zero) the OSE update flag.
3. If the bit is off (set to zero), issue an IATXOSUP TYPE=SET to turn it on before accessing the OSE, then proceed with updating the OSE.
4. When finished, turn the flag off by issuing IATXOSUP TYPE=RESET. Now, another routine can access the OSE.

The IATXOSUP macro sets and resets the RQOSESUP bit and places a DSP number into the RQ. This DSP number identifies the DSP that invokes the installation exit. IATXOSUP is not a serialization technique like the AENQ macro, therefore you must follow the explicit directions previously listed.

Do not change RQOSESUP without knowledge of the environment under which the exit is called because you can permanently AWAIT JES3 functions. You can avoid a permanent AWAIT if you:

- Check to see if RQOSESUP has been turned on (set to one) by any code executing under this DSP.
- After issuing IATXOSUP TYPE=SET and before issuing IATXOSUP TYPE=RESET, do not invoke any code that changes the status of RQOSESUP.

The following table describes the footprints in IATXOSUP and which module sets them.

<table>
<thead>
<tr>
<th>Footprint</th>
<th>Set in Module</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>OSUP01</td>
<td>IATDJOT</td>
<td>Checks if the job is in a phase of output service where it can be dumped before issuing a JESREAD</td>
</tr>
<tr>
<td>OSUP10</td>
<td>IATIQOI</td>
<td>Searches the OSE chain looking for data sets which match the output service inquiry command selection criteria</td>
</tr>
<tr>
<td>OSUP20</td>
<td>IATMOOI</td>
<td>Searches the OSE chain looking for data sets which match the output service modify command selection criteria</td>
</tr>
<tr>
<td>OSUP30</td>
<td>IATOSBM</td>
<td>Job selection routine checks the OSE chain for Bulk Data Transfer (BDT) output</td>
</tr>
<tr>
<td>OSUP40</td>
<td>IATOSDR</td>
<td>Spin-off output processing routine</td>
</tr>
<tr>
<td>OSUP41</td>
<td>IATOSDR</td>
<td>Normal output service processing routine</td>
</tr>
<tr>
<td>OSUP50</td>
<td>IATOSPC</td>
<td>SYSOUT request processing</td>
</tr>
<tr>
<td>OSUP51</td>
<td>IATOSPC</td>
<td>Job request processing</td>
</tr>
<tr>
<td>OSUP52</td>
<td>IATOSPC</td>
<td>Job scheduling routine</td>
</tr>
<tr>
<td>OSUP60</td>
<td>IATOSSC</td>
<td>Release/delete OSE routine</td>
</tr>
<tr>
<td>OSUP61</td>
<td>IATOSSC</td>
<td>Output service element (OSE)Bulk Data Transfer (BDT) processing</td>
</tr>
<tr>
<td>OSUP70</td>
<td>IATOSWS</td>
<td>Get output service element (OSE) routine</td>
</tr>
<tr>
<td>OSUP71</td>
<td>IATOSWS</td>
<td>Output service element (OSE) schedule routine</td>
</tr>
<tr>
<td>OSUP72</td>
<td>IATOSWS</td>
<td>Delete / release output service element (OSE) routing</td>
</tr>
<tr>
<td>OSUP73</td>
<td>IATOSWS</td>
<td>PUT output service element (OSE) routine</td>
</tr>
<tr>
<td>OSUP74</td>
<td>IATOSWS</td>
<td>Remove resident job queue (RSQ) from the writer queue</td>
</tr>
<tr>
<td>OSUP80</td>
<td>IATPURG</td>
<td>Purge any existing output service elements (OSEs)</td>
</tr>
<tr>
<td>OSUP90</td>
<td>IATUTCB</td>
<td>Output service element (OSE) print routine</td>
</tr>
</tbody>
</table>
Footprint | Set in Module | Explanation
---|---|---
OSUP100 | IATNTRS | Copies the JMR from the APPC networking stream to the target job’s JMR
OSUP110 | IATOSGR | Reads the OSE chain to produce a dump of the OSEs for a DM754 abend

**Syntax**

```
IATXOSUP

 TYPE=SET,INDEX=number,RESET,RQ=(reg)
```

**Parameters**

**TYPE=**

Specifies the type of function requested.

**SET**

Indicates to other routines that the OSE is being accessed. When RQOSESUP is set (to one), the DSP number and index are saved in the RESQUEUE.

**RESET**

Indicates to other routines that the OSE is available. When RQOSESUP is reset (to zero), the DSP number and index are cleared from the RESQUEUE.

**INDEX=**

A decimal number indicating where the OSE update flag is being set. For installation-supplied modules, you should use a unique number in the range 3000 through 4095. Mapping macro IATYOSUP maps the index values.

**RQ=**

Indicates the register that contains the address of the RESQUEUE. This parameter is required when the address of the RESQUEUE is not included in the module and defined using a USING instruction. If this parameter is not specified, the macro uses the current RESQUEUE.

You may not specify register 1.

**Linkage**

None. This macro expands inline.

**Restrictions**

None.

**IATXOSWS (Access Output Service Elements)**

The IATXOSWS macro accesses output service elements (OSEs) for SYSOUT data sets in the output service queues. This macro is for output service writer requests.
IATXOSWS

Syntax

```
IATXOSWS PARM=,NORMAL=address
```  

Parameters

**PARM=**
Specifies the address of a work area (mapped by macro IATYWSP).

**EOD=**
Specifies the address to which the routine returns when the end of the output service queue is found without satisfying the request. EOD= is valid only when you also specify TYPE=GET or TYPE=SCHEDULE. In both of these cases, EOD= is required.

**NORMAL=**
Specifies the address to which the routine returns when the request has been satisfied.

**TYPE=**
Specifies the action required.

- **SCHEDULE**
  Specifies a request for a job and OSE for writer output.

- **GET**
  Specifies a request for access to the next scheduled OSE.

- **PUT**
  Specifies that any remaining scheduled OSEs for the writer are to be made available to any writer.

- **REL**
  Specifies that all completed OSEs and data sets for output to the same device type as the caller are to be reset for transmission.

- **DEL**
  Specifies that all OSEs and data sets for output to the same device type as the caller are to be marked complete.

Linkage

This macro uses ASAVE linkage.

Restrictions

Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task.
IATXPDQ (Call the Pending Data Set Queue Manager)

The IATXPDQ macro calls the pending data set queue manager (module IATOSFP) to perform a variety of functions on behalf of an output writer functional subsystem (FSS). Each function is described below, with the parameter that invokes it.

Note: The IATYPDQ, IATYOSE, IATYWTR, and IATYWSP data areas are used in conjunction with this macro.

Syntax

```
IATXPDQ

label

TYPE=Parm,

NORMAL=address,

ERROR=address
```

Parm:

```
Parm
```

Parm:

```
PDQJBSEL,
PDQOSSEL,
PDQOSSEND,
PDQINQ,PARM=((Rx),(Ry))
PDQJLOG
PPQ=
DPQCLEAR
PDQJBSNT
PPQ=
PDQWOSUP,BUSY=address,EOD=address,PPQ=
PDQWOSCN
SUP=address,RQ=address,OSEIDPTR=address
PDQERR
,PARM=errcode
PDQSYNCH
PPQ=
PDQDSDSP
PPQ=
PDQDSUPD
PPQ=
PDQWOSRD
PPQ=
PDQWOSWR
PPQ=
PDQJBDSP,PARM=label
PDQCLEAR
PDQRECR
SUBCALL,PARM=label
PDQGET
SAVE=PPQ=
PDQGETST
PDQDEL
PDQDEQ
PDQLOC
PDQWSIN
PDQREST
PDQDSR
PDQSUPCK
```
IATXPDQ

**parameters**

**type=**

Specifies which routine of the pending data set queue manager is requested on behalf of an output writer FSS.

**PDQJBSEL**

Specifies a request for the job selection routine. The PDQ job selection routine creates a pending data set queue (PDQ) entry for the new job the output writer FSS has selected for printing.

**PDQOSSEL**

Specifies a request for the output scheduling element (OSE) selection routine. The PDQ OSE selection routine creates a PDQ entry for a new OSE that the output writer FSS has selected for printing.

**PDQDSSEL**

Specifies a request for the data set selection routine. The data set selection routine creates or updates a PDQ entry for a data set that is being sent to the output writer FSS for printing. The data set is also marked as pending in the OSE.

**PDQDSEND**

Specifies a request for the data set end routine. The data set end routine records the completion or reschedule of a data set’s printing in the PDQ and OSE. If possible, the routine deletes the data set’s PDQ entry.

**PDQINQ**

Specifies a request for the inquiry routine. The inquiry routine returns, one at a time in register 1, the RESQUEUE addresses of all the jobs with data sets that have been sent to the output writer FSS but have not yet been completely printed.

**PDQJBLOG**

Specifies a request for the job logon routine. The job logon routine issues message IAT7001 when the first data set of a job reaches the printer’s transfer station.
PDQJBSNT
Specifies a request for the job sent routine. The job sent routine records the completion of a job when the last data set of a job has been sent to the output writer FSS.

PDQWOSUP
Specifies a request for the WOSE update routine. The WOSE update routine changes the status of all data sets in a WOSE. The status to which the data sets are to be changed is passed by the issuer of this macro using fields WTRISET and WTRIRSET in the writer data CSECT IATODFD. The WOSE update routine returns the status of the OSE to the caller by returning to different offsets following the point of invocation.

PDQWOSCN
Specifies a request for the WOSE cancel routine. The WOSE cancel routine marks all PDQ entries for data sets belonging to the specified WOSE for cancellation following their release by the output writer FSS. When the last data set is released by the writer, the entire WOSE is marked complete and copied to the disk OSE.

PDQERR
Specifies a request for the PDQ error routine. The PDQ error routine issues the FAILDSP macro with dump code DM655 and an error reason code determined by the PARM parameter specification.

PDQSYNCH
Specifies a request for the transfer station synchronization routine. The transfer station synchronization routine marks all data sets newer than the one at the printer’s transfer station for rescheduling following their release by the output writer FSS. The routine also restores the output writer DSP to the state it was in when the data set at the transfer station was sent to the output writer FSS.

PDQDSDSP
Specifies a request for the data set disposition routine. The data set disposition routine marks a data set’s PDQ entry for cancellation or rescheduling following its release by the output writer FSS.

PDQDSUPD
Specifies a request for the data set update routine. The data set update routine changes the status of a specific data set in the work OSE (WOSE). The status to which the data set is to be changed is passed by the issuer of this macro using fields WTRISET and WTRIRSET in the writer data CSECT IATODFD.

PDQWOSRD
Specifies a request for the WOSE read routine. The WOSE read routine reads a WOSE from the spool into main storage and returns its buffer address in register 1.

PDQWOSWR
Specifies a request for the WOSE write routine. The WOSE write routine checkpoints a WOSE buffer to the spool.

PDQJBDSP
Specifies a request for the job disposition routine. The job disposition routine marks all PDQ entries for data sets belonging to a job for cancellation or rescheduling following their release by the output writer FSS.

PDQCLEAR
Specifies a request for the PDQ clear routine. The PDQ clear routine
removes all entries from the PDQ and reschedules them when the output writer FSS has ended without releasing data sets.

**PDQRECR**
Specifies a request for the PDQ recreate routine. The PDQ recreate routine rebuilds the PDQ chain from the output service restart chain (IATYOSR) after a hot start.

**SUBCALL**
Specifies a request for an internal subroutine within module IATOSFP. The TYPE=SUBCALL parameter can only be used by the pending data set queue manager.

**PARM=**

**label**
Specifies a field containing the address of the RESQUEUE of the job to be canceled. This option is valid only with the TYPE=PDQJBDSP parameter.

**((Rx),(Ry))**
Specifies, in (Rx), the register containing the address of the PDQ that was returned in register 0 on the prior IATXPDPQ macro call. On the first IATXPDPQ macro call by the calling routine, the (Rx) register should be set to zero. (Ry) specifies the register containing the address of the SUPUNITTS table entry for the device supported by the output writer FSS. This option is valid only with the TYPE=PDQINQ parameter.

**errcode**
Specifies an equated name for the error code to be passed to the error routine. The error codes are defined in module IATOSFP. This subparameter is valid only with the PDQERR parameter.

**PDQGET**
Requests that a usable PDQ entry be obtained. The address of the PDQ entry is returned in register 1.

**PDQGETST**
Requests that a new cell be obtained from the PDQ cell pool.

**PDQDEL**
Requests the deletion of a PDQ entry, if possible.

**PDQDEQ**
Requests that a PDQ entry be removed from the queue and the storage returned to the cell pool.

**PDQLOC**
Requests the location of the PDQ entry whose data set ID matches the ID in field WTRFDSID in the writer data CSECT IATODFD.

**PDQWSPIN**
Requests initialization of the PDQ manager's private writer selection parameter (WSP) for use in issuing the IATXOSWS macro.

**PDQREST**
Requests the restoration of the writer data CSECT IATODFD to the same state as when the data set in the PDQ entry was first passed to the output writer FSS.

**PDQDSPRG**
Requests that a data set with its own TAT (track allocation table) be purged.
**PDQSUPCK**

Requests that a SUPUNIT entry be updated to reflect the setup of a given PDQ representing a data set.

**SAVE=**

**YES**

Specifies that this macro should use ASAVE linkage.

**label**

Specifies a label where the value in register 10, which is the base register for all routines in module IATOSFD, is to be saved.

**EOD=**

With the TYPE=PDQWOSUP parameter, specifies the address to which control returns when all data sets in the WOSE are complete or held.

**BUSY=**

With the TYPE=PDQWOSUP parameter, BUSY= specifies the address to which control returns when no more work can be scheduled from the WOSE, but at least one data set is not yet completely printed.

**PDQ=**

Specifies the address of a PDQ entry. This parameter can only be specified with the PDQWOSRD, PDQWOSWR, PDQWOSUP, PDQDSUPD, PDQDSEND, PDQSYNCH, PDQJBLOG, PDQJBSNT, PDQSUPCK, and SUBCALL options of the TYPE parameter.

**SUP=**

Specifies the address of the SUPUNITs associated with the writer processing the WOSE. If no address is specified, register 0 contains the address.

**RQ=**

Specifies the address of the RESQUEUE the given WOSE belongs to. If no address is specified, register 1 contains the address.

**OSEIDPTR=**

Specifies the address of a pointer to the six-byte WOSE identifier, which consists of its sequence number (four bytes) and offset into the disk OSE buffer (two bytes). If no address is specified, the pointer is found in register 2.

**NORMAL=**

Specifies the address to which control returns when the requested function has completed normally.

**ERROR=**

Specifies the address to which control returns when an error occurs while the requested function is processing.

**Linkage**

This macro uses ASAVE linkage, unless SAVE=label is specified with the TYPE=SUBCALL parameter.

**Restrictions**

Do not use this macro in any routine that might run under the JES3 auxiliary task.

---

**IATXPGXM (Page-Fix or Page-Free in JES3AUX)**

The IATXPGXM macro page-fixes or page-frees an area in the JES3 auxiliary address space (JES3AUX).
The IATYSVT and IATYPCD mapping macros must be included in the calling routine.

Syntax

\[
\text{IATXPGXM} \quad \text{label} \quad \text{BEGADD} = \text{address} \quad \text{ENDADD} = \text{address} \quad \text{OP} = \text{FIX}\text{FREE}
\]

Parameters

**BEGADD=**
- Specifies the starting address of the area to be fixed.

**ENDADD=**
- Specifies the address of the last byte to be fixed or freed.

**OP=**
- **FIX**
  - Specifies page-fix. The real address of the page-fixed area will be returned in register 1.
- **FREE**
  - Specifies page-free.

Return Codes

Following are the return codes returned in register 15:

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Normal completion</td>
</tr>
<tr>
<td>Other</td>
<td>Abnormal completion</td>
</tr>
</tbody>
</table>

Linkage

This macro issues a PC 0 (R2) instruction for linkage. The IATXPGXM macro does not save the registers for the calling routine. Therefore, a routine must save its own registers before issuing this macro. The routine must also provide in register 13 the address of a 72-byte save area in CSA. This save area is for use by the IATXPGXM service routine (IATDMXM). Furthermore, the routine is responsible for testing the return code and branching to any code that handles normal or error returns. The routine must restore registers before performing this return code handling.

Restrictions

Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task.

IATXPHEX (Internal Data Translator)

The IATXPHEX macro translates internal data into printable hexadecimal values.
Syntax

```
label  IATXPHEX—to address from address (reg)

,WORK= work area label TRTBL= table

FCTWORK TVTHXCHR
```

Parameters

**label**
Optional label associated with the first instruction generated by the IATXPHEX macro.

**to address**
Specifies the address of the area in which the converted data is to be placed.

**from address (reg)**
Specifies the address of the data to be converted. If register notation is used, the register must contain the data to be converted.

**WORK=**
Specifies the work area used to store data before it is placed in the “TO” area.

**TRTBL=**
Specifies the translation table used by the TR instruction to convert data.

**IATXPJ3 (Post JES3)**

The IATXPJ3 macro can be issued from any address space to post either the JES3 master event control block (ECB), which is used by the JES3 primary task, or the JES3 auxiliary task ECB, or both. The specified ECB(s) is posted synchronously to the call if “fast posting” (that is, compare and swap) is possible. Otherwise, a service request block (SRB) is scheduled to branch enter the MVS post service asynchronously.

If this macro is issued from a routine in a C/I FSS address space, the JES3 global address space is posted.

Syntax

```
label  IATXPJ3 TYPE= JPOST APOST ADDR= TVT SSVT MPC NONE
```

Parameters

**TYPE=**
Specifies the type of post that is to be done.

**JPOST**
Specifies that the JES3 master ECB is to be posted.

**APOST**
Specifies that the auxiliary task ECB is to be posted.
IATXPJ3

JAPOST
Specifies that both the JES3 master ECB and the auxiliary task ECB are to be posted.

ADDR=
Specifies the way that the address of the JES3 post routine is to be obtained.

 TVT
Specifies that the address is to be obtained from the transfer vector table (TVT). The TVT DSECT (IATYTVT) must be defined using a USING instruction.

 SSVT
Specifies that the address is to be obtained from the subsystem vector table (SSVT). The SSVT DSECT (IATYSVT) must be defined using a USING instruction.

 MPC
Specifies that the address is to be obtained using the main processor control table (MPC). The MPC DSECT (IATYMP) and the SSVT DSECT (IATYSVT) must be included in the module and the MPC must be defined using a USING instruction.

 NONE
Specifies that the address is to be obtained using the CVT. If ADDR=NONE, the following must be defined:
• The CVT DSECT with DSECT=YES specified on the CVT macro call
• The JESCT DSECT (IEFJESCT)
• The SSCT DSECT (IEFJSCVT)
• The SSVT DSECT (IATYSVT)

Linkage
This macro issues a BALR or BAL instruction for linkage. Registers 14 and 15 are used to call the global JES3 post routine and all other registers remain unchanged.

Restrictions
The issuer of IATXPJ3 must be in key 0, supervisor state. Do not specify ADDR=TVT with this macro in any routine that might run in a C/I FSS address space.

IATXPOSE (Update an Output Service Summary Entry)
The IATXPOSE macro updates an existing entry in the output service element (OSE) scheduling matrix. Based on the status of the parameter OSE, output service summary entries can be added, deleted, or updated.

Syntax

```
IATXPOSE

label
OSE=
WSP=
(R1)
address
RQ=
(reg)
address
NORMAL=
address
BUFNUM=
address
OPTION=CLEANOS
```

Parameters

OSE=
Specifies the address of a parameter OSE reflecting the new status for the OSE to be changed. If coded, the writer selection parameter WSP is not allowed. The high order bit of R1 should be "off."

WSP=
Specifies the address of the writer scheduling parameter (WSP) area. This is used whenever all OSE put requests have been made for a particular job. This request causes a summarization of all output service summary (OSS) entries to be posted back to the master OSE (MOSE). If this parameter is coded, the OSE parameter is not allowed. The high order bit of R1 should be "on."

RQ=
Specifies the address of the resident job queue (RESQUEUE) entry for the job related to the OSE or WSP.

NORMAL=
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

BUFNUM=
Specifies the address of the sequence number (OSECNT4) of the OSE buffer that contains the OSE variable entry. This parameter is valid only if you have coded OSE=.

OPTION=CLEANOSS
Indicates that all OSS's belonging to the jobs that have not been queued (using IATXGOSE QUE=YES) should be removed. If this option is coded, neither the OSE parameter nor the WSP parameter is allowed.

Linkage

This macro uses ASAVE linkage.

Restrictions

Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task.

IATXPOST (Post a Function Using the COMPARE AND SWAP Instruction)

The IATXPOST macro performs either or both of the following functions:
- Post an event completion flag (ECF) using the COMPARE AND SWAP instruction.
- Post either the JES3 master event control block (ECB) used by the JES3 primary task, or the JES3 auxiliary task ECB, or both ECBs.

Syntax

```
IATXPOST label,flag,flagmask,ON,OFF Other Operands
```

Chapter 2. JES3 Macros 395
IATXPOST

Parameters

flag
Specifies the address of the ECF to be posted.

flagmask
Specifies the mask name of the ECF bit to be posted.

ON
Specifies that the bit is to be posted on.

OFF
Specifies that the bit is to be posted off.

JPOST
Specifies that the JES3 master ECB is to be posted.

TYPE=
  JPOST
  Specifies that the JES3 master ECB is to be posted.
  APOST
  Specifies that the JES3 auxiliary task ECB is to be posted.
  JAPOST
  Specifies that both the JES3 master ECB and auxiliary task ECB are to be posted.

REF=
Specifies the name of the DSECT which contains the flag.

Linkage

In the case of the ECF post function, the IATXPOST macro uses registers 0, 1, and 2 to do the COMPARE AND SWAP. In the case of the ECB post function, registers 2, 14, and 15 are used to invoke the IATXPF3 post service. All other registers remain unchanged.

Restrictions

When IATXPOST is used to post an ECB, the routine which invokes the macro must be in supervisor state and must have addressability to the TVT. Do not use this macro in any routine that might run in a C/I FSS address space.

IATXPRES (Set Bytes in a Specified Area)

The IATXPRES macro sets each byte of a specified area to a specified value.

Syntax
Parameters

area address (reg)
  Specifies the address of the area being set.

hexexp char
  Specifies the value to which to set the area.

number
  Specifies the length of the area.

IATXPRMD (Define Operation for Process Mode Table)

The IATXPRMD macro enables you to define the operation of the JES3 process mode function.

Syntax

```
IATXPRMD TYPE=

INIT
  REJECT= Options1
  ADD Options1
  INQ NAME= SUPINIT= INQ NAME=
  VALIDATE Options1

Options1:

,NAME = (R1),LENGTH = (R0),SUPUNIT = (R2)
  (reg) address, (reg) address, (reg) address

NAME=:

,NAME = (R1)
  (reg) address

SUPINIT=:

,SUPUNIT = (R2)
  (reg) address

REJECT=:

,REJECT = address
```

Chapter 2. JES3 Macros
Parameters

**TYPE=**
Defines the operation to be performed with the process mode table. This parameter is *required*.

**INIT**
Defines the process mode tables and initializes the data structures. Used only in JES3 initialization services.

Control returns to the address specified on the REJECT= parameter if the table was unable to be initialized because of insufficient storage for the AGETMAIN request. Otherwise, control returns to the caller plus 4 bytes if the table is initialized correctly.

**ADD**
Adds the supplied process mode (using NAME=) to the supplied SUPUNIT. Additionally, if this parameter is used during initialization, it will add either to the PM or ALTPM lists depending on the PLACE= parameter, if supplied.

Control returns to the caller as follows:
- Returns to the caller plus 0 bytes if there are too many additions for this SUPUNIT.
- Returns to the caller plus 4 bytes if there are too many additions for the complex.
- Returns to the caller plus 8 bytes if the device already supports the given process mode.
- Returns to the caller plus 12 bytes if the addition executes correctly.

**DEL**
Returns to the caller plus 4 bytes if the delete action was successful for the provided SUPUNIT and the supplied process mode name (see NAME=). Otherwise, control is passed to the address specified by the REJECT= parameter.

**INQ**
Used in inquiry or other modules to obtain a list of process modes for the particular device. On original entry, register 1 must be set to 0. Length is not used or returned; the process mode returned will be 8 bytes long, filled on the right with blanks. The calling module must not modify the returned process mode. Instead it should only use the pointer returned in register 1 to move the characters into its own buffer.

TYPE=INQ returns to the address specified on EOD= if no more entries are to be returned.

TYPE=INQ returns to the caller plus 4 bytes if the entry was returned, and may be reentered to obtain the next process mode name after the caller finishes processing the returned entry. Register 2 must be the same upon subsequent reentries to the service routine as it was set on exiting from the prior call.

**VALIDATE**
Determines if the supplied process mode (from NAME=) is valid in a syntactical sense.

This parameter returns to the caller if the process mode is valid.

The REJECT= parameter is *required* if you specify TYPE=VALIDATE. If the process mode fails validation because of an incorrect length or incorrect characters, control will be passed to the address specified on REJECT=.
LENGTH=
Defines the length of the process mode for TYPE=ADD, TYPE=DEL, or TYPE=VALIDATE operations. If not specified for either of these three operations, the default length is 8. If a value other than 1 to 8 is specified for TYPE=ADD or TYPE=DEL operations, the default length of 8 will be used.

An incorrect length for TYPE=VALIDATE causes the validation routine to take the REJECT= path, indicating the process mode name is not syntactically valid.

NAME=
Defines the particular process mode name being processed. NAME= is required for all TYPE= parameters except for TYPE=INIT.

SUPUNIT=
Defines the SUPUNIT in question for the process mode operation. SUPUNIT= is required for all TYPE= parameters except for TYPE=INIT and TYPE=VALIDATE.

PLACE=
Defines whether a PM or ALTPM process mode is being added to the process mode list and should only be used in JES3 initialization modules. PLACE= is optional and is only allowed if you also specify TYPE=ADD.

REJECT=
Specifies the address to return to if a problem occurs as follows:
For TYPE=INIT requests, REJECT= indicates that the service routine rejected the initialization attempt because there was insufficient storage for the process mode table using the AGETMAIN macro. Control is returned to the address specified on REJECT=.
For TYPE=DEL requests, REJECT= indicates that the delete request is rejected because the given process mode name cannot be found for the particular SUPUNIT that was specified. Control is returned to the address specified on REJECT=.
For TYPE=VALIDATE requests, REJECT= indicates that the name presented to the service routine failed syntax validation. Control is returned to the address specified on REJECT=.
The REJECT= parameter is valid (and must be specified) only for TYPE=INIT, TYPE=DEL, and TYPE=VALIDATE requests.

EOD=
Defines the address for control to return to when the inquiry routine has determined that the particular SUPUNIT had no more process modes to be scheduled to.
EOD= is valid (and must be specified) for TYPE=INQUIRY requests.

Linkage
This macro uses ASAVE/ACALL linkage to enter the service routine in module IATOSSC.

Restrictions
This macro uses registers 0, 1, and 2 to pass parameters to and from the service routine.
The IATXPRT macro provides the facility for printing storage areas in hexadecimal format.

Syntax

```
IATXPRT, SIZE=number, DATA=address, TYPE=

OPEN

FDB=(R0) address

JTAT=

TIME

TITLE

RESQ=

CBOPEN

WORK=(R1) address

FDB=(R1) address

WORK=(R1) address

RESQ=(R1) address

ID1=address

ID2=address

ID3=address

ID4=address

IDC1=address

IDC2=address

IDC3=address

IDC4=address
```

JTAT=:

```
JTAT=(R0) address
```

FDB=:

```
FDB=(R1) address
```

WORK=:

```
WORK=(R1) address
```
Parameters

**WORK=**
Specifies the address of a fullword that contains the work area address. If coded, then TYPE=OPEN or TYPE=CBOOPEN may not be coded.

**SIZE=**

- number
  - Specifies the length of the data to be printed. If you specify TYPE=TITLE, the length must not be more than 122. If you specify TYPE=ATITLE, the length must not be more than 98. Otherwise, the maximum value is 32,767.

- address
  - Specifies the address of a halfword containing the length. This subparameter must be a symbol.

**DATA=**
Specifies the address of the data to be printed.

**TYPE=**

- OPEN
  - Must be issued before any printing can be accomplished. If coded, then the WORK parameter may not be coded; however, the RESQ or FDB parameter must be coded.

- CBOPEN
  - Reserved for use by module IATUTCMB.

- CLOSE
  - Must be issued to close out the printing process.

- TIME
  - Causes a time stamp to be placed on the first line of the print. This option is not valid if the ID1 or IDC1 parameter is specified.

- TITLE
  - Causes the data area to be moved as a title.

- ATITLE
  - Causes the data area to be moved as a title, centered on the page, and surrounded by lines of asterisks.

- EQU
  - Generates reserved equates.

- LINE
  - Causes a line of data to be written to the output data set without any special formatting.

**RESQ=**
Specifies the address of the resident job queue (RESQUEUE) entry to be used when creating the data set DEBUG. This data set will contain the print of the caller’s requests and will be set up with a print class defined at initialization. If coded, then the FDB parameter may not be coded.

**FDB=**
Specifies the address of the multirecord FDB to be used for receiving the print output. The file may be open or not; the entry status will be maintained when TYPE=CLOSE is issued. If this FDB is contained in a job data set table (JDS), it must be ‘JDSHELD’ by the caller. Print format will be ASA and will be set by a print header record during TYPE=OPEN. If coded, then the RESQ parameter may not be coded.
IATXPRT

JTAT=
Job track allocation table (TAT) FDB address. This option is only valid if TYPE=OPEN and the FDB parameter are specified.

DISP=
ADISP
  Specifies that an absolute displacement for each line will be printed.
CDISP
  Specifies that the address for each line will be printed.

ID1-ID4=
  Specifies the address of an 8-byte character string that will be printed along with lines 1-4, respectively. ID1 and IDC1 are mutually exclusive, ID2 and IDC2 are mutually exclusive.

IDC1-IDC4=
  Specifies the address of a fullword which will be converted to hexadecimal and printed along with lines 1-4, respectively. ID1 and IDC1 are mutually exclusive, ID2 and IDC2 are mutually exclusive.

NORMAL=
  This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

Linkage
This macro uses ASAVE linkage.

Restrictions
The IATYXPR mapping macro must be coded in the calling module. Do not use this macro in any routine that might run under the JES3 auxiliary task. This macro uses register 2.

IATXPSCL (Request Presentation Services)
Function management (inbound and outbound) uses the IATXPSCL macro to request a service from presentation services. Depending on the original form of the data being sent from or to the work station, the request will result in either:
- The compression or decompression of the data.
- The translation of the data from ASCII to EBCDIC or from EBCDIC to ASCII.

Whenever registers are used, the registers must be within the range from 2 to 9.

Syntax
```
/SM590000/SM590000
IATXPSCL label LUCB=(reg) address, FLOW=IN OUT
```
```
,ERROR=(reg) address, NORMAL=(reg) address
```

Parameters
LUCB=
  Specifies the address of the logical unit control block (LCB) representing the
session on which data is sent to and received from the work station logical unit (WS LU). The LCB contains a work area that is used as a parameter buffer.

**FLOW=**
Specifies whether the flow is inbound (IN) or outbound (OUT).

**ERROR=**
This parameter specifies the address to which the macro processing routine returns if an error occurs. When control is passed to the address, register 0 contains a dump code and register 15 contains a return code, unless otherwise specified. The return codes are described below.

If an error occurs and ERROR was not specified, the dynamic support program (DSP) will be ended using the FAILDSP macro.

**NORMAL=**
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

### Return Codes
Following are the error return codes returned in register 15:

<table>
<thead>
<tr>
<th>Return Code In Hex</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>04</td>
<td>Data conversion error occurred</td>
</tr>
<tr>
<td>08</td>
<td>Incorrect JESTAE control block (IATYSCB)</td>
</tr>
</tbody>
</table>

**Linkage**
This macro uses ASAVE linkage.

**Restrictions**
The module issuing this macro must run under the control of a DSP. Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task.

**IATXPTCH (Generate Patch Area)**
The IATXPTCH macro generates a patch area in a module. The patch area is preceded by the fullword aligned character constant PTCH and is filled with halfwords indicating the current base register and displacement. The area is either 2% of the CSECT's size or 52 bytes, whichever is larger.

**Note:** Addressability to the patch area must be available from all areas of the program, including the patch area itself, except in the case noted in the PSTRT parameter description below.

A statement label may be specified. If no label is coded, the name PTCHxxxx is generated, where xxxx is the fourth through seventh characters of the CSECT name.
IATXPTCH

Syntax

```
IATXPTCH label LT, PSTRT=patch_area_label
```

Parameters

- **LT** Specifies that a LTORG statement is to be generated preceding the patch area.
- **PSTRT** Specifies that the patch area size calculation is to be made from the supplied label rather than from the start of the CSECT. This would normally be used when a CSECT consists of multiple sections of code with separate entry points or base register allocations, and a common patch area would not be within the range of a USING statement.

Linkage

This macro does not generate executable code.

Restrictions

None.

IATXPTCK (Checkpoint the Partition TAT)

The IATXPTCK macro checkpoints the partition track allocation table (partition TAT) or invalidates a checkpoint record for the partition TAT. The IATXPTCK service routine uses only MVS services to checkpoint the partition TATs.

Register 1 is used as a parameter register.

Syntax

```
IATXPTCK label TYPE=VALID
```

Parameters

- **TYPE** Specifies the type of PTAT checkpoint operation to be performed.
  - **VALID** Specifies that a valid copy of the current PTAT is to be written to the checkpoint data set(s).
  - **INVALID** Specifies that the current PTAT checkpoint record is to be marked incorrect in the checkpoint data set(s).

Linkage

This macro issues the IATZCALL macro without using ASAVE linkage. The caller’s registers are saved by the IATXPTCK service routine.
Restrictions

Only JES3 recovery processing in the JES3 global address space may use this macro.

IATXRABC (Create Record Allocation Block)

The IATXRABC macro allocates track groups to create a record allocation block (RAB).

The issuing routine must specify the number of track groups to be allocated in field RABTRKG1 of the RAB specified by the RAB parameter. The IATXRABC macro returns the number of track groups actually allocated in field RABTRKG2 of the RAB.

If no track groups are available in the spool partition identified by the job or data set TAT (specified indirectly using the FDB parameter), the IATXRABC service routine issues an AWAIT TYPE=OFF macro to wait until track groups become available. The service routine also issues an AWAIT TYPE=OFF macro if a minimal spool space condition exists for the identified spool partition. If enough track groups are available in the spool partition to obtain at least one, but not all, of the requested track groups, those track groups that were obtained will be returned to the caller.

Syntax

```
>> label IATXRABC RAB=(R1) address, FDB=(R0) address
>> ,ERROR=address ,NORMAL=address
```

Parameters

RAB=
Specifies the address of an area containing the RAB to be allocated track groups. The routine issuing this macro must have previously initialized the RAB.

FDB=
Specifies the address of the FDB for the job or data set track allocation table (TAT) where the track group allocations are to be recorded.

ERROR=
Specifies the address to which control is returned if an error is encountered while processing the request.

NORMAL=
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

Return Codes
Following are the return codes and dump codes returned in registers 15 and 0, respectively:
IATXRABC

<table>
<thead>
<tr>
<th>Return Code in Hex</th>
<th>Dump Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>08</td>
<td>721</td>
<td>Spool I/O error while reading the job or data set TAT for allocation</td>
</tr>
<tr>
<td>0C</td>
<td></td>
<td>ATRACK error</td>
</tr>
<tr>
<td>10</td>
<td>721</td>
<td>Spool I/O error while checkpointing the job TAT</td>
</tr>
<tr>
<td>14</td>
<td></td>
<td>ARELEASE error</td>
</tr>
<tr>
<td>24</td>
<td></td>
<td>Spool extent number too low</td>
</tr>
<tr>
<td>28</td>
<td></td>
<td>Spool extent not found</td>
</tr>
<tr>
<td>2C</td>
<td></td>
<td>Spool record count too large</td>
</tr>
<tr>
<td>30</td>
<td></td>
<td>Spool record count too low</td>
</tr>
<tr>
<td>34</td>
<td></td>
<td>Spool record number too large</td>
</tr>
<tr>
<td>38</td>
<td></td>
<td>Spool record number too low</td>
</tr>
<tr>
<td>3C</td>
<td></td>
<td>Error returned by the Slot Location routine</td>
</tr>
<tr>
<td>40</td>
<td></td>
<td>The track group is not allocated to the caller</td>
</tr>
</tbody>
</table>

Linkage

This macro uses ASAVE linkage.

Restrictions

Do not use this macro in any routine that might run under the JES3 auxiliary task.

IATXRABD (Purge Record Allocation Block)

The IATXRABD macro purges a record allocation block (RAB) and frees the RAB’s track groups for allocation. The RAB to be purged must have been created by issuing the IATXRABC macro.

Syntax

```
//SM590000 SM590000
IATXRABD label(R1)RAB=(R1)address,FDB=(R0)address
//SM590000 SM590000

,ERROR=address
''NORMAL=address
```

Parameters

**RAB=**

Specifies the address of the RAB to be purged. The RAB must have been created using the IATXRABC macro.

**FDB=**

Specifies the address of the FDB for the job or data set track allocation table (TAT) where the track group allocations were recorded.

**ERROR=**

Specifies the address to which control returns if an error is encountered while processing the request.

**NORMAL=**

This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.
Return Codes
Following are the return codes returned in register 15. A DM753 failure will occur for all of these errors.

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Hex</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>04</td>
<td>04</td>
<td>Incorrect RAB header</td>
</tr>
<tr>
<td>0C</td>
<td>0C</td>
<td>Spool I/O error while reading the job or data set TAT for purging</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>Entry to be purged not found in the job or data set TAT</td>
</tr>
<tr>
<td>14</td>
<td>14</td>
<td>Spool I/O error while checkpointing the job or data set TAT</td>
</tr>
<tr>
<td>18</td>
<td>18</td>
<td>ARELEASE error</td>
</tr>
<tr>
<td>1C</td>
<td>1C</td>
<td>Error purging a track group from the RAB</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>Job or data set TAT FDB data address is zeros</td>
</tr>
<tr>
<td>24</td>
<td>24</td>
<td>Spool extent number too low</td>
</tr>
<tr>
<td>28</td>
<td>28</td>
<td>Spool extent not found</td>
</tr>
<tr>
<td>2C</td>
<td>2C</td>
<td>Spool record count too large</td>
</tr>
<tr>
<td>30</td>
<td>30</td>
<td>Spool record count too low</td>
</tr>
<tr>
<td>34</td>
<td>34</td>
<td>Spool record number too large</td>
</tr>
<tr>
<td>38</td>
<td>38</td>
<td>Spool record number too low</td>
</tr>
<tr>
<td>3C</td>
<td>3C</td>
<td>Error returned by the Slot Location routine</td>
</tr>
<tr>
<td>40</td>
<td>40</td>
<td>The track group is not allocated to the caller</td>
</tr>
</tbody>
</table>

Linkage
This macro uses ASAVE linkage.

Restrictions
Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task.

IATXRABP (Move Partial Track Group Between RAB and JOB TAT)
The IATXRABP macro moves partial track groups between a RAB (record allocation block) and a job TAT (track allocation table) FDB. This macro is issued by C/I (converter/interpreter) code to prevent the unnecessary allocation of additional track groups when alternating between USAM and JSAM data sets for a job. This helps prevent the wasting of spool space. Note that USAM obtains spool space allocations from the RAB, while JSAM obtains space from the track allocation table (JBTAT).

Syntax
```
IATXRABP(label=TYPE=CLOSE,RAB=(R1)address
```

Parameters
- **TYPE**
IATXRABP

OPEN
Specifications that the partial track group in the job TAT FDB is to be moved into
the RAB for subsequent USAM allocations.

CLOSE
Specifications that the partial track group in the RAB is to be moved into the job
TAT FDB for subsequent JSAM allocations.

FDB=
Specifies the address of the job TAT FDB.

RAB=
Specifies the address of the RAB.

Linkage
This macro uses ASAVE linkage.

Restrictions
Do not use this macro in any routine that might run under the JES3 auxiliary task.

Note: This macro is reserved for IBM.

IATXRCL (Return Cell to Cell Pool)
The IATXRCL macro returns a cell to the cell pool.

Syntax

\[
\text{label} \quad \text{IATXRCL} \quad \text{CPADDR} = (\text{reg}) \quad \text{CEL} = (R0) \quad \text{NORMAL} = (\text{R1}) \quad \text{ERROR} = (\text{R0})
\]

Parameters

CPADDR=
Specifies the primary cell pool control block (CPB) address for the cell pool
from which the requested cell is to be deallocated.

CELL=
Specifies the address of the cell that is to be deallocated. This address must be
on a fullword boundary.

NORMAL=
This parameter specifies that after the macro processing routine executes, the
issuing program can resume execution at an instruction other than the
instruction that immediately follows the macro.

ERROR=
This parameter specifies the address to which the macro processing routine
returns if an error occurs. When control is passed to the address, register 0
contains a dump code and register 15 contains a return code, unless otherwise
specified. The return codes are described below.

If an error occurs and ERROR was not specified, the dynamic support program
(DSP) will be ended by using the FAILDSP macro.
**Return Codes**
Following are the return codes returned in register 15:

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>04</td>
<td>The CPADDR parameter does not specify the address of a primary CPB.</td>
</tr>
<tr>
<td>08</td>
<td>The address specified in the CELL parameter is not valid.</td>
</tr>
<tr>
<td>20</td>
<td>The SPAN and PGRLSE (page release) keywords were specified when the cell pool was created (see the IATXBPL macro).</td>
</tr>
<tr>
<td>1XX</td>
<td>Return code from internal call to macro IATXDPL.</td>
</tr>
</tbody>
</table>

**Linkage**
This macro issues the IATZCALL macro without using ASAVE linkage.

**Restrictions**
Do not use this macro in any routine that might run under the JES3 auxiliary task.

**IATXRCVL (Validate a Routing Code or Destination Class)**
IATXRCVL validates a routing code or destination class and converts it to a routing code mask.

**Syntax**

```plaintext
IATXRCVL(label) INPARM=(reg) address, LENGTH=(reg) address, ROUT=(reg) address, MF=L(address)
```

**Parameters**

**INPARM=**
Specifies the address of a destination class or routing code to be validated and converted to a routing code mask. The parameter may be either a field containing the value to be validated and converted or a register containing the address of the value to be validated and converted. If the input parameter is a routing code, it must be in character format as opposed to binary data. If the input parameter is a JES3 destination class, it must be in 'name' format (e.g. MLG, LOG, ERR, and so on.). This parameter is required.

**LENGTH=**
Specifies the length of the INPARM input field containing the destination class or routing code. This parameter may be specified as a label of a halfword field which contains the length, a register which contains the length, or as L'FIELD. This parameter is required.

**ROUT=**
Specifies the address of a 17 byte field that will be used to hold the converted routing code mask. The 17-byte field is a bit mask where each of the 128 bits in the first 16 bytes represent a single MVS routing code. The last byte of the 17-byte field represents the JES3 ALL and MLG destination classes because
these classes do not have a routing code equivalent. The first bit of the 17th byte represents destination class ALL, while the 2nd bit represents destination class MLG. This parameter is required.

**MF=**

Specifies if the IATXRCVL request is in the list or execute form. The address specifies the location of the list form expansion of the macro. If this parameter is not specified, the IATXRCVL executable form is generated inline.

**Exit Registers — Normal**

R0  - Reason code indicating whether the input value was a destination class or a route code:

0 = Input was a destination class
4 = Input was a routing code

R1  - Unpredictable

R2-14 - Same as on entry

R15 - 0 - processing successful

**Exit Registers — Error**

R0  - Error reason code:

0 = Input area length is too small
4 = Input value is too long
8 = Error converting dest class to a routing code mask
12 = Error converting routing code to a routing code mask

R1  - Unpredictable

R2-14 - Same as on entry

R15 - 4 - processing unsuccessful

**Linkage**

This macro uses ASAVE linkage.

**Restrictions**

Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task.

**IATXRDCCH (Read a Chained, Single-Record File into Storage)**

Use IATXRDCCH to read a chained, single record file (SRF) into storage.

**Notes:**

1. If an error occurs, the caller is responsible for issuing an IATXRELC request to release any buffers that JES3 might have read.
2. This macro uses register 2 in its expansion.
Parameters

FDB=
 Specifies the address of the FDB for the chained single record file to be read.

ID=name
 Specifies the control block id associated with the single record file. JES3 examines each buffer as it is read in to make sure that it contains this id.

CHAIN=name
 Specifies the name of the field that contains the chain FDB. For example, for IATYJST the chain FDB is JSTCHN.

If you do not specify this parameter, then JES3 concatenates the ID parameter with the string ‘CHN’ to set the name of the chain field. For example, if you specify ID=JST and omit CHAIN, JES3 uses JSTCHN as the chain field.

BASE=name
 Specifies the name of the field that represents the base of the control block JES3 is reading. For example, for IATYJST the base is JSTSTART.

If you do not specify this parameter, then JES3 concatenates the ID parameter with the string ‘START’ to set the name of the base field. For example, if you specify ID=JST and omit BASE, JES3 uses JSTSTART as the base.

CHECK_ONSPOOL={NO | YES | ALL}
 Specifies whether JES3 should the FDB(s) to determine if the control block chain is on spool. Specify one of the following:

NO
 Tells JES3 to not check the FDB(s) to determine if the control block chain is on spool.

YES
 Tells JES3 check the first FDB to determine if the control block chain is on spool. If it is on spool, read the control block chain.

ALL
 Tells JES3 to check all of the FDBs to determine if the control block chain is on spool. JES3 start reading the control block chain when it encounters the the first buffer on spool. This is useful if you read one or more control block buffers into storage and then want to use IATXRDCCH to read the rest of the buffers.

ERROR=address
 Specifies the address to which to JES3 returns control if processing is unsuccessful. For example, an I/O error occurs or the control block id is not valid.

The following registers contain information related to the error. See the description of the JESREAD macro for more information about the error:

Register
IATXRDCH

R0               Contains the dump code
R15              Contains the return code

NORMAL=address   Specifies the address to which JES3 returns control to if processing is successful.

Linkage

None.

Restrictions

Do not use this macro in any routine that might run under the JES3 auxiliary task.

IATXRELC (Restore Spool Address in FDB)

The IATXRELC macro restores the spool address in the FDB for a group of chained single-record files (SRFs) to allow successive JESREAD macros.

Syntax

```
  label  IATXRELC FDB=(R1) address , ID= name

  DISP= 12 (reg) number  , NORMAL= address  , ERROR= address
```

Parameters

FDB=
Specifies the address of the FDB of the first SRF in the chain.

ID=
Specifies the name to be associated with the FDB. The name may not be more than four characters long.

Before restoring the spool address, IATXRELC compares the ID specified by this parameter (if any) to the contents of the ID field of the buffer associated with the FDB. If the IDs are not the same, the error exit is taken.

DISP=
Specifies the displacement into the SRF of the chain FDB, or a register containing the value in the low-order byte.

NORMAL=
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

ERROR=
This parameter specifies the address to which the macro processing routine returns if an error occurs. When control is passed to the address, register 0 contains a dump code and register 15 contains a return code, unless otherwise specified. The return codes and dump codes are described below. For a description of every dump code, see [z/OS JES3 Diagnosis]
If an error occurs and ERROR was not specified, the dynamic support program (DSP) will be ended by using the FAILDSP macro.

**Return Codes**
Following are the return codes and dump codes returned in registers 15 and 0, respectively:

<table>
<thead>
<tr>
<th>Return Code In Hex</th>
<th>Dump Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>04</td>
<td>723</td>
<td>Incorrect buffer address</td>
</tr>
<tr>
<td>08</td>
<td>nnn</td>
<td>Failure on internal call (APUTBUF)</td>
</tr>
<tr>
<td>0C</td>
<td>701</td>
<td>Closed FDB</td>
</tr>
<tr>
<td>10</td>
<td>723</td>
<td>Zero FDB, or IDs did not match</td>
</tr>
<tr>
<td>24</td>
<td>736</td>
<td>CSBT used with previous JESREAD</td>
</tr>
<tr>
<td>1C</td>
<td>736</td>
<td>CSBT identifier is incorrect</td>
</tr>
<tr>
<td>14</td>
<td>736</td>
<td>SRF identifier is incorrect</td>
</tr>
<tr>
<td>20</td>
<td>736</td>
<td>CSBT not used with previous JESREAD</td>
</tr>
</tbody>
</table>

**Linkage**
This macro uses ASAVE linkage.

**Restrictions**
Do not use this macro in any routine that might run under the JES3 auxiliary task.

**IATXRMEP (Branch to the JMF SSI Response Sampling Routine)**
The IATXRMEP macro is used by certain JES3 modules, for example, module IATSSCM, to invoke JMF sampling routines. This macro provides linkage to the specific JMF routine and checking macro IATYRMTV (Resource Monitoring Vector Table) control block for the following:

- **RMEPADR**
  Entry point address of the JMF sampling routine.

- **RMEPUCNT**
  Number of concurrent users of the JMF sampling routine.

- **RMEPFL1**
  Flag used to communicate information between JES3 and JMF driver modules.

- **RMEPSADR**
  Address of the Data Collection Area for JMF.

Using the SVT, the IATXRMEP macro determines if the RMVT table exists. If it does exist, IATXRMEP macro checks for and retrieves the address of the JMF SSI sampler code from an address in the SVT named SVTRMVT. If the SVTRMVT field contains zeros, it means that the sampling routine is not active. If JES3 is active, control is passed to the sampling routine.

**Syntax**

```
label IATXRMEP PASSADR=(R1) (Rx) address label, SVT=(Rx) (R9) label
```
IATXRMEP

\[ \text{\textasciitilde\textasciitilde} \rightarrow, \text{EP}={\textit{entry point label}}, \text{INUSEFLG}={\textit{label}}, \text{INUSEVAL}={\textit{label}} \text{\textasciitilde\textasciitilde} \]

Parameters

**PASSADR**
Defines an address to the JMF sampling routine that is passed in Register 1. Register 9 can not be used. PASSADR is an optional parameter, but certain JMF sampling routines may require it.

**SVT**
Defines the location of the address of the SVT control block passed in Register 9. The value specified may be a register or the label of the SVT control block. Register 1 cannot be used.

**EP**
Indicates which RMTV entry should be processed. The value of the EP parameter is combined with the string RMVT to produce the name of a predefined equate which identifies the index to the RMVT table. For example, EP=SSI were specified the equate which identifies the index to the SSI RMVT entry is RMVTSSI.

**INUSEFLG**
Identifies a flag that indicates whether the JMF sampling routine is in use.

**INUSEVAL**
Defines the value (a label) to be used in the INUSEFLG parameter.

Linkage

This macro uses ASAVE linkage.

Restrictions

Do not use this macro in any routine that might run under the JES3 auxiliary task. This macro can only be used when using JMF on z/OS hardware and may not issue an AWAIT.

IATXRST (Activate I/O Activity for a SNA RJP Session)

The IATXRST macro activates I/O activity for a SNA RJP session.

The routine using this macro must set register 11 according to the SNA RJP convention. In this convention, the high-order bit of register 11 is used to indicate the type of address (in the low-order 31 bits) as follows:

- If the bit is off (0), it is an FCT address.
- If the bit is on (1), it is an IRB address or an SRB address.

To distinguish an IRB from an SRB, test the PSATOLD word in the PSA. If this word is zero, the SRB is active. If the word is not zero, the IRB is active.

Syntax

```
IATXRST LUCB=(reg) address, MODE=DSP R8
```

414 z/OS V1R11.0 JES3 Customization
Parameters

**LUCB=**
Specifies the address of a logical unit control block (LCB) identifying the session that needs its I/O activity started or restarted.

**MODE=**
- **DSP**
  Indicates that the caller runs under a DSP.
- **RB**
  Indicates that the caller runs under a service request block (SRB) or an interruption request block (IRB).

Linkage

When the **MODE=DSP** parameter is specified, this macro uses ASAVE linkage. When the **MODE=RB** parameter is specified, this macro issues a BALR R14,R15 instruction for linkage.

Restrictions

Do not use this macro in any routine that might run in a C/I FSS address space.

IATXRUGT (Obtain an RU)

The IATXRUGT macro obtains a request/response unit (RU). If a previously obtained RU buffer exists, this macro releases it and makes it available for use.

Syntax

```
IATXRUGT label, SUP=(reg), EOD=(reg), address, NORMAL=(reg), ERROR=(reg), address
```

Parameters

**SUP=**
Specifies the address of the SUPUNITS table entry for the device requesting an RU.

**EOD=**
Specifies the address to which control returns if an EDS is received.

**NORMAL=**
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

**ERROR=**
This parameter specifies the address to which the macro processing routine returns if an error occurs. When control is passed to the address, register 0 contains a dump code and register 15 contains a return code, unless otherwise specified. The return codes are described below.
If an error occurs and ERROR was not specified, the dynamic support program (DSP) will be ended by using the FAILDSP macro.

**Return Codes**
Following are the return codes returned in register 15:

<table>
<thead>
<tr>
<th>Return Code In Hex</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>RU received. The address of the buffer entry (BFE) for the RU is returned in the LCBRUCR field of the logical unit control block (LCB). The address of the LCB is in register 1.</td>
</tr>
<tr>
<td>04</td>
<td>RU not received. No RU is returned.</td>
</tr>
<tr>
<td>08</td>
<td>Temporary device error. The device had errors, but is now available.</td>
</tr>
<tr>
<td>0C</td>
<td>Permanent device error. The device is not available.</td>
</tr>
<tr>
<td>10</td>
<td>Session error. The session is lost.</td>
</tr>
<tr>
<td>14</td>
<td>Session is ended by using a CANCEL or LOGOFF command.</td>
</tr>
</tbody>
</table>

**Linkage**
This macro uses ASAVE linkage.

**Restrictions**
This macro can be used only under the control of a DSP. Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task.

**IATXRUPT (Obtain an Empty RU)**
The IATXRUPT macro obtains an empty request/response unit (RU). Any previous RUs are available.

**Syntax**

```
IATXRUPT LUCB=(reg) (R1) address,ERROR=(reg) address,
```

**Parameters**

**LUCB=**
Specifies the address of a logical unit control block (LCB) of a session for which an empty RU is requested.

**ERROR=**
This parameter specifies the address to which the macro processing routine returns if an error occurs. When control is passed to the address, register 0 contains a dump code and register 15 contains a return code, unless otherwise specified. The return codes are described below.
If an error occurs and ERROR was not specified, the dynamic support program (DSP) will be ended by using the failDSP macro.

**NORMAL=**
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

**Return Codes**
Following are the return codes returned in register 15:

<table>
<thead>
<tr>
<th>Return Code</th>
<th>In Hex</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>00</td>
<td>Normal return.</td>
</tr>
<tr>
<td>04</td>
<td>04</td>
<td>Send end of chain (EOC) at next logical record.</td>
</tr>
<tr>
<td>0C</td>
<td>0C</td>
<td>Temporary device error. The device is not available for retry.</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>Permanent device error. The device is not available.</td>
</tr>
<tr>
<td>14</td>
<td>14</td>
<td>Session error. The session is lost.</td>
</tr>
<tr>
<td>18</td>
<td>18</td>
<td>Session is ended by using a CANCEL or LOGOFF command.</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>Intervention required on writer.</td>
</tr>
</tbody>
</table>

**Linkage**
This macro uses ASAVE linkage.

**Restrictions**
This macro must run under FCT control. Do not use this macro in any routine that might run in a C/I FSS address space.

**IATXSAS (Locate a Spool Data Set Entry or Validate a Spool Record Address)**
The IATXSAS macro locates a spool data set entry or validates a spool record address.

**Syntax**

```
label: IATXSAS TYPE=VAL,SPADR=address ERROR=|LOC,EXT=address ERROR=|
      MEM=J,REG=R1 REG=reg ERROR=:
```

```
,ERROR=address
```
Parameters

**TYPE=**
Specifies the type of operation to be performed.

**VAL**
Validate the JES3 spool record address identified by the SPADR parameter.
If you specify this parameter, you must also specify the ERROR parameter.

**LOC**
Locate the spool data set entry whose index is identified by the EXT parameter.

**EXT=**
Specifies the index of the spool data set to be located.

**SPADR=**
Specifies the address of a 6-byte field containing the spool record address to be validated.

**MEM=**
Specifies whether the request is for the JES3 or user address space.

- **J** The request is for the JES3 address space. Use the transfer vector table (TVT) to locate the I/O parameter area. The calling routine must have established addressability to the TVT.
- **U** The request is for the user address space. Use the JES3 subsystem vector table (SVT) to locate the I/O parameter area. The calling routine must have established addressability to the SVT.

**REG=**
Specifies a register that may be used by the macro expansion for addressing the I/O parameter block (IOP) spool data set entry. The macro expansion routine then places the address of the IOP spool data set entry in the specified register.

**ERROR=**
If you specify the TYPE=LOC parameter, this parameter specifies the location to which the macro expansion branches if the spool data set index is incorrect.

If you specify the TYPE=VAL parameter, this parameter is required and specifies the location to which the macro expansion branches if the spool record address is incorrect or the record number portion of the spool record address is incorrect.

**Return Codes**
Following are the return codes returned in register 15 when the TYPE=VAL parameter is specified:

<table>
<thead>
<tr>
<th>Return Code In Hex</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>04</td>
<td>The spool data set index portion of the spool record address is incorrect. The register specified using the REG parameter contains the address of the IATYIOP control block.</td>
</tr>
</tbody>
</table>
Return Code In Hex | Meaning
---|---
08 | The record number portion of the spool record address is incorrect. The register specified using the REG parameter contains the address of the corresponding IOP data set entry.

This macro does not return any error codes when the TYPE=LOC parameter is specified.

**Linkage**

This macro generates inline code.

**Restrictions**

Do not use this macro in any routine that might run under the JES3 auxiliary task.

**IATXSCH (Schedule a Job to a CI, POSTSCAN, or DISABLE DSP)**

The IATXSCH macro schedules a job to a CI, POSTSCAN, or DISABLE DSP. The macro sets up parameters and calls the needed scheduling routine. The macro can also map the parameter list to be passed to the scheduling routine.

**Syntax**

```
<table>
<thead>
<tr>
<th>TYPE</th>
<th>OPTIONS</th>
<th>PTYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>CI</td>
<td>DSPCHECK</td>
<td>JQE</td>
</tr>
<tr>
<td>POSTSCAN</td>
<td>OPTIONS</td>
<td>RQ</td>
</tr>
<tr>
<td>DISABLE</td>
<td>PTYPE</td>
<td>OPTIONS</td>
</tr>
</tbody>
</table>
```

**PARMS:**

- `PARMS=address`
- `NAVAIL=address`
- `NORMAL=address`

**OPTIONS:**

- `OPTIONS=DSPCHECK`
- `OPTIONS=RESTART`

**PTYPE:**

- `PTYPE=JQE`
- `PTYPE=JCT`
- `PTYPE=RQ`
Parameters

**TYPE=**
- **CI** Specifies that the job is to be scheduled for the CI DSP.
- **POSTSCAN** Specifies that the job is to be scheduled for the POSTSCAN DSP.
- **DISABLE** Specifies that the job is to be scheduled for the DISABLE DSP.
- **DSECT** Specifies that a mapping of the parameter list to be passed to the scheduling routine should be generated. The return code definitions should also be generated. This parameter must be used alone (without any other parameters).

**PARMS=**
Specifies the address of a parameter list to be passed to the scheduling routine. The parameter list is mapped by this parameter specifying TYPE=DSECT. The PTYPE parameter identifies which parameters are included in the parameter list.

**PTYPE=**
- **JQE** Specifies that the parameter list contains only the address of the job’s job queue element (JQE). This parameter is valid only with the TYPE=CI parameter.
- **JCT** Specifies that the parameter list contains the addresses of the job’s JQE, JCT, and current scheduler element.
- **RQ** Specifies that the parameter list contains the address of the job’s RESQUEUE entry. This parameter is valid only with the TYPE=POSTSCAN parameter.

**OPTIONS=**
- **DSPCHECK** Specifies that the scheduling routine should check to see if a DSP is available and return the result to the caller. The job is not actually scheduled to the DSP if this parameter is coded. This parameter is valid only with the TYPE=CI parameter.
- **RESTART** Specifies that the scheduling routine should perform only restart processing. No FCT or RESQUEUE is to be set up. This parameter is valid only with the TYPE=DISABLE parameter.

**NAVAIL=**
Specifies the return address if the job could not be scheduled. Register 15 contains the return code.

**NORMAL=**
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.
Return Codes
Following are the return codes returned in register 15 for a TYPE=CI request:

<table>
<thead>
<tr>
<th>Return Code In Hex</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>No CI DSPs are available.</td>
</tr>
<tr>
<td>04</td>
<td>No processor is available for LOCATE processing.</td>
</tr>
<tr>
<td>08</td>
<td>A needed procedure library is unavailable for use.</td>
</tr>
<tr>
<td>0C</td>
<td>No RESQUEUE is available.</td>
</tr>
<tr>
<td>10</td>
<td>The address space for the CI DSP selected for this job was rejected by installation exit routine</td>
</tr>
<tr>
<td></td>
<td>IATUX49 and all otherwise eligible CI DSPs were rejected as a result of the CIBATCH or CIDemand</td>
</tr>
<tr>
<td></td>
<td>specification.</td>
</tr>
</tbody>
</table>

Following are the return codes returned in register 15 for a TYPE=POSTCAN request:

<table>
<thead>
<tr>
<th>Return Code In Hex</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>No POSTSCAN DSPs are available.</td>
</tr>
<tr>
<td>04</td>
<td>No processor is available for LOCATE processing.</td>
</tr>
<tr>
<td>08</td>
<td>No RESQUEUE is available.</td>
</tr>
</tbody>
</table>

Following are the return codes returned in register 15 for a TYPE=DISABLE request:

<table>
<thead>
<tr>
<th>Return Code In Hex</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>No DISABLE DSP is available.</td>
</tr>
<tr>
<td>04</td>
<td>A needed procedure library is unavailable for updating.</td>
</tr>
<tr>
<td>08</td>
<td>No RESQUEUE is available.</td>
</tr>
</tbody>
</table>

Linkage
This macro uses ASAVE linkage.

Restrictions
Only the job segment scheduler (JSS) and the converter/interpreter driver (CIDRVR) DSP may issue this macro.

IATXSCN1 (Scan Input Commands)

The IATXSCN1 macro scans the operands of an input command in succession. You use this macro to scan one operand at a time. If a command is examined that contains multiple operands within a sublist, the next non-sublist operand is returned. You should use the IATXSCN2 macro to scan operands within a sublist. If two consecutive commas are detected by this macro, registers 0 and 1 will contain zero.
IATXSCN1

On return from this macro, register 0 will contain either 0 or the address of the operand being scanned, and register 1 will contain either 0 or the address of the keyword.

The service routine for this macro passes back the length values in register 2. The high-order byte of register 2 contains the length of the keyword being scanned. The low-order byte of register 2 contains the length of the operand being scanned.

If an error was detected, register 0 will contain the address of the parameter, and register 2 will contain the length of the residual portion of the command in its low-order byte.

Syntax

```
IATXSCN1 PFX=label(R1),EOD=address,ERROR=address

,BLANK=DLM,EOD,
,NORMAL=address
```

Parameters

**PFX=**

Specifies the address of the console message prefix of the command to scan.

**EOD=**

Specifies the address to which the routine returns when the end of the command is detected.

**ERROR=**

Specifies the address to which the routine returns when a syntax error is detected. Register 0 will contain the address of the parameter being scanned. The residual length of the command is in the low-order byte of register 2.

**BLANK=**

Specifies how a blank is interpreted.

- **DLM**
  
  Specifies that a blank is a normal delimiter and any number of contiguous blanks is treated as one comma.

- **EOD**
  
  Specifies that a blank is to be interpreted as the end of data (EOD).

**NORMAL=**

This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

Linkage

This macro uses ASAVE linkage.

Restrictions

None.
IATXSCN2 (Scan Input Command Sublists)

The IATXSCN2 macro scans sublist operands (that is, multiple operand values within parenthesis) from console input commands. This macro scans sublist operands in succession until it finds a closing parentheses “)”, in which case an immediate end-of-data (EOD) return is taken.

On return from this macro, register 0 will contain either 0 or the address of the parameter being scanned, and register 1 will contain either 0 or the address of the keyword.

The service routine for this macro passes back the length values in register 2. The high-order byte of register 2 contains the length of the keyword being scanned. The low-order byte of register 2 contains the length of the operand being scanned.

If an error was detected, register 0 will contain the address of the operand, and register 2 will contain the length of the remaining portion of the command in its low-order byte.

Syntax

```
/SM590000
IATXSCN2
PFX=\(R1\)
EOD=\(address\)
ERROR=\(address\)

NO
HYPHEN=\(YES\)
NORMAL=\(address\)
```

Parameters

PFX=  
Specifies the address of the console message prefix for the command to scan.

EOD=  
Specifies the address to which the routine returns when the end of the sublist is detected.

ERROR=  
Specifies the address to which the routine returns when a syntax error is detected. Register 0 will contain the address of the operand. The low-order byte of register 2 will contain the length of the remainder of the command.

HYPHEN=  
Specifies if JES3 should scan for a hyphen as a delimiter in the parameter sublist. If you omit this parameter, the default value is NO.

NORMAL=  
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

Linkage

This macro uses ASAVE linkage.

Restrictions

Do not use this macro in any routine that might run under the JES3 auxiliary task.
IATXSDM (Initialize the Spool Data Management List)

The IATXSDM macro provides your installation DSP, when it has created a subtask environment, with a method for the subtask to initialize access to the spool data set. The IATXSDM macro initializes the spool data management parameter list and invokes the spool access initialization routine (IATDMGR).

Syntax

```
IATXSDM FUNC= [ALLOCU | DEALLOCU | OPENRU | OPENWU | CLOSEU | ALLOCB | DEALLOCB | SDM= \(R1\), SDMAD= address ]

,...SSVT= \(R0\), SSVTAD= address

,...MEM= (reg), MEMAD= address

,...DSS= (reg), DSSAD= address

,...DEB= (reg), DEBAD= address

,...RAB= (reg), RABMAD= address

,...BLK= (reg), BLKAD= address

,...CPP= (reg), CPPAD= address

,...JOBNO= (reg), DDANME= (reg)

,...FSPADR= (reg), LSPADR= (reg)

,...PRMXTNT= (reg), SECXTNT= (reg)

,...CELLSP= (reg), CELLID= (reg), ERROR= address

,...CALL= YES
```
Parameters

The following table summarizes the parameter interdependencies of this macro. The parameter explanations follow this table.

Table 18. Parameter Interdependencies for IATXSDM

<table>
<thead>
<tr>
<th>Value of FUNC=</th>
<th>Other Parameters Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALLOCU</td>
<td>MEM= or MEMAD=</td>
</tr>
<tr>
<td>DEALLOCU</td>
<td>DSS= or DSSAD=</td>
</tr>
<tr>
<td></td>
<td>DEB= or DEBAD=</td>
</tr>
<tr>
<td></td>
<td>MEM= or MEMAD=</td>
</tr>
<tr>
<td>OPENRU</td>
<td>DSS= or DSSAD=</td>
</tr>
<tr>
<td></td>
<td>DEB= or DEBAD=</td>
</tr>
<tr>
<td></td>
<td>RAB= or RABAD=</td>
</tr>
<tr>
<td></td>
<td>ACB= or ACBAD=</td>
</tr>
<tr>
<td></td>
<td>DDNAME=</td>
</tr>
<tr>
<td></td>
<td>FSPADR=</td>
</tr>
<tr>
<td>OPENWU</td>
<td>DSS= or DSSAD=</td>
</tr>
<tr>
<td></td>
<td>DEB= or DEBAD=</td>
</tr>
<tr>
<td></td>
<td>RAB= or RABAD=</td>
</tr>
<tr>
<td></td>
<td>ACB= or ACBAD=</td>
</tr>
<tr>
<td></td>
<td>JOBNO=</td>
</tr>
<tr>
<td></td>
<td>DDNAME=</td>
</tr>
<tr>
<td></td>
<td>FSPADR=</td>
</tr>
<tr>
<td></td>
<td>LSPADR=</td>
</tr>
<tr>
<td>CLOSEU</td>
<td>DSS= or DSSAD=</td>
</tr>
<tr>
<td></td>
<td>DEB= or DEBAD=</td>
</tr>
<tr>
<td></td>
<td>ACB= or ACBAD=</td>
</tr>
<tr>
<td>ALLOCB</td>
<td>BLK= or BLKAD=</td>
</tr>
<tr>
<td></td>
<td>CPP= or CPPAD=</td>
</tr>
<tr>
<td></td>
<td>PRMXTNT=</td>
</tr>
<tr>
<td></td>
<td>SECXTNT=</td>
</tr>
<tr>
<td></td>
<td>CELLS=</td>
</tr>
<tr>
<td>DEALLOCB</td>
<td>BLK= or BLKAD=</td>
</tr>
<tr>
<td></td>
<td>CELLD=</td>
</tr>
</tbody>
</table>

**FUNC=**

Specifies the function to be performed.

**ALLOCU**
Allocate – USAM function.

**DEALLOCU**
Deallocate – USAM function.

**OPENRU**
Open for read – USAM function.

**OPENWU**
Open for write – USAM function.

**CLOSEU**
Close – USAM function.

**ALLOCB**
Allocate – block spooler function.

**DEALLOCB**
Deallocate – block spooler function.
SDM=
Specifies the address of the spool data management parameter list (mapped by IATYSDM) to be used by the IATXSDM service routine. If address is coded, SDM= specifies the label name where the SDM is defined. Either SDM or SDMAD must be specified, but not both.

SDMAD=
Specifies a 4-byte field containing the address of the spool data management parameter list (mapped by IATYSDM). Either SDM or SDMAD must be specified, but not both.

SSVT=
Specifies the address of the subsystem vector table (mapped by IATYSVT). If address is coded, SSVT= specifies the label name where the SSVT is defined. Either SSVT or SSVTAD must be specified, but not both.

SSVTAD=
Specifies a 4-byte field containing the address of the subsystem vector table (mapped by IATYSVT). Either SSVT or SSVTAD must be specified, but not both. If neither SSVT or SSVTAD is specified, register zero is assumed.

MEM=
Specifies the address of a memory data block (mapped by IATYMEM). If address is coded, MEM= specifies the label name where the MEM is defined. If MEM= is specified, MEMAD= must not be specified. Either MEM or MEMAD must be specified for FUNC=ALLOCU and FUNC=DEALLOCU.

MEMAD=
Specifies a 4-byte field containing the address of the memory data block (mapped by IATYMEM). If MEMAD= is specified, MEM= must not be specified. Either MEM or MEMAD must be specified for FUNC=ALLOCU and FUNC=DEALLOCU.

DSS=
Specifies the address of a data set status block (mapped by IATYDSS). If address is coded, DSS= specifies the label name where the DSS is defined. If DSS= is specified, DSSAD= must not be specified. Either DSS or DSSAD must be specified for FUNC=DEALLOCU, FUNC=OPENRU, FUNC=OPENWU, and FUNC=CLOSEU.

DSSAD=
Specifies a 4-byte field containing the address of the data set status block (mapped by IATYDSS). If DSSAD= is specified, DSS= must not be specified. Either DSS or DSSAD must be specified for FUNC=DEALLOCU, FUNC=OPENRU, FUNC=OPENWU, and FUNC=CLOSEU.

DEB=
Specifies the address of a data extent block (mapped by IEZDEB). If address is coded, DEB= specifies the label name where the DEB is defined. If DEB= is specified, DEBAD= must not be specified. Either DEB or DEBAD must be specified for FUNC=DEALLOCU, FUNC=OPENRU, FUNC=OPENWU, and FUNC=CLOSEU.

DEBAD=
Specifies a 4-byte field containing the address of the data extent block (mapped by IEZDEB). If DEBAD= is specified, DEB= must not be specified. Either DEB or DEBAD must be specified for FUNC=DEALLOCU, FUNC=OPENRU, FUNC=OPENWU, and FUNC=CLOSEU.

RAB=
Specifies the address of a record allocation block (mapped by IATYRAB). If
address is coded, RAB= specifies the label where the RAB is defined. If RAB= is specified, RABAD= must not be specified. Either RAB or RABAD must be specified for FUNC=OPENRU and FUNC=OPENWU.

RABAD=
Specifies a 4-byte field containing the address of the record allocation block (mapped by IATYRAB). If RABAD= is specified, RAB= must not be specified. Either RAB or RABAD must be specified for FUNC=OPENRU and FUNC=OPENWU.

ACB=
Specifies the address of an access method control block (mapped by IFGACB). If address is coded, ACB= specifies the label where the ACB is defined. If ACB= is specified, ACBAD= must not be specified. Either ACB or ACBAD must be specified for FUNC=OPENRU, FUNC=OPENWU, and FUNC=CLOSEU.

ACBAD=
Specifies a 4-byte field containing the address of the access method control block (mapped by IFGACB). If ACBAD= is specified, ACB= must not be specified. Either ACB or ACBAD must be specified for FUNC=OPENRU, FUNC=OPENWU, and FUNC=CLOSEU.

BLK=
Specifies the address of a block I/O parameter list (mapped by IATYBLK). If address is coded, BLK= specifies the label where the block is defined. If BLK= is specified, BLKAD= must not be specified. Either BLK or BLKAD must be specified for FUNC=ALLOCB and FUNC=DEALLOCB.

BLKAD=
Specifies a 4-byte field containing the address of a block I/O parameter list (mapped by IATYBLK). If BLKAD= is specified, BLK= must not be specified. Either BLK or BLKAD must be specified for FUNC=ALLOCB and FUNC=DEALLOCB.

CPP=
Specifies the address of an MVS cell pool parameter list. If address is coded, CPP= specifies the label where the CPP is defined. If CPP= is specified, CPPAD= must not be specified. Either CPP or CPPAD must be specified for FUNC=ALLOCB.

CPPAD=
Specifies a 4-byte field containing the address of a MVS cell pool parameter list. If CPPAD is specified, CPP= must not be specified. Either CPP or CPPAD must be specified for FUNC=ALLOCB.

JOBNO=
Specifies the binary job number of the job being processed. If address is coded, JOBNO= specifies a 4-byte field containing the binary job number. If a register is specified, the register must contain the job number. JOBNO must be specified for FUNC=OPENWU.

DDNAME=
Specifies the DDNAME of the data set being opened. If address is coded, DDNAME= specifies an 8-byte field containing the DDNAME. DDNAME must be specified for FUNC=OPENRU, and FUNC=OPENWU.

FSPADR=
Specifies the address of the data area that contains the first spool address being opened. If address is coded, FSPADR= specifies a 6-byte field containing the first spool address. FSPADR must be specified for FUNC=OPENRU and FUNC=OPENWU.
LSPADR=
Specifies the address of the data area that contains the last spool address being opened. If address is coded, LSPADR= specifies a 6-byte field containing the last spool address. LSPADR must be specified for FUNC=OPENWU.

PRMXTNT=
Specifies the primary extent size, in number of cell pool buffers, to be used for the block spooler. If address is coded, PRMXTNT= specifies a 2-byte field containing the primary extent size. PRMXTNT must be specified for FUNC=ALLOCB.

SECTNT=
Specifies the secondary extent size, in number of cell pool buffers, to be used for the block spooler. If address is coded, SECTNT= specifies a 2-byte field containing the secondary extent size. SECTNT must be specified for FUNC=ALLOCB.

CELLSP=
Specifies the subpool of the buffer cell pool to be used for the block spooler. CELLSP must be specified for FUNC=ALLOCB.

CELLID=
Specifies the cell pool identifier of the buffer cell pool used for the block spooler. If address is coded, CELLID= specifies a 4-byte field containing the cell pool identifier. CELLID must be specified for FUNC=DEALLOCB.

ERROR=
Specifies a return address that will get control if error occurs. This parameter is useful because it allows processing to continue even if an error occurs.

If ERROR= is specified, register 15 contains the ABEND reason code. These reason codes are defined in the IATY8FB macro as follows:

<table>
<thead>
<tr>
<th>Code</th>
<th>Label</th>
<th>Module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X'06'</td>
<td>R8FBUPWT</td>
<td>IATDMGR</td>
<td>USAM point or write error</td>
</tr>
<tr>
<td>X'07'</td>
<td>R8FDSSB</td>
<td>IATDMGR</td>
<td>DSS/DSB initialization error</td>
</tr>
<tr>
<td>X'08'</td>
<td>R8FUBFI</td>
<td>IATDMGR</td>
<td>USAM buffer initialization error</td>
</tr>
<tr>
<td>X'11'</td>
<td>R8FIVFC</td>
<td>IATDMGR</td>
<td>invalid function code</td>
</tr>
<tr>
<td>X'12'</td>
<td>R8FGRBK</td>
<td>IATDMGR</td>
<td>block spooler error</td>
</tr>
</tbody>
</table>

**Note:** Labels are equates for the values as defined in macro IATY8FB.

CALL=
Specifies whether the spool access initialization routine, IATDMGR, should be called (YES), or should not be called (NO). The default is CALL=YES. CALL=NO may be used if you want to build the parameter list (or partially build it), but then wait until after certain events (such as a resource becoming available) before invoking IATDMGR.

Linkage
This macro uses ASAVE linkage.

Restrictions
Only a JES3 subtask may issue this macro.
IATXSEC (Security Authorization Services)

The IATXSEC macro is an interface to the security authorization facility (SAF). Before passing control to SAF, installation exit IATUX58 is invoked. Before control returns to your routine, installation exit IATUX59 may be invoked depending on the return code returned from IATUX58. For more information about SAF, see z/OS MVS Programming: Assembler Services Guide.

This macro is issued to perform one of the following requests:

**REQUEST=AUDIT**
Create and audit record.

**REQUEST=AUTH**
Request authorization to use a JES3 resource. This JES3 resource is identified through the CLASS= and ENTITY= parameters.

**REQUEST=EXTRACT**
Extract information from a profile (TYPE=EXTRACT) or encrypt information (TYPE=ENCRYPT).

**REQUEST=TOKENBLD**
Modify a security token with the information in the coded parameters.

**REQUEST=TOKENMAP**
Obtain an unencrypted version of the encrypted security token or encrypt an unencrypted security token.

**REQUEST=TOKENXTR**
Extract a user’s token from RACF.

**REQUEST=VERIFYX**
Verify that a user can perform a particular task.

**Syntax**

```
IATXSEC label PARMLIST=(reg) address, INDEX=SSX-xxxx (reg)
```

```
MODE=NUCTASK INIT SUBTASK USER
```

```
REQUEST=AUDIT REQUEST=AUTH
REQUEST=EXTRACT, TYPE=EXTRACT
REQUEST=EXTRACT, TYPE=ENCRYPT
REQUEST=TOKENBLD
REQUEST=TOKENMAP
REQUEST=TOKENXTR
REQUEST=VERIFY
REQUEST=VERIFYX
```

**General Options:**

```
DEFAULT=(reg) address
```

Chapter 2. JES3 Macros 429
REQUEST=EXTRACT, TYPE=ENCRYPT:

REQUEST=TOKENBLD:

REQUEST=TOKENMAP:
Parameters

**ABEND=**
Specifies whether control should be returned to the calling routine if an abnormal end occurs while processing the request.

**ACCEPT=**
Specifies the address to which control returns if the decision is to accept the security request. If this parameter is not specified, control returns to the next sequential instruction after the IATXSEC macro invocation.

**ACTINFO=**
Specifies the address of an area containing accounting information. The accounting information area is 144 bytes in length and must be specified in the following format:
- The first byte of the area contains the number (in binary) of accounting fields.
- The remaining bytes contain accounting fields. Each accounting field entry begins with a 1-byte length field (in binary) followed by the field.

**ATTR=**
Specifies the access authority that the user or group must have to be permitted access to the JES3 resource for which authorization checking is to be performed:
- **Alter**
  User or group has total control over the resource.
- **Control**
  For VSAM data sets, the user or group has authority equivalent to the VSAM control password. For non-VSAM data sets and other resources, the user or group has UPDATE authority.
Update
User or group can open the resource to read or write.

Read
User of group can open the resource to read only.

This parameter is valid only for REQUEST=AUTH macro invocations. See [z/OS JES3 Initialization and Tuning Guide](https://www.ibm.com/support/knowledgecenter/SSSA5P_5.2.0/com.ibm.zos.v52.cic.lguides.jes3init/jes3init_tuning.html) for specific information about how the authorization levels affect the specific JES3 classes (CLASS=).

**CLASS=**
Specifies the name of the SAF class as one of the following:

<table>
<thead>
<tr>
<th>Resource:</th>
<th>SAF Class:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Job data sets (SYSIN/SYSOUT)</td>
<td>JESSPOOL</td>
</tr>
<tr>
<td>JES3 writers</td>
<td>WRITER</td>
</tr>
<tr>
<td>TSO SUBMIT/CANCEL</td>
<td>JESJOBS</td>
</tr>
<tr>
<td>Commands (other than RJP/NJE)</td>
<td>OPERCMDS</td>
</tr>
<tr>
<td>USERIDS</td>
<td>USER</td>
</tr>
</tbody>
</table>

This parameter is valid only for REQUEST=AUTH or REQUEST=EXTRACT (TYPE=EXTRACT) macro invocations.

**DEFAULT=**
Specifies the address to which control returns if a decision could not be made about whether to accept or reject the security request. If this parameter is not specified, control returns to the next sequential instruction after the IATXSEC macro invocation.

**INDEX=**
Specifies the particular invocation of IATXSEC. Installation exits IATUX58 and IATUX59 use this value to determine the security call that is being made by JES3. The format of the equates for the index value as defined in IATYSSX are SSXiZZXX, where:

- **zz** is:
  - IA if REQUEST=AUTH
  - IE if REQUEST=EXTRACT
  - IB if REQUEST=TOKENBLD
  - IM if REQUEST=TOKENMAP
  - IX if REQUEST=TOKENXTR
  - VF if REQUEST=TOKENXTR
  - IV if REQUEST=VERIFYX
  - AU if REQUEST=AUDIT

- **xxx** is:
  - A value of U01 – U32 for a user generated IATXSEC invocation.

**ENCRYPT=**
When used with REQUEST=EXTRACT, TYPE=ENCRYPT, ENCRYPT= specifies the address of the data to be encrypted.

When used with REQUEST=VERIFYX, ENCRYPT= specifies whether (YES) or not (NO) the password needs to be encrypted.

**ENTITY=**
Specifies the address of the resource name for which security processing is to be performed. For example, for an *INQUIRY,X command the entity name is JES3.DISPLAY.X
If a TSO user cancels a job, the entity name is 
CANCEL.node.userid.jobname

The entity name can be from zero to 44 bytes long. This parameter is valid only for REQUEST=AUTH or REQUEST=EXTRACT (TYPE=EXTRACT) macro invocations.

**ENVIR=**  
Specifies the disposition of the ACEE. This parameter is only valid for REQUEST=VERIFY macro invocations.

- **CREATE**  
  Creates an ACEE for the VERIFY request.

- **DELETE**  
  Deletes an ACEE for the VERIFY request.

**EXENODE=**  
Specifies the address of the execution node name that is valid only when used with the TOKENBLD and VERIFYX parameters.

**FDB=**  
Specifies the address of the FDB for data sets containing messages. It is used with MSGCNTL=(, JESMSG).

**FIELDS=**  
Specifies the address of the user’s password to be extracted from the profile. This parameter is valid only for REQUEST=EXTRACT, TYPE=EXTRACT macro invocations.

**FORMOUT=**  
Specifies the format for the output token (TOKNOUT) as one of the following:

- **INTERNAL**  
  SAF converts the token from the external to the internal (encrypted) format.

- **EXTERNAL**  
  SAF converts the token from the internal (encrypted) to the external format.

This parameter is valid only for REQUEST=TOKENMAP macro invocations.

**GROUP=**  
Specifies the address of the group name for the job. See your RACF administrator for valid RACF group names. This parameter is valid only for REQUEST=TOKENBLD and REQUEST=VERIFYX macro invocations.

**JOBNAME=**  
Specifies the address of the eight-character job name. This parameter is valid only for REQUEST=VERIFYX macro invocations.

**LOG=**  
Specifies how RACF should log the IATXSEC call (LOGSTR=) in the SMF data sets. This parameter is valid only for REQUEST=AUTH macro invocations.

- **ASIS**  
  RACF should record the calls in the manner specified in the profile that protects the resource. See [z/OS Security Server RACF Security Administrator’s Guide](https://www.ibm.com/docs/en/zos-security-server) for more information about RACF profiles.

- **NOFAIL**  
  If the authorization check fails, the attempt is not recorded. If the authorization check succeeds, the attempt is recorded in the manner specified in the RACF profile that protects the resource.
NOSTAT
   The attempt is not recorded and no resource statistics are updated.

NONE
   The call is not recorded.

LOGSTR=
   Specifies the address of a character string that RACF adds to the SMF record. This character string gives an indication of why the IATXSEC macro call was made. The SMF log string can be a maximum of 255 bytes long. The first byte must contain the length of the string. This parameter is valid only for REQUEST=AUTH, REQUEST=AUDIT, REQUEST=VERIFY, and REQUEST=VERIFYX macro invocations.

MODE=
   Specifies the phase of JES3 processing the caller of IATXSEC is in as one of the following:

NUCTASK
   The caller is running under the JES3 Nucleus task.

INIT
   IATXSEC is being issued during JES3 initialization.

SUBTASK
   The caller is running under a JES3 subtask.

USER
   The caller is running in a user's address space.

MSGCNTL=
   Specifies how messages from SAF, installation exit IATUX58, or installation exit IATUX59 are handled.

WTO
   Specifies that SAF or the installation exit should write messages to the operator.

SUPP
   Specifies that SAF or the installation exit should not write messages to the operator.

RETURN
   Specifies that messages should be returned to the caller. If RETURN is specified, the field SECMSGHD in IATYSEC contains the address of the message prefix for the first message in the chain upon return. The message prefix is mapped by IATYSEMS and the message is mapped by IEZWPL.

   The caller is responsible for freeing the storage used for the messages. The subpool that should be used on the APUTMAIN or FREEMAIN macro is SAFMSGSP which is defined in IATYEQU.

NEWPASS=
   Specifies the address of an eight-byte field that contains the user's new password. The field must be left justified and padded with blanks. This parameter is valid only for REQUEST=VERIFYX macro invocations.

NORETURN
   Specifies that messages should not be returned to the caller.

JESMSG
   Specifies that messages should be written to the job's JESMSGLG data
set. If JESMSG is specified, the caller must be executing under the JES3 NUC task and either the FDB or the RQ= parameter must be specified.

**NOJESMSG**
Specifies that messages should not be written to the job’s JESMSGLG data set.

**PARMLIST=**
Specifies the address of the security check control block mapped by IATYSEC. [z/OS JES3 Diagnosis] provides a procedure for formatting JES3 mapping macros.

**PASCHK=**
Specifies whether the password should be checked by RACF. This parameter is valid only for REQUEST=VERIFY and REQUEST=VERIFYX macro invocations.

**PASSWORD=**
Specifies the address of an eight-byte field that contains the password. The field must be left justified and padded with blanks. This parameter is valid only for REQUEST=VERIFYX macro invocations.

**POE=**
Specifies the address of an eight-byte field containing the name of the input device (port of entry) that the job was submitted from. For example, the port of entry for a job submitted through a card reader is the SUPUNIT DD name of the card reader device. The field must be left justified and padded with blanks. This parameter is valid only for REQUEST=TOKENBLD and REQUEST=VERIFYX macro invocations.

**RECVR=**
Specifies the address of an eight-byte field containing the user id of the user who has the authority to access the resource when there are no profiles for the resource. For example, in normal cases, if user A sends a data set to user B, user B would have to be authorized to access user A’s data sets. However, by specifying RECVR=useridB, user B can receive the data set without being authorized. This parameter is valid only for REQUEST=AUTH macro invocations.

**REJECT=**
Specifies the address to which control returns if the decision is to reject the security request. If this parameter is not specified, control returns to the next sequential instruction after the IATXSEC macro invocation. This parameter is valid only for REQUEST=AUTH, REQUEST=EXTRACT, and REQUEST=VERIFYX macro invocations.

**REMOTE=**
Specifies whether the job came through the network. If REMOTE=YES, the job is an NJE job and propagation is not required. If REMOTE=NO, the job is a local job and propagation is required. This parameter is valid only for REQUEST=TOKENBLD and REQUEST=VERIFYX macro invocations. The default is REMOTE=NO.

**REQUEST=**
Specifies the type of IATXSEC macro invocation as one of the following:

**REQUEST=AUDIT**
Create an audit record.

**REQUEST=AUTH**
Request authorization to use a JES3 resource. This JES3 resource is identified through the CLASS= and ENTITY= parameters.
REQUEST=EXTRACT
Extract information from a profile (TYPE=EXTRACT) or encrypt information (TYPE=ENCRYPT). See your RACF administrator for information about the active profiles for your installation.

REQUEST=TOKENBLD
Modify a security token with the information in the coded parameters.

REQUEST=TOKENMAP
Obtain a decoded version of the encrypted security token or encrypt a decoded security token.

REQUEST=TOKENXTR
Extract a user’s token from RACF.

REQUEST=VERIFYX
Verify that a user can perform a particular task.

RQ=
Specifies the address of a resident job queue (RQ) that JES3 uses when the macro specifies that message are to written to the JESMSGGLG data set. The FDB or the RQ= parameter must be specified when MSGCNTL=(*,*,JESMSG) is specified.

RTOKEN=
Specifies the address of a resource security token. The resource token is the user token the creator of the resource has at the time the resource was created. For example, when a TSO user cancels a job, the resource token is the user token of the person who submitted the job in the first place. This parameter is valid only for REQUEST=AUTH macro invocations.

SECLABL=
Specifies the address of an eight-byte field containing the security label. The security label represents the association between a particular security level (for example, FOR YOUR EYES ONLY) and a set of security categories (for example, CONTROL and KAOS). The field must be left-justified and padded with blanks. This parameter is valid only for REQUEST=TOKENBLD and REQUEST=VERIFYX macro invocations. For information about the security labels for your installation, contact your RACF administrator.

SESSION=
Specifies the type of session as one of the following:
- INTBATCH – For example, a job submitted through an internal reader.
- EXTBATCH – For example, a job submitted through a card reader.
- NJEBATCH – For example, a job submitted through NJE.
- RJEBATCH – For example, a job submitted through RJP.
- NJEOPER – For example, commands received from NJE.
- RJEOPER – For example, commands received from RJE.
- STARTED – Started task job.
- TSO – TSO logon.
- NJSYSOUT – For example, NJE SYSOUT received from another node.
- TKNUNKWN – For example, a request for a TOKEN containing the NJE UNKNOWN USERID.

This parameter is valid only for REQUEST=TOKENBLD and REQUEST=VERIFYX macro invocations.

SGROUP=
Specifies the address of an eight-byte field containing the submitor’s group name. This field must be left justified and padded with blanks. See your RACF
administrator for your installation's valid group names. This parameter is valid only for REQUEST=TOKENBUILD and REQUEST=VERIFYX macro invocations.

**SNODE=**
Specifies the address of an eight-byte field containing the submittor's node name. This field must be left justified and padded with blanks. This parameter is valid only for REQUEST=TOKENBUILD and REQUEST=VERIFYX macro invocations.

**START=**
Specifies the address to start verification.

**STOKEN=**
Specifies the address of the submittor's security token. This parameter is valid only for REQUEST=TOKENBUILD and REQUEST=VERIFYX macro invocations.

**SUSERID=**
Specifies the address of an eight-byte field containing the submitter's user id. The field must be left justified and padded with blanks. This parameter is valid only for REQUEST=TOKENBUILD and REQUEST=VERIFYX macro invocations.

**TRUSTED=**
Specifies whether (YES) or not (NO) the user is a member of the trusted computer base.

**TYPE=**
When invoking the macro with REQUEST=EXTRACT, TYPE= specifies the type of extract request:

- **EXTRACT**
  Specifies that fields are to be extracted from a profile. If TYPE=EXTRACT is specified, CLASS=, ENTITY=, and FIELDS= must also be specified.

- **ENCRYPT**
  Specifies that the data is to be encrypted. If TYPE=ENCRYPT is specified, ENTITY= and ENCRYPT= must also be specified.

**TOKNIN=**
Specifies the address of an input security token. This parameter is valid only for REQUEST=TOKENBUILD, REQUEST=TOKENMAP, REQUEST=TOKENXTR, REQUEST=VERIFY, and REQUEST=VERIFYX macro invocations.

**TOKNOUT=**
Specifies the address of a field to contain the output security token. This parameter is valid only for REQUEST=TOKENBUILD, REQUEST=TOKENMAP, REQUEST=TOKENXTR and REQUEST=VERIFYX macro invocations.

**USERID=**
Specifies the address of an eight-byte field containing the user id. The field must be left justified and padded with blanks. This parameter is valid only for REQUEST=TOKENBUILD and REQUEST=VERIFYX macro invocations.

**UTOKEN=**
For REQUEST=AUTH macro invocations, specifies the address of the security token of the user. The user token represents the person that requires authorization to the resource. For example, when a TSO user cancels a job, the user token represents the TSO user performing the cancel.

**Linkage**

When IATXSEC is issued with MODE=NUCTASK or MODE=INIT, linkage is through the ACALL macro using ASAVE linkage.
When IATXSEC is issued with MODE=SUBTASK or MODE=USER, linkage is through BAKR 0,15.

**IATXSIO (Initiate Spool I/O Operation)**

The IATXSIO macro starts an I/O operation to the spool data set.

**Syntax**

```
IATXSIO BUFFER=address,DMC=address,ERROR=address,
MEM=U,B,NORM=address
```

**Parameters**

- **BUFFER=** Specifies the address of the buffer to be used for the I/O operation.
- **DMC=** Specifies the address of the data management control block to be used for the I/O operation.
- **ERROR=** Specifies the address to which the routine returns if the request is not satisfied.
- **MEM=** Specifies whether the request is for the JES3 (J), user (U), or functional subsystem (B) address space.
- **TYPE=** Specifies the special handling to be performed by the channel end appendage. N indicates none, P indicates post JSAM FCT, and J indicates job TAT.
- **OPER=** Specifies the type of I/O operation: write (W), read (R), or multiread (M).
- **JTFDB=** Specifies the address of the job's track allocation table (TAT) to be used if track allocation becomes necessary.
- **NORMAL=** This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

**Linkage**

This macro issues a BALR R5,R6 for linkage.
IATXSIO

Restrictions
If this macro is issued by a routine running under the JES3 auxiliary task, it can be used only to do a multi-buffer read request. Other types of I/O requests are not supported by the auxiliary task.

If this macro is invoked outside of the JES3 Nuc or JES3 Aux task, the LOCAL lock must be held before invoking the IATXSIO service. This includes invocations from any user address space.

IATXSMF (Write SMF Records)
The IATXSMF macro writes a system management facility (SMF) record.

Syntax
```
ENTRY = (RI)
    label
    address
    JMR
    "MODE = A"

RQ = (reg)
    NORMAL = address
```

Parameters

**ENTRY=**
Specifies the address of the SMF record to be queued for writing.

**JMR**
Specifies that an SMF type 26 record should be queued for writing.

**MODE=**
Specifies the mode of operation.

- **A** Specifies asynchronous mode. The SMF record is copied and queued for writing. Control then returns to the caller.
- **S** Specifies synchronous mode. The SMF record is queued for writing, and an AWAIT macro is issued pending completion of the write. Return is made to the caller after the record is written.

**RQ=**
Specifies the address of the resident job queue (RESQUEUE) for the associated work. The IATXSMF processing routine uses the RESQUEUE address to determine the category for the work and to pass the category to SMF. The following table indicates how the work category is determined based on various RESQUEUE values.

<table>
<thead>
<tr>
<th>RQ=</th>
<th>Demand Select ?</th>
<th>From TSO ?</th>
<th>Called DSP ?</th>
<th>Work Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>STC</td>
</tr>
<tr>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>STC</td>
</tr>
<tr>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>TSO</td>
</tr>
<tr>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>BATCH</td>
</tr>
<tr>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>BATCH</td>
</tr>
</tbody>
</table>
If RQ=0 is specified or if the RQ parameter is omitted, then the work has no RESQUEUE associated with it (that is, no SMF records which describe JES3 activity).

**NORMAL=**

This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

**Linkage**

This macro uses ASAVE linkage.

**Restrictions**

Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task.

**Note:** This macro returns a return code from SMF. For the explanation of the SMF return code, consult [z/OS MVS System Management Facilities (SMF)](https://www.ibm.com/systems/z/os/zos/bkserv/).

### IATXSMGR (Set Up Parameter Linkage for the SNA RJP State Manager)

The IATXSMGR macro sets up parameter linkage for the SNA RJP state manager (module IATSNDM). This macro is issued whenever a first of chain or end of chain is sent or received.

The routine using this macro must set register 11 according to the SNA RJP convention. In this convention, the high-order bit of register 11 is used to indicate the type of address (in the low-order 31 bits) as follows:

- If the bit is off (0), it is an FCT address.
- If the bit is on (1), it is an IRB address or an SRB address.

To distinguish an IRB from an SRB, test the PSATOLD word in the PSA. If this word is zero, the SRB is active. If the word is not zero, the IRB is active.

**Syntax**

```plaintext
IATXSMGR LUCB=label (R1) (reg) address, MODE=DSP RB
```

**Parameters**

**LUCB=**

Specifies the address of the logical unit control block (LCB) of the session being processed.

**MODE=**

**DSP**

Specifies that the caller runs under the control of a DSP.
**IATXSMGR**

**RB**
Specifies that the caller runs under the control of an interruption request block (IRB) or service request block (SRB).

**RESP=**
Specifies whether the caller wants a positive response sent to the LU.

**Linkage**
When the MODE=DSP parameter is specified, this macro uses ASAVE linkage.
When the MODE=RB parameter is specified, this macro issues a BALR R14,R15 instruction for linkage.

**Restrictions**
Do not use this macro in any routine that might run in a C/I FSS address space.

---

**IATXSNFG (Obtain a Work Area for SNARJP Termination)**

The IATXSNFG macro obtains a work area to be used during SNARJP termination. The macro returns in register 2 the address of a SNARJP FAILDSP work area (mapped by IATYSNFS). IATXSNFG also provides a level of JESTAE ESTAE protection that will free this work area and percolate to the next level of recovery.

The routine using this macro must set register 11 according to the SNA RJP convention. In this convention, the high-order bit of register 11 is used to indicate the type of state as follows:

- If the bit is off (0), the routine is running under the FCT.
- If the bit is on (1), the routine is running under the IRB or the SRB.

To distinguish an IRB from an SRB, test the PSATOLD word in the PSA. If this word is zero, the SRB is active. If the word is not zero, the IRB is active.

**Syntax**

```
IATXSNFG label,,NORMAL=address
```

**Parameters**

**NORMAL=**
Specifies an address other than the next sequential instruction that the routine can return to after normal execution of the macro. If you omit NORMAL=, control returns to the next sequential instruction.

**Linkage**
This macro uses standard MVS entry linkage and can be called from all execution modes (DSPs, IRBs, and SRBs).

**Register Conventions to Follow**

The contents of the registers on entry must be:

**R0-1**
Not used by this macro. The contents are ignored by the SNARJP FAILDSP processor.

**R2**
Irrelevant on entry. The macro will use register 2 to store the address of the work area.
Register Values on Exit for a Normal Return

The contents of the registers on exit should be:

- **R0-1**: Not changed by the SNARJP FAILDSP processor.
- **R2**: Contains the address of the SNARJP FAILDSP work area that will be displayed in the JES3 formatted dump. This work area is used to provide additional diagnostic information on the failure.
- **R3-13**: Not changed by the SNARJP FAILDSP processor.
- **R14**: Return address.
- **R15**: Return code of zero.

Restrictions

Use this macro only in the SNARJP environment.

Recovery Considerations

The SNARJP FAILDSP work area will be released (using FREEMAIN) after the JES3 formatted dump is taken, or after the user requests that no dump be taken. Also, if the user of this work area ABENDs before issuing the IATXSNFS macro, the work area will be released. Therefore, no changes need be made to JESTAE or ESTAE exits to free this area on failure.

Example of Use

In the following example a test is made to determine if a storage area is a logical unit control block (LCB). If the control block is incorrect, a DM553 ABEND should occur. The IATXSNFG macro is used to obtain a SNARJP FAILDSP work area. Inside this work area, JES3 attempts to store any data that might be useful in solving the failure. Once the work area has been initialized, the IATXSNFS macro is issued to ABEND the caller.

```
CLC LCBCBID,=CL4'LCB' IS THIS AN LCB?
BC EQ,LABEL YES, GO CONTINUE PROCESSING
IATXSNFG USING IATYSNF5,R2 GET FAILDSP WORK AREA
LA R3,SNFSDATA POINT TO USER AREA
USING YSNFD553,R3 ESTABLISH ADDRESSABILITY TO IT
LA R4,D553LENX GET LENGTH OF INFORMATION
ST R4,D553LEN STORE IN USER AREA
ST R5,D553LCB STORE LCB ADDRESS
IATXSNFS CODE=DUMP553,DATA=(R2) ABEND THE CALLER
LABEL DS 0H
```

**IATXSNFM (Determine the Caller’s State and Release Storage)**

The IATXSNFM macro generates code that determines the state the caller is in (either DSP, IRB, or SRB), and releases storage using the appropriate method (either APUTMAIN, FREEMAIN, or branch-entry FREEMAIN, respectively).
The routine using this macro must set register 11 according to the SNA RJP convention. In this convention, the high-order bit of register 11 is used to indicate the type of state as follows:

- If the bit is off (0), the routine is running under the FCT.
- If the bit is on (1), the routine is running under the IRB or the SRB.

To distinguish an IRB from an SRB, test the PSATOLD word in the PSA. If this word is zero, the SRB is active. If the word is not zero, the IRB is active.

**Syntax**

```
IATXSNFM AREA=(R1) label,SIZE=(R0) reg
```

```
SP=number,MODE=(DSP,IRB,SRB)
```

**Parameters**

**AREA=**

Specifies the address of the area to be released. Do not use registers 4 or 7.

**SIZE=**

Specifies the number of bytes of storage to be released. Do not use registers 4 or 7.

**SP=**

Specifies the number of the subpool from which to release the storage. If SP= is omitted, subpool 13 is assumed. SP= must be specified as a constant (an integer), or in the low-order byte of the register. The other bytes may contain any values. If in register 1, the other bytes will be changed by the macro expansion.

The SP and SIZE parameters can not specify the same register. Do not use registers 4 or 7 for SP=.

**MODE=**

Specifies whether the caller can be running in DSP, IRB, or SRB modes. This parameter generates code that determines the mode at run time. You may specify one value, two values, or three values, and in any order. But you may use an individual value only one time. If you specify more than one value for this parameter, you must enclose the values in parentheses. If you specify only one value, you may omit the parentheses.

An example of MODE= is: MODE=(IRB,SRB,DSP)

**Linkage**

This macro uses standard MVS entry linkage and can be called from all execution modes (DSPs, IRBs, and SRBs).
Register Conventions to Follow

The contents of the registers on entry must be:

- R0-2 Will be used as work registers.
- R3 Unused.
- R4 Contains the TCB address used during branch-entry FREEMAIN (that is, during the macro processing). After the FREEMAIN, this register may no longer point to the TCB.
- R5-6 Unused.
- R7 Contains the ASCB address used during branch-entry FREEMAIN (that is, during the macro processing). After the FREEMAIN, this register might not point to the ASCB.
- R8-10 Unused.
- R11 See the note on SNARJP conventions above.
- R12 Must point to the TVT (IATYTVT).
- R13 Must point to a 72-byte save area.
- R14-15 Work registers.

Register Values on Exit for a Normal Return

The contents of the registers on exit should be:

- R0-2 Unpredictable
- R3 Unchanged
- R4 Unpredictable
- R5-6 Unchanged
- R7 Unpredictable
- R8-13 Unchanged
- R14-15 Unpredictable

Restrictions

Use this macro only in the SNARJP environment.

Recovery Considerations

This macro expands inline. There is no recovery. The calling routine must provide for proper recovery.

Example of use

Assume a temporary work area was obtained and is pointed to by register 7. The size of the area is stored in a field named TEMPSIZE. JES3 can only be running in DSP or IRB mode.

LTR R7 R7 SHOULD AREA BE FREEMAINED?
BC ZERO, LABEL NO, CONTINUE
IATXSNFM AREA=(R7), SIZE=TEMPSIZE, MODE=(DSP, IRB)

IATXSNFS (Interface to the SNARJP FAILDSP Processor)

The IATXSNFS macro allows you to indicate that a specified DSP should be failed, with a failure code, upon recognition of an error.

If you specify the constant form of the CODE=parameter, this macro requires that JES3 macro IATYDUM be expanded in the module in which IATXSNFS is coded. If IATYDUM is not coded, an assembly error will occur.

The routine using this macro must set register 11 according to the SNA RJP convention. In this convention, the high-order bit of register 11 is used to indicate the type of state as follows:
IATXSNSF

- If the bit is off (0), it is running under the FCT.
- If the bit is on (1), it is running under the IRB or the SRB.

To distinguish an IRB from an SRB, test the PSATOLD word in the PSA. If this word is zero, the SRB is active. If the word is not zero, the IRB is active.

**Syntax**

```
\text{IATXSNSF} \text{CODE} \leftarrow \text{DUMP} \leftarrow \text{DATA} \leftarrow (\text{R1}) \leftarrow (\text{R2})
```

```
\text{FCT} \leftarrow (\text{R0}) \leftarrow \text{DUMP} \leftarrow \text{YES} \leftarrow \text{NO} \leftarrow (\text{R2})
```

**Parameters**

**CODE=**
Specifies the failure code. If the DUMPnnn parameter is specified, the IATYDUM macro must be coded elsewhere in this module.

**DATA=**
Specifies the address of the SNARJP FAILDSP work area that will contain diagnostic information about the failure. The work area pointed to by this parameter is obtained beforehand using the IATXSNFG macro.

**FCT=**
Specifies the address of the FCT to be failed. If omitted, the active FCT is assumed, or the SNARJP FCT if the caller is running under an IRB or SRB.

**DUMP=**
Specifies whether a JES3 abend dump is to be taken. If the register form is used, it must be the same register as coded for the FCT parameter. If the content of the register is positive, a dump will be taken. If the content is negative, JES3 assumes the register contains the complement of the FCT address, and no dump will be taken.

**NORMAL=**
Specifies an address other than the next sequential instruction that the routine can return to after normal execution of the macro. If you omit NORMAL=, control returns to the next sequential instruction.

**Linkage**

This macro uses standard MVS entry linkage and can be called from all execution modes (DSPs, IRBs, and SRBs).

**Register Conventions to Follow**

The contents of the registers on entry must be:

- **R0**
  Contains the FCT address for the DSP to be failed. If you are not running under an FCT, the macro will zero-out the register.

- **R1**
  Will be used to contain the ABEND code.

- **R2**
  Contains the address of the SNARJP FAILDSP work area that will
be displayed in the JES3 formatted dump. This work area is used to provide additional diagnostic information on the failure.

**Register Values on Exit for a Normal Return**

The contents of the registers on exit should be:

<table>
<thead>
<tr>
<th>Register</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>R0</td>
<td>Contains the FCT address for the DSP to be failed. If you are not running under an FCT, the macro will zero-out the register.</td>
</tr>
<tr>
<td>R1</td>
<td>Contains the ABEND code to be assigned to the failing DSP.</td>
</tr>
<tr>
<td>R2-13</td>
<td>Same as on entry</td>
</tr>
<tr>
<td>R14</td>
<td>Return address</td>
</tr>
<tr>
<td>R15</td>
<td>Return code of zero</td>
</tr>
</tbody>
</table>

**Restrictions**

Use this macro only in the SNARJP environment.

**Recovery Considerations**

The macro service routine provides a level of recovery. The type of recovery depends on the mode of the caller (JESTAE for MODE=DSP, ESTAE for MODE=IRB, or SETFRR for MODE=SRB). These recovery routines use FREEMAIN to release the FAILDSP work area, then they percolate to the caller's level of recovery.

**Example of Use**

In the following example a test is made to determine if a storage area is a logical unit control block (LCB). If the control block is incorrect, a DM553 ABEND should occur. The IATXSNFG macro is used to obtain a FAILDSP work area. Inside this work area, JES3 attempts to store any data that might be useful in solving the failure. Once the work area has been initialized, the IATXSNFS macro is issued to ABEND the caller.

```
CLC   LCBCBID,=CL4'LCB'   IS THIS AN LCB?
BC    EQ, LABEL          YES, GO CONTINUE PROCESSING
IATXSNFG USING IATYSNFS,R2 GET FAILDSP WORK AREA
USING YSNFS0553,R3       ESTABLISH ADDRESSABILITY TO IT
LA    R3,SNFSDATA        POINT TO USER AREA
USING YSNFS0553,R3       ESTABLISH ADDRESSABILITY TO IT
LA    R4,D553LENX        GET LENGTH OF INFORMATION
ST    R4,D553LEN         STORE IN USER AREA
ST    R5,D553LCB         STORE LCB ADDRESS
IATXSNFS CODE=DUMP553,DATA=(R2) ABEND THE CALLER

LABEL  DS 0H
```
IATXSNGM

IATXSNGM (Determine Caller’s State and Obtain Storage)

The IATXSNGM macro is an inline macro that generates code to determine what state the caller is in (DSP, IRB, or SRB) at run time, and also obtain storage using the appropriate method (either AGETMAIN, GETMAIN, and branch-entry GETMAIN, respectively).

The routine using this macro must set register 11 according to the SNA RJP convention. In this convention, the high-order bit of register 11 is used to indicate the type of state as follows:

- If the bit is off (0), it is running under the FCT.
- If the bit is on (1), it is running under the IRB or the SRB.

To distinguish an IRB from an SRB, test the PSATOLD word in the PSA. If this word is zero, the SRB is active. If the word is not zero, the IRB is active.

Syntax

```
/IATXSNGM SIZE=(R0) (reg) address, LOC=BELOW

/IATXSNGM SP=(R1) (reg) number, MODE=(DSP, IRB, SRB)

/IATXSNGM, DBLWD, BNDRY=PAGE
```

Parameters

SIZE=

Specifies the number of bytes of storage to obtain. If the address form is specified, it must be the address of a fullword containing a number that specifies the number of bytes required. If you use register 0, the number of bytes must be in the low-order three bytes of the register.

The SIZE and SP parameters can not specify the same register.

For storage obtained in virtual storage above 16 megabytes, the size value specified may be greater than 16 megabytes. The amount available is determined by the REGION parameter on the JCL used to start JES3 or by the default set by the IEFUSI exit.

Note: Do not use registers 4 or 7 for SIZE=.

LOC=

Defines the residence requirements of the storage being obtained, using the same manner as LOC= on the MVS GETMAIN macro. This is as follows:
BELOW
Obtain storage in virtual storage from below 16 megabytes. Pages are backed below 16 megabytes central when fixed.

(BELOW,ANY)
The same as BELOW except that pages may be backed anywhere in central when fixed.

ANY
Obtain storage from anywhere in virtual storage (normally above 16 megabytes). Pages may be backed anywhere when fixed.

(ANY,ANY)
The same as ANY.

RES
Obtain storage from the same residence as the calling program. If the calling program resides in storage below 16 megabytes, the storage is obtained below 16 megabytes virtual, and central storage is backed below 16 megabytes central when fixed. If the calling program resides in virtual storage above 16 megabytes, the storage is obtained from anywhere in virtual storage and is backed anywhere in central storage when fixed.

(RES,ANY)
The same as RES except storage may be backed anywhere in central storage when fixed regardless of the residence of the calling program.

SP=
Specifies the number of the subpool from which to get the storage. If SP= is omitted, subpool 13 is assumed. If SP= is specified as a register, the subpool number must be in the low-order byte of the register. The other bytes may contain any values. If in register 1, the other bytes will be changed by the macro expansion.

The SP and SIZE parameters can not specify the same register. Do not use registers 4 or 7 for SP=.

MODE=
Specifies whether the caller can be running in DSP, IRB, or SRB modes. This parameter generates code that determines the mode at run time. You may specify one value, two values, or three values, and in any order. But you may use an individual value only one time. If you specify more than one value for this parameter, you must enclose the values in parentheses. If you specify only one value, you may omit the parentheses.

An example of MODE= is: MODE=(IRB,SRB,DSP)

BNDRY=

DBLWD
Specifies that the obtained storage is to be on a doubleword boundary.

PAGE
Specifies that the obtained storage is to be on a page boundary.

Linkage
This macro uses standard MVS entry linkage and can be called from all execution modes (DSPs, IRBs, and SRBs).

Register Conventions to Follow
The contents of the registers on entry must be:
IATXSNGM

R0-2  Work registers.
R3    Unused.
R4    Contains the TCB address used during branch-entry GETMAIN.
R5-6  Unused.
R7    Contains the ASCB address used during branch-entry GETMAIN.
R8-10 Unused.
R11   See the note on SNARJP conventions above.
R12   Must point to the TVT (IATYTVT).
R13   Must point to a 72-byte save area.
R14-15 Work registers.

Register Values on Exit for a Normal Return

The contents of the registers on exit should be:

R0    Unpredictable
R1    Pointer to the storage area obtained
R2    Unpredictable
R3    Unchanged
R4    Unpredictable
R5-6  Unchanged
R7    Unpredictable
R8-13 Unchanged
R14-15 Unpredictable

Restrictions

Use this macro only in the SNARJP environment.

Recovery Considerations

This macro expands inline. There is no recovery. The calling routine must provide for proper recovery.

Example of use

JES3 needs to obtain a temporary work area. JES3 can only be running in IRB or SRB mode.

LA R4,WORKLEN GET THE WORK AREA LENGTH
IATXSNGM SIZE=(R3),LOC=ANY,MODE=(IRB,SRB)
USING TEMPWORK,R1

IATXSNLK (Increase or Decrease the LCB Use Count)

You can use the IATXSNLK macro to increase or decrease the use count in the LCB. The LCB use count must be incremented before an LCB is used under a scheduling element. When work using the LCB is complete, the use count must be decremented. The LCB cannot be FREEMAINed until the use count is down to zero.

The routine using this macro must set register 11 according to the SNARJP convention. In this convention, the high-order bit of register 11 is used to indicate the type of state as follows:

- If the bit is off (0), it is running under the FCT.
- If the bit is on (1), it is running under the IRB or the SRB.

To distinguish an IRB from an SRB, test the PSATOLD word in the PSA. If this word is zero, the SRB is active. If the word is not zero, the IRB is active.
For SRB callers register 11 bytes 1 through 3 must contain zeros for the first use of the IATXSNLK macro. Upon returning to the caller, register 11, bytes 1 through 3, will contain a token that must be in register 11 for subsequent uses. This token must be used for the entire path of this SRB.

Syntax

```
IATXSNLK OPTION=INCREMNT,TERM=address

Lrove=

NORMAL=ERROR=
```

Parameters

**OPTION=**
Specifies the operation to be performed. DECALL should only be used in recovery routines. It will remove all footprint entries containing the token that is in register 11, bytes 1 through 3, and decrement the LCB use count by the number of footprint entries that were removed. This parameter is required.

**INCREMNT**
Specifies that the use count manager is to increase the LCB use count by one.

**DECREMNT**
Specifies that the use count manager is to decrease the LCB use count by one. The LCB will not be freed until the use count is lowered to zero.

**LCB=**
Specifies the address of the LCB. This parameter is required.

**RELATED=**
Specifies the routine that is incrementing or decrementing the LCB use count. This parameter is required.
ERRCHKDS
The routine was the error check routine running under a DSP. The routine is located in module IATSNDA.

ERRCHKRB
The routine was the error check routine running under an IRB or SRB. The routine is located in module IATSNDA.

WSCLOSE
The workstation close routine in module IATSNDC.

RDOPEN
The workstation open routine for readers in module IATSNDO.

CONOPENI
The workstation open routine for inbound consoles in module IATSNDO.

CONOPENO
The workstation open routine for outbound consoles in module IATSNDO.

CONOPENS
The workstation open routine for simulated consoles in module IATSNDO.

WTOPEN
The workstation open routine for a writer in module IATSNDO.

SEARCH
The search-for-free LCB routine in module IATSNDO.

RESPONSE
The response exit in module IATSNDR.

SENDEXIT
The send exit in module IATSNDS.

FNTMANIN
The functional management inbound routine in module IATSNFI.

FNTMANOT
The functional management outbound routine in module IATSNFO.

IATNUC
The SNARJP FCT is set by IATSNLB when the LCB is created, and is removed by IATSNRS when the LCB is being freed.

OPNDST
The OPNDST exit in module IATSNLS.

CLSDST
The CLSDST exit in module IATSNLS.

LOSTTERM
The LOSTTERM exit in module IATSNLS.

INQUIRE
The workstation inquiry command in module IATIQRJ.

CNCLWS
The workstation CANCEL command processor in module IATSNLC.

(RX)
The equivalent hexadecimal value for the related field can be contained in byte three of registers 1 through 15.
NORMAL=
Specifies for a normal return, an address to return to other than the instruction following the macro.

ERROR=
Specifies an address to return to if an error occurs. When control is passed to this address, register 0 contains a dump code and register 15 contains a return code.

TERM=
Specifies an address to return to if the session is terminating when the IATXSNLK macro is issued. This parameter is required if OPTION=INCREMNT is specified. If any other value is specified on OPTION=, TERM= is not allowed.

Linkage
This macro uses standard MVS entry linkage and can be called from all execution modes (DSPs, IRBs, and SRBs).

Register Conventions to Follow
The contents of the registers on entry must be:
R0  Contains the operation to be performed.
R1  Contains the LCB address.
R2-10 Not used by this macro. The contents are ignored by the termination status manager.
R11 See the note on SNARJP conventions above.
R12 Must point to the TVT (IATYTVT).
R13 Must point to a 72-byte save area.
R14 Return address to be used by the termination status manager.
R15 Address of the termination status manager.

Register Values on Exit for a Normal Return
The contents of the registers on exit should be:
R0  Byte 3 contains the related token value
R1-13 Not changed by the use count manager
R14 Return address
R15 Return code of zero

Register Values on Exit for an Error Return
The contents of the registers on exit should be:
R0  Contains the dump code to be used if the calling routine cannot recover.
R1-13 Not changed by the use count manager
R14 Return address
R15 Return code used to determine the type of error. Valid return codes and dump codes (returned in registers 15 and 0, respectively, are:

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Dump Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>552</td>
<td>LCB address was incorrect</td>
</tr>
<tr>
<td>08</td>
<td>552</td>
<td>IATXSNLK ABEND occurred</td>
</tr>
</tbody>
</table>
IATXSNLK

Restrictions

Use this macro only in the SNARJP environment.

Recovery Considerations

The macro service routine provides a level of recovery. If no error exit is specified, the service routine will ABEND with a dump code of DM552. The recovery routines in IATXSNLK will clean up any storage obtained and percolate to the next level of recovery. If an error exit was specified on the macro, a return code and dump code are returned to the caller who should attempt to recover.

Example of Use

Assume VTAM schedules an exit as an SRB. If an LCB is used in this exit, the IATXSNLK macro must be used to increment the LCB use count before the control block is accessed. It must also be used to decrease the use count after using the control block.

```
L    R7,DVELCB   GET ADDRESS OF LCB
IATXSNLK OPTION=INCREMNT,LCB=(R7)
    perform work on LCB
IATXSNLK OPTION=DECREMNT,LCB=(R7)
```

IATXSNM (Send a SNA RJP Message)

This macro sends a SNA RJP message to a remote workstation.

For a summary of JES3 macros involved in message processing, see the section "Choosing the Appropriate Macro to Issue a Message" on page 42

Syntax

```
IATXSNM label MSG=(reg)address-,MODE=DSPSRBIRB-

,VAR=(reg),(reg),PARMA=(reg)address-

,CNDB=(reg)address-
```

Parameters

MSG=

Specifies the address of the message to be sent.
MODE=
Specifies whether the caller is running under a DSP, a service request block (SRB), or an interruption request block (IRB). If DSP is specified, the IATYFCT DSECT must be defined.

VAR=
Specifies the addresses of the message parameters. The addresses specified here must be in the same order that the parameters appear in the message.

PARMA=
Specifies the address of a 36-byte parameter work area. The PARMA parameter is applicable only when MODE=SRB is specified; and is required if MODE=SRB is specified. If the caller is an SRB, then the caller must obtain 36 bytes of work area and specify the work area's address in this parameter. In the case of a DSP or IRB user, any address specified by the PARMA parameter is ignored. If the caller is a DSP operating in NUCTASK mode, the macro uses the SRDDMPR work area in the SNA RJP data CSECT (IATYSRD). If the caller is a DSP operating in AUXTASK mode, the macro uses the SRDAMPR work area in the SNA data CSECT. If the caller is an IRB, the macro uses the SRDIMPR work area in the SNA RJP data CSECT.

CNDB=
Specifies the address of a Console Destination Block (IATYCNDB) which contains destination information for the message.

This parameter can be specified as an address or as a register. If the register form is used, the register must contain the address of a CNDB area mapped by IATYCNDB.

Linkage
If the macro caller runs under a DSP, the macro uses ASAVE linkage. If not, the macro issues a BALR R14,R15 for linkage.

Restrictions
Do not use this macro in any routine that might run in a C/I FSS address space.

IATXSNST (Set Up Interface to SNARJP Termination Status Manager)
You can use the IATXSNST macro to provide an interface to the SNARJP termination status manager. This interface will control access to the SNARJP termination status mappings, and also will provide a standard interface for setting and testing the various mappings.

The routine using this macro must set register 11 according to the SNA RJP convention. In this convention, the high-order bit of register 11 is used to indicate the type of state as follows:
• If the bit is off (0), it is running under the FCT.
• If the bit is on (1), it is running under the IRB or the SRB.

To distinguish an IRB from an SRB, test the PSATOLD word in the PSA. If this word is zero, the SRB is active. If the word is not zero, the IRB is active.

Syntax
Parameters

**OPTION=**
Specifies the operation to be performed.

**SETON**
Specifies that the given state mapping is to be turned on.

**SETOFF**
Specifies that the given state mapping is to be turned off.

**TEST**
Causes the given state mapping bit to be tested, and the results of the test returned in register 15. Return code 0 means the bit was off. Return code 4 means the bit was on.

**STATE=**
Specifies the state mapping bit on which to perform a given operation.

**QI** Quiesce immediate processing should occur.

**Q** Quiesce processing should occur.

**CLSDSTRQ**
CLSDST processing has been requested.

**CLSDSTCM**
CLSDST processing is complete.

**FREECBLK**
Control block will be freed when the LCB use count is reduced to zero.

**LCB=**
Specifies the address of the logical control block.

**NORMAL=**
Specifies an address other than the next sequential instruction that the routine can return to after normal execution of the macro. If you omit NORMAL=, control returns to the next sequential instruction.

**ERROR=**
Specifies an address to return to if the termination manager detects an error. If you omit the ERROR= parameter, the termination status manager will issue the ABEND under the active FCT or the SNARJP FCT if the request was made from under an IRB or SRB.

Register 15 will contain a return code, and register 0 will contain a dump code.

Linkage

This macro uses standard MVS entry linkage and can be called from all execution modes (DSPs, IRBs, and SRBs).
Register Conventions to Follow

The contents of the registers on entry must be:
- R0: Contains the operation to be performed.
- R1: Contains state mapping information.
- R2: Contains the LCB address.
- R3-10: Not used by this macro. The contents are ignored by the termination status manager.
- R11: See the note on SNARJP conventions above.
- R12: Must point to the TVT (IATYTVT).
- R13: Must point to a 72-byte save area.
- R14: Return address to be used by the termination status manager.
- R15: Address of the termination status manager.

Register Values on Exit for a Normal Return

The contents of the registers on exit should be:
- R0-13: Not changed by the termination status manager
- R14: Return address
- R15: Return code used to determine the status of the bit. If OPTION=TEST is specified, a return code of 4 is set if the tested bit is on, otherwise the return code is 0. If OPTION=SETON, a return code of 4 is returned if the bit being set on is on when the compare-and-swap is done; otherwise the return code is 0.

Register Values on Exit for an Error Return

The contents of the registers on exit should be:
- R0: Contains the dump code to be used if the calling routine cannot recover.
- R1-13: Not changed by the termination status manager
- R14: Return address
- R15: Return code used to determine the type of error. Valid return codes and dump codes (returned in registers 15 and 0, respectively, are:

<table>
<thead>
<tr>
<th>Return Code in Hex</th>
<th>Dump Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>553</td>
<td>LCB address was incorrect</td>
</tr>
<tr>
<td>04</td>
<td>553</td>
<td>LCB use count was zero</td>
</tr>
<tr>
<td>08</td>
<td>553</td>
<td>IATSNLK ABEND occurred</td>
</tr>
</tbody>
</table>

Restrictions

Use this macro only in the SNARJP environment.

Recovery Considerations

The macro service routine provides a level of recovery. If no error exit is specified, the service routine will ABEND with a dump code of DM553. The recovery routines in IATXSNST will clean up any storage obtained and percolate to the next level of recovery. If an error exit was specified on the macro, a return code and dump code are returned to the caller who should attempt to recover.
IATXSNST

Example of Use

Assume VTAM schedules an exit as an SRB. If an LCB is used in this exit, the IATXSNLK macro must be used to increment the LCB use count before the control block is accessed. It must also be used to decrease the use count after using the control block.

```assembly
L R7,DVELCB GET ADDRESS OF LCB
IATXSNLK OPTION=INCREMNT,LCB=(R7)
IATXSNST OPTION=TEST,STATE=QI,LCB=(R7)
LTR R15,R15 ARE WE QUIESCING?
BC ZERO,LABEL NO, CONTINUE
IATXSNLK OPTION=DECREMNT,LCB=(R7)
```

IATXSNTR (Create an Entry in the SNA RJP Trace Table)

The IATXSNTR macro creates an entry in the SNA RJP trace table.

**Syntax**

```
<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IATXSNTR LCB= (reg), ID= (module, subpath)</td>
<td></td>
</tr>
<tr>
<td>TYPE= PATH</td>
<td></td>
</tr>
<tr>
<td>ABND DATA</td>
<td></td>
</tr>
<tr>
<td>DFC,BFE (reg)</td>
<td></td>
</tr>
<tr>
<td>RESP,RPL (reg)</td>
<td></td>
</tr>
<tr>
<td>ERR,RPL (reg)</td>
<td></td>
</tr>
<tr>
<td>DATA:</td>
<td></td>
</tr>
<tr>
<td>DATA= (reg), DLATALEN= 1, 2, 3, 4</td>
<td></td>
</tr>
</tbody>
</table>
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LCB=</td>
<td>Specifies the address of the session control block.</td>
</tr>
<tr>
<td>ID=</td>
<td>Indicates a 4-character field consisting of the last two characters of the name of the module issuing the macro followed by a number from 0-255 identifying the subpath within the module.</td>
</tr>
<tr>
<td>TYPE=</td>
<td>Specifies the type of trace to be taken.</td>
</tr>
<tr>
<td>PATH</td>
<td>Indicates that the module and subpath are to be traced.</td>
</tr>
</tbody>
</table>
**ABND**
Indicates that an X'ABF' abnormal termination or a DM551 has occurred.

**DFC,BFE=**
Indicates that the data flow control (DFC) component is to be traced. See [z/OS JES3 Diagnosis](#) Do not use the DATA parameter with this parameter.

**RESP**
Indicates that four bytes of negative data are to be traced.

**ERR,RPL=**
Specifies the address of the request parameter list (RPL) used to send or receive the negative response.

**FLOW=**
Specifies whether the response was sent (IN) or received (OUT).

**DATA=**
Specifies the register that contains the data to be traced. Do not use this parameter when TYPE=DFC is specified. You must specify the DATLEN parameter with this parameter.

**DATLEN=**
Indicates, starting with the high-order byte of the register, how many bytes are to be traced. You must specify this parameter when the DATA parameter is specified.

For TYPE=ERR, the value of DATLEN cannot exceed 3. For TYPE=RESP, the value of DATLEN cannot exceed 2.

**Linkage**
This macro generates inline code.

**Restrictions**
Do not use this macro in any routine that might run in a C/I FSS address space.

**IATXSPCK (Checkpoint Status of Spool Data Sets)**
The IATXSPCK macro checkpoints the status of spool data sets and partitions. Register 1 is used as a parameter register.

**Syntax**

```
/IATXSPCK
label TYPE=PART,BUSY=address,NORMAL=address
```

**Parameters**

**TYPE=**

**EXT**
Indicates that the status of the spool data sets is to be checkpointed.

**PART**
Indicates that the spool partition configuration status is to be checkpointed.

**BUSY=**
Specifies the address to which control returns if the request cannot be satisfied
because the checkpoint record access method is being used by another JES3 function. An ECF mask and address are returned which may be used to wait (using the AWAIT TYPE=OFF macro) for the checkpoint record access method to become available. The 31-bit ECF address is returned in register 0. The ECF mask needed is contained in the low-order byte of register 1, with the other three bytes containing zeros.

If this parameter is omitted and the checkpoint access method is busy, the spool checkpoint service waits.

**NORMAL=**

This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

**Linkage**

This macro uses ASAVE linkage.

**Restrictions**

Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task.

**IATXSPR (Enqueue SETPRT Requests)**

The IATXSPR macro enqueues SETPRT requests for execution by the JES3 SETPRT subtask.

**Syntax**

```
IATXSPR label PARM=(R1) address,TYPE=ADD
```

**Parameters**

**PARM=**

Specifies the address of the SETPRT parameter list to be queued for execution. (The area is mapped by the IATYSPP macro.)

**TYPE=**

Specifies the type of action to be taken.

**ADD**

Specifies that a new request is to be queued.

**DEL**

Specifies that a previously queued request is to be deleted.

**NORMAL=**

This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.
Linkage

This macro uses ASAVE linkage.

Restrictions

Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task.

IATXSQE (Add or Delete Entry from Storage Management Queue)

The IATXSQE macro adds or deletes an entry from the JES3 storage management queue. The entries are called storage queue elements (SQEs). The storage management queue establishes accountability for JES3 use of common storage, auxiliary address space storage, and important JES3 private storage. JES3 also uses the queue to dump these storage areas.

This macro may also be used to add a data area to the JES3 dump list. This list is used by various JES3 recovery routines to reduce the amount of data produced for dump requests.

Syntax

```
IATXSQE label LOG=ADD,SIZE=(R0),AREA=(R1),DUMP=YES, HDR='cccc', TVT, TYPE=SVT, SP=ITK, AUX=Y, DULADD=NO, SP=:
```

Parameters

LOG=

- **ADD**
  Specifies that an entry is to be added to the storage management queue.

- **DELETE**
  Specifies that an entry is to be deleted from the queue.

SIZE=

Specifies the number of bytes of storage obtained. If the address form is specified, it must be the address of a fullword containing the number of bytes required. If you use a register, the number of bytes must be in the low-order three bytes of the register.
You may not specify register 1, register 2, or a register specified for the SP= or AREA= parameters.

AREA=
Specifies the address of the area obtained. If a register is used, the address is in the register. If an address is used, it must be the address of a word containing the area address.

You may not specify register 0, register 2, or a register specified for the SP= or SIZE= parameters.

SP=
Specifies the subpool number where the storage was obtained. If SP= is specified as a register, the subpool number must be in the low-order byte of the register. If not register 2, the other bytes may contain any values, and they will not be changed when this macro expands.

If TYPE= does not specify TVT, then SP= must be coded.

If you do not specify SP=, JESPOOL is assumed. You may not specify register 0, register 1, or a register specified for the SIZE= or AREA= parameters.

DUMP=
    YES
    Specifies that the JES3 abend formatting routines will include the storage area in the JES3-formatted dump.
    NO
    Specifies that the storage area will not be included in the dump.

HDR=
Specifies a 1- to 20-byte character string defining the content of the storage area. The characters must be enclosed in apostrophes.

TYPE=
Specifies the control block from which the storage management routine’s entry point is to be obtained.

AUX=
    N
    Indicates that the storage being logged was obtained from the JES3 address space, or from CSA.
    Y
    Indicates that the storage being logged was obtained from the auxiliary address space.

DULADD=
    NO
    Specifies that the storage area will not be added to the dump list.
    YES
    Specifies that the storage area being logged is to be added to the dump list for use by recovery routines during dump processing.

If a LOG=DELETE request is issued for a storage area in the dump list, the entry on the dump list is also deleted.

Linkage
This macro issues a BALR R14,R15 for linkage.

Restrictions
This macro destroys the contents of register 2.
IATXSRS (Post/Test DSPs Waiting for Specialized Rescheduling)

The IATXSRS macro posts DSPs that are waiting for specialized device rescheduling or tests for DSPs requiring devices of the same class or type.

Syntax

```plaintext
label IATXSRS TYPE=

TEST, SUPUNIT=(reg), NAVAIL=address

POST

NORMAL=address
```

Parameters

**TYPE=**

**TEST**

Indicates that DSPs of a higher priority than the caller are tested to determine if any of them require a device of the same type or if the main device scheduler (MDS) requires a device of the same class.

**POST**

Indicates that any DSPs that are waiting for devices of the type indicated in the TVT (field UAVLFFG) are to be posted.

**SUPUNIT=**

Specifies the SUPUNITs table entry for the device type and class to be tested.

**NAVAIL=**

Specifies the return address if the device type and class (specified by using the SUPUNIT parameter) is not available.

**NORMAL=**

This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

Linkage

This macro issues the IATZCALL macro specifying SAVE=NO, which generates a BALR R14,R15. However, registers 0 and 2 through 14 are saved.

Restrictions

Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task.

IATXSTAT (Gather JES3 Statistics)

You use the IATXSTAT macro to collect statistics associated with JES3. The statistics are put in a control block called the statistics data area (SDA or IATYSDA).
The SDA consists of a header and multiple function specific entries. The SDA header resides in CSA. The SDA entries can reside in either CSA or the JES3 private area.

Each SDA entry contains a fixed section that is common to all SDA entries and a function specific variable section. The fixed portion of the SDA entry contains information, such as a description of the data, the total length of the SDA entry, and a version number. To add additional function specific information, you need to:

- Add a pointer to the function specific SDA entry in the SDA header portion of IATYSAD (e.g. initialization related information is pointed to by SDAHINIT).
- Create a new mapping macro to contain the function specific information. For example, the initialization specific information is kept in IATYSAD1 and the connect specific information is kept in IATYSAD2.
- Update macro IATYSAD with the following information about the function you are adding:
  - The name of the macro which maps the function specific data
  - The field in the SDA header which points to the data
  - A description of the function specific data
- Recompile the module IATINSV. This module creates the statistics data area. This module will then run and create the SDA header and SDA entries, causing the new function specific SDA entry to be created automatically.
- Add the appropriate IATXSTAT calls to modules to collect the information for about new function.

Syntax

```
IATXSTAT

The SDA consists of a header and multiple function specific entries. The SDA header resides in CSA. The SDA entries can reside in either CSA or the JES3 private area.

Each SDA entry contains a fixed section that is common to all SDA entries and a function specific variable section. The fixed portion of the SDA entry contains information, such as a description of the data, the total length of the SDA entry, and a version number. To add additional function specific information, you need to:

- Add a pointer to the function specific SDA entry in the SDA header portion of IATYSAD (e.g. initialization related information is pointed to by SDAHINIT).
- Create a new mapping macro to contain the function specific information. For example, the initialization specific information is kept in IATYSAD1 and the connect specific information is kept in IATYSAD2.
- Update macro IATYSAD with the following information about the function you are adding:
  - The name of the macro which maps the function specific data
  - The field in the SDA header which points to the data
  - A description of the function specific data
- Recompile the module IATINSV. This module creates the statistics data area. This module will then run and create the SDA header and SDA entries, causing the new function specific SDA entry to be created automatically.
- Add the appropriate IATXSTAT calls to modules to collect the information for about new function.

Syntax

```

Parameters

**FUNC=**

Use this keyword to specify the particular function you want performed. The choices are:
CONNECT
Use this function to connect a function specific SDA entry to the SDA. You
use this function to add a data area to the SDA dynamically (The function
specific SDA entry is not created during JES3 initialization.).

TIMESTAMP
Use this function to set a time stamp in the SDA.

RESET
Use this function to reset the specified fields in the function specific SDA
entry.

INC
Use this function to increase the count in the specified fields of the SDA.
Multiple fields can be increased by specifying multiple fields in the DATA
parameter. For example,
IATXSTAT FUNC=INC,DATA=(field1,field2,...,fieldn)

DEC
Use this function to decrease the count in specified fields of the SDA entry.
Multiple fields can be decreased by specifying multiple fields in the DATA
parameter. For example,
IATXSTAT FUNC=INC,DATA=(field1,field2,...,fieldn)

ADD
Use this function to add information to the specified field in the function
specific SDA entry. The FROM parameter specifies the data to be added to
the field. The FROM= value must be specified as a register or fullword field.

SUBTRACT
Use this function to subtract information from the specified field in the
function specific SDA entry. The FROM parameter specifies the data to be
subtracted from the field. The FROM= value must be specified as a register
or fullword field.

COPY
Use this function to copy the data specified by the FROM= keyword to the
specified field in the function specific SDA entry. You use this function to
copy a field directly to the SDA without concern for the fields contents. This
is useful when you want to copy status or option flags, such as the JES3
start type.

MAX
Use this function to determine if the value specified in the FROM= keyword
value is the maximum value to occur thus far. If the new value is the
maximum value so far, it is placed in the function specific SDA entry.

EXTRACT_VERSION
Use this function to extract the version number associated with the function
specific SDA entry. The version number is returned in register 15.

EXTRACT_DATA_PTR
Use this function to extract the address and length of the requested data
field. The address and length of the data are returned in registers 1 and 15
respectively.

DATATYPE
Use this keyword to specify the type of data being processed. For example,
DATATYPE=INIT indicates that you want to process initialization-related data.
The choices for DATATYPE= are:

INIT Initialization related information.
**RESTART**  Restart/Connect related information.

**OUTSERV**  Output service related information.

**USER1**  Reserved for user.

**USER2**  Reserved for user.

**USER3**  Reserved for user.

**USER4**  Reserved for user.

**USER5**  Reserved for user.

**DATA**  Use this keyword to specify the name of the data fields in the SDA entry to be processed. For example, if you are increasing a counter you use the DATA keyword to specify the name of the counter. If permitted by the function, multiple data fields can be specified on the DATA keyword, such as DATA=(a,b,c,d).

The name specified on the DATATYPE parameter in conjunction with the name specified on the DATA parameter is used to generate the name of the field to be processed. For example, if DATATYPE=INIT and DATA=START_TIME is specified, the name of the field to be processed will be "SDA_INIT_START_TIME".

**Note:** All fields within the function specific SDA entry must have this format.

**POINTER**  Use this keyword to specify the address of another data area that contains the statistical data. This is useful when the size of the data area needs to be determined dynamically. For the most part, the data to be accessed by the IATXSTAT macro is within the SDA entry itself. However, you can anchor other data areas from the SDA entry and use the IATXSTAT macro to extract/update data in those data areas as well. The POINTER parameter allows you to access data in a data area that is directly or indirectly addressed from the SDA entry.

If you want to increase the count in the field, SDA_INIT_CNTL, you issue the following IATXSTAT request.

<table>
<thead>
<tr>
<th>Increasing the Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>IATXSTAT FUNC=INC,</td>
</tr>
<tr>
<td>DATATYPE=INIT,</td>
</tr>
<tr>
<td>DATA=CNTS,</td>
</tr>
<tr>
<td>INDEX=(R2),</td>
</tr>
<tr>
<td>POINTER=(SDA_INIT_PTR,SDA_DATA)</td>
</tr>
</tbody>
</table>

The first subparameter of the POINTER parameter specifies the name of the SDA entry field that contains the address of the requested data area, SDA_INIT_PTR. The second subparameter specifies the DSECT name that is used to map the data when you obtain the address of SDA_DATA. With the address of SDA_DATA, the count in the SDA_INIT_CNTL field can be increased after it is indexed using the contents of register 2.

Multiple levels of indirection are allowed by specifying additional pointer/DSECT pairs on the POINTER parameter.

**FROM**  Use this keyword to specify the location from which to get data. For example, if
you are adding a value to a field in an SDA entry, the name of the field that contains the value is specified with FROM=.

**SERIALIZE**

Use this keyword to specify whether the field in the SDA entry is to be serialized through compare and swap. Specify SERIALIZE=YES if the field can be updated by multiple tasks concurrently.

**INDEX**

Use this keyword to specify an index value to be used as a displacement into the SDA entry. The INDEX keyword is useful when multiple entities are being tracked by the same field. For example, if the SDA contains a "connect started" time stamp for each main processor in the JES3 complex, then the time stamp field can be defined such that there is one double word for each main processor:

```
SDA_CONNECT_START_TIME DC (J3MAXMP)D'0'
```

The main processor sequence number (MPSEQNO) can then be specified as the index value. This will cause the time stamp to be associated with the particular main processor.

The INDEX parameter has two subparameters:

- The first subparameter specifies the value to be used to index into the SDA field. It can be specified as a register, a one byte field, a halfword field, or a fullword field.
- The second subparameter specifies whether the index is a relative index (starts at zero) or is not. If RELATIVE or REL is specified, then the index is a relative index. Otherwise, the index value that is specified is decreased by one to create a relative index.

**ENVIRON**

Use this keyword to specify what environment the caller is in. This is used to determine where to obtain storage for the SDA. If the environment is J3MAINTASK or J3SUBTASK, the SDA address is obtained from the TVT. Otherwise, the SDA address is obtained from the SSVT. If the ENVIRON parameter is not specified, the environment that was specified on the IATYMOD macro is used.

**COND**

Use this keyword to specify whether the function to be performed is to be conditionally or unconditionally performed. Specify COND=YES for the TIMESTAMP function. This indicates that the time stamp will be set in the SDA entry only if it was not set previously.

**NAVAIL**

Use this keyword to specify the address in your module where control should be passed if the SDA header or the specified SDA entry is not available for use. If NAVAIL is not specified, JES3 assumes that the SDA header and SDA entry are available for use.

**Linkage**

This macro expands in line.

**Restrictions**

None.
IATXSTAT

Examples of Use

The following are examples showing how you would use the IATXSTAT macro to perform JES3 statistical gathering functions.

---- Set the TIMESTAMP ----

SDA_INIT_START_TIME DC D'0'  Start time
SDA_INIT_END_TIME DC D'0'  End time

IATXSTAT FUNC=TIMESTAMP,  Set start time
        DATATYPE=INIT,
        DATA=START_TIME

---- RESET Specified Fields ----

SDA_USER1_FIELD1 DC F'0'
SDA_USER1_FIELD2 DC F'0'

IATXSTAT FUNC=RESET,  Reset information
        DATATYPE=USER1,
        DATA=(FIELD1,FIELD2)

---- INC to Increase the Count ----

SDA_INIT_JDS_READS DC F'0'  JDS reads

IATXSTAT FUNC=INC,  Increase the JDAB reads
        DATATYPE=INIT,
        DATA=JDS_READS

---- DEC to Decrease the Count ----

SDA_INIT_JDS_READS DC F'0'  JDS reads

IATXSTAT FUNC=DEC,  Decrease the JDAB reads
        DATATYPE=INIT,
        DATA=JDS_READS

---- ADD Information to a Specified Field ----

SDA_INIT_JDS_READS DC F'0'  JDS reads for all jobs
JVDJDSRD DC F'0'  JDS reads for this job

IATXSTAT FUNC=ADD,  Add JDS read for this job
        DATATYPE=INIT,
        DATA=JDS_READS
        FROM=JVDJDSRD

---- SUBTRACT Information ----

IATXSTAT FUNC=SUBTRACT,  Subtract buffers used by this job
        DATATYPE=SPPOOL,
        DATA=BUFFER_COUNT
        FROM=(R2)

---- COPY the Data ----

SDA_INIT_RESTART_TYPE DC X'0'  Restart type
TVRSTFLG DC X'0'  Restart type

IATXSTAT FUNC=COPY,  Copy JES3 start type
        DATATYPE=INIT,
        DATA=RESTART_TYPE,
        FROM=TVRSTFLG

---- MAX to Determine a Maximum Value ----
**IATXSTMD (Set Dispatching Mode)**

The IATXSTMD macro changes the execution mode of writer DSPs and the services they call. There are two possible execution modes:

- **NUCTASK** mode indicates that the calling program is to be executed under the JES3 primary task (IATNUC).
- **AUXTASK** mode indicates that the calling program is to be executed under the JES3 auxiliary task (IATAUX) when the task is enabled for work. To enable the auxiliary task, issue an *MODIFY MT=ON command or specify MT=ON on the OPTIONS statement in the initialization stream.

(If IATXSTMD is issued by a DSP running on a JES3 local processor, the set dispatching mode service routine immediately returns control to the DSP without performing the requested function.)

When IATXSTMD is invoked, if the caller is already in the requested execution mode, the “set mode” service immediately returns to the caller without any loss of control. Otherwise, the function performed is as follows:

1. **NUCTASK** mode request – The new execution mode is set in the caller’s FCT. If the caller’s FCT is not already running under control of the JES3 primary task, the FCT is made dispatchable under the primary task. Control is then relinquished to the multifunction monitor (MFM) to dispatch another FCT under the auxiliary task.

2. **AUXTASK** mode request – If the DSP has not previously attached an auxiliary task dispatching element (ATDE) to allow execution under the auxiliary task, control is immediately returned to the caller. Otherwise, the new execution mode is set in the caller’s FCT, and if the auxiliary task is currently enabled for work, the caller’s FCT is made dispatchable under the auxiliary task. Control is then relinquished to the MFM to dispatch another FCT under the primary task. If, however, the auxiliary task is not enabled for work, then control is returned.
directly to the caller without a switch in FCT dispatchability, but with the FCT indicating AUXTASK mode. Should the auxiliary task subsequently become enabled for work (as the result of an *MODIFY MT=ON command), the FCT is made dispatchable under the auxiliary task at that time (providing the FCT is still in AUXTASK mode).

In the case where the IATXSTMD request actually results in a switch in FCT dispatching, control returns to the caller at the next sequential instruction after the macro interface when the caller’s FCT is redispached under the required task.

**Note:** The execution mode of an FCT only indicates the JES3 task under which its DSP desires to run. It does not necessarily indicate under which task its DSP is actually executing. When in NUCTASK mode, the DSP always executes under the primary task. However, when in AUXTASK mode, the DSP executes under the auxiliary task only if the auxiliary task is enabled for work. Otherwise, it will continue to run under the primary task even though the mode indicated in the FCT is AUXTASK mode. The IATXTEST macro can be used by a DSP to determine what mode it is in and to determine what task it is actually executing under.

The following table illustrates the variables and results when a mode change request is issued.

<table>
<thead>
<tr>
<th>Requested MODE</th>
<th>MULTITASKING ENABLED (MT=ON)</th>
<th>MULTITASKING DISABLED (MT=OFF)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Running Under IATNUC Task</td>
<td>Running Under Auxiliary Task</td>
</tr>
<tr>
<td></td>
<td>with ATDE</td>
<td>with ATDE</td>
</tr>
<tr>
<td></td>
<td>without ATDE</td>
<td>without ATDE</td>
</tr>
<tr>
<td>NU CTASK</td>
<td>Return to caller</td>
<td>Return to caller</td>
</tr>
<tr>
<td>AU XTASK</td>
<td>FCT mode set to AUXTASK and made dispatchable under auxiliary task. Return control to MFM.</td>
<td>Return to caller</td>
</tr>
</tbody>
</table>

**Syntax**

```
IATXSTMD  
label MODADDR = (R0) address 
MODE = ATASK SAVMODE = NTASK
```
SAVMODE=:

\[
\text{SAVMODE} = \begin{cases} 
\text{(reg)} \\
\text{address} 
\end{cases}
\]

**Parameters**

**MODADDR=**
Specifies a location one byte in length that contains the dispatching mode that is to be set for the FCT. If a register is specified, the mode is contained in the high-order byte. This parameter permits a previously saved mode to be restored.

**MODE=**
Specifies the FCT execution mode.

- **ATASK**
  Specifies that the caller desires to execute under the JES3 auxiliary task.

- **NTASK**
  Specifies that the caller desires to execute under the JES3 primary task.

**SAVMODE=**
Specifies a location one byte in length where the current FCT execution mode is to be saved. If a register is specified, the mode is saved in the high-order byte. Register 0 cannot be specified because it is used as a parameter register to the “set mode” service. The SAVMODE parameter may be specified only with the MODE parameter.

**Linkage**

The IATXSTMD macro expansion uses register 0 as a parameter register. The macro issues the IATZCALL macro specifying SAVE=NO, which generates a BALR R14,R15 for linkage.

**Restrictions**

Do not use this macro in any routine that might run in a C/I FSS address space.

The syntax diagram below describes the parameters supported by this macro.

**IATXSTTA (Search for Allocated STT Extents)**

Inquiry and modify modules use the IATXSTTA macro to determine the status of single track table (STT) allocations on a spool data set. The number of STT spool records that are defined on the spool data set is returned in register 1. The number of available spool records within the STT extents is returned in register 0. If registers 1 and 0 have equal, non-zero values, then none of the STT spool records on the data set are being used.

**Syntax**

\[
\text{IATXSTTA} \rightarrow \text{label} \rightarrow \text{FIND} \rightarrow \text{OP=} \rightarrow \text{USE} \rightarrow \text{DRAIN} \rightarrow \text{EXT=} \rightarrow \text{(reg)} \rightarrow \text{address}
\]
Parameters

**OP=**

**DRAIN**
Specifies that STT record allocation be suspended on the data set identified using the EXT parameter.

**USE**
Specifies that STT record allocation be resumed on the data set identified using the EXT parameter. This option reverses the effect of specifying **OP=(DRAIN)**.

**FIND**
Requests that the status of the STT allocations on the data set identified by the EXT parameter be returned in registers 0 and 1.

**EXT=**
Specifies the address of a halfword field containing the extent number of the spool data set to be processed. If a register is used, the register contains the extent number.

Linkage

This macro uses ASAVE linkage.

Restrictions

Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task.

IATXSYSU (Access SYSUNITs Table)

Use IATXSYSU to access the SYSUNITs table.

Syntax

```
IATXSYSU FUNC= DIRECT,SEQUENTIAL,ERROR=label

NORMAL=label

DIRECT:
    DIRECT_ACCESS INDEX=

SEQUENTIAL:
    SEQ_ACCESS INDEX=

INDEX=:
    INDEX=label (Rx)
```
Parameters

FUNC=DIRECT_ACCESS I SEQ_ACCESS
Specifies the type of SYSUNIT access you want to perform.

DIRECT_ACCESS
Use this function to access a particular SYSUNITs table entry directly by its SYSUNITs index value. The address of the SYSUNITs entry is returned in register 1.

SEQ_ACCESS
Use this function to access the SYSUNITs entries sequentially. The address of the SYSUNITs entry is returned in register 1.

INDEX=label | (Rx)
Specifies the address or register containing the SYSUNITs index value for the SYSUNITs entry you want to access.

Note: This parameter is required for FUNC=DIRECT_ACCESS requests.

TOKEN=label | (Rx,Ry)
For SEQ_ACCESS requests only.
Specifies an 8-byte token that identifies the last SYSUNIT entry that was accessed for the last sequential access request. To begin accessing the SYSUNIT entries sequentially, the caller must first set the token value to zero. Each IATXSYSU macro call updates the token value, thereby allowing JES3 to access the next SYSUNIT entry on the next IATXSYSU macro call.

To obtain faster SYSUNIT table scanning, specify this token value as a pair of registers. The first register in the pair cannot be R0 or R1. The second register in the pair can be any register except R0. The registers do not have to be contiguous. When the SEQ_ACCESS function is called for the first time, the contents of the registers must be set to zero. Also the calling routine must not alter the register contents until it is finished calling the SEQ_ACCESS service.

KEYS=(field,data,cond,...,fieldn,datan,condn)
For SEQ_ACCESS requests only.
Specifies the search keys to be used in determining which SYSUNIT entry should be returned next.

field
Indicates the name of a field in the SYSUNIT control block (for example, SYSFLAG1 or SYSSETX).

data
Indicates the data JES3 compares or tests against the field. If the field is a bit mask, data must be the name of one or more equates to be tested.

If the field is a data field (indicating JES3 should perform a comparison against the field), data must be the value to compare against the field.

cond
Specifies the condition associated with the comparison or test that causes
IATXSYSU

JES3 to return the SYSUNIT entry. For example, for bit mask tests, specify ALLON if the SYSUNIT entry should be returned when all bits are on.

If you specify multiple keys, JES3 returns the SYSUNIT entry if any of the keys match.

**Examples**

```
KEYS=(SYSFLAG1,SYSNALOC+SYSRSRV,NALLOFF)
KEYS=(SYSFLAG1,SYSRSRV,ALLON,
      SYSVOLAD,TVTZERO,NE)
```

**EOD=label**

Specifies the address to which to return control when JES3 reaches the end of the SYSUNIT entries.

**Note:** This parameter is valid only for FUNC=SEQ_ACCESS requests.

**ERROR=label**

Specifies the address to which JES3 returns control if processing is unsuccessful. For example, for a FUNC=DIRECT_ACCESS request, JES3 determines the SYSUNIT index value to be not valid.

If you do not specify ERROR=, JES3 performs no error checking.

**NORMAL=label**

Specifies the address to which JES3 returns control if processing is successful. If you do not specify NORMAL=, JES3 continues processing at the next sequential instruction.

**Linkage**

Generates inline code.

**Restrictions**

None.

**IATXTEST (Test FCT Mode or Task Control)**

The IATXTEST macro determines the current execution mode of an FCT or determines under what task the FCT is currently running.

**Syntax**

```
/SM590000/SM590000
IATXTEST
  label
  MODE
  TASK=R11,(Rx)
,FCT=(R11)
  NTSKRET=address
,ATSKRET=address
/SM590000/SM630000
```

**Parameters**

**MODE**

Specifies that the IATXTEST macro is to test the execution mode of the FCT.
TASK
   Specifies that the IATXTEST macro is to test the task under which the FCT is running.

FCT=
   Specifies a register containing the address of the FCT to be tested.

NTSKRET=
   When used with the MODE parameter, this parameter specifies the address to receive control if the FCT is in NUXTASK mode. If this parameter is omitted and the FCT is in NUXTASK mode, control is returned to the next sequential instruction after the macro expansion. If the FCT is in AUXTASK mode, the results depend on whether the ATSKRET parameter is specified.

   When used with the TASK parameter, this parameter specifies the address to receive control if the FCT is running under the JES3 primary task. If this parameter is omitted and the FCT is running under the JES3 primary task, control is returned to the next sequential instruction after the macro expansion. If the FCT is running under the JES3 auxiliary task, the results depend on whether the ATSKRET parameter is specified.

ATSKRET=
   When used with the MODE parameter, this parameter specifies the address to receive control if the FCT is in AUXTASK mode. If this parameter is omitted and the FCT is in AUXTASK mode, control is returned to the next sequential instruction after the macro expansion. If the FCT is in NUXTASK mode, the results depend on whether the NTSKRET parameter is specified.

   When used with the TASK parameter, this parameter specifies the address to receive control if the FCT is running under the JES3 auxiliary task. If this parameter is omitted and the FCT is running under the JES3 auxiliary task, control is returned to the next sequential instruction after the macro expansion. If the FCT is running under the JES3 primary task, the results depend on whether the NTSKRET parameter is specified.

Linkage
   This macro expands as inline code.

Restrictions
   When this macro is used, the IATYFCT mapping macro must be included in the module.

IATXTOD (Obtain Local Time of Day)
   The IATXTOD macro obtains the time of day in binary, packed decimal, or EBCDIC format.

Syntax

```
label IATXTOD BIN address
```

Chapter 2. JES3 Macros 475
IATXTRC (Make Entry in a JES3 Trace Table)

The IATXTRC macro makes an entry in a JES3 trace table. The trace table can be displayed in a JES3 ABEND dump or on the operator console. The trace event will be uniquely identified and may optionally contain user-supplied data or registers. Entries also will have the time-of-day value included. When the macro is issued, the system must be in supervisor state, enabled, and running under the JES3 storage protection key.

Figure 6 on page 31 has an example of this macro.

Syntax

```
IATXTRC—ID=number—ATRADDR=address
```

Parameters

**ID=**

Specifies a number from 1 to 4095 which specifies the trace ID number of the event. IDs 0001-2999 are reserved for IBM; IDs 3000-4095 are available to users. See z/OS JES3 Diagnosis for the JES3 trace ID numbers.
ATRADDR
Specifies the address of a Trace Control Data Area mapped by macro IATYATR. This allows the caller to use a different trace table for this trace entry. The trace table is assumed to be in a data space whose ALET is in the field ATRALET. The trace control data area must be initialized with trace table pointers for the first, current, and last trace entries.

REG=
Y  Specifies registers 2-9 are to be traced.
N  Specifies registers 2-9 are not to be traced.

DATA=
Specifies the address of the data to be appended to the trace entry. If coded, the LNG or XLNG parameter must also be coded.

LNG=
Specifies the length of the data to which the DATA parameter points. The value must be some number of fullwords. The value may be specified as either a decimal number or a symbol. The maximum value is 64.

If coded, the DATA parameter must also be coded and the XLNG parameter cannot be coded. LNG should be used if your user data is between 1 and 64 words long, inclusively.

XLNG=
Specifies the length of the data to which the DATA parameter points. The value specifies the number of fullwords. The value may be specified as either a decimal number or a symbol. The maximum value is 1023.

If coded, the DATA parameter must also be coded and the LNG parameter cannot be coded. XLNG should be used if your user data is between 64 and 1023 words long.

TYPE=
IAT  Specifies that the address of the trace routine is to be obtained from the TVT.

SVT  Specifies that the address of the trace routine is to be obtained from the subsystem vector table (SSVT).

FSS  Specifies that the address of the trace routine is to be obtained from the functional subsystem/functional subsystem application common control block, IATYFSCB.

NORMAL=
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

Linkage
This macro issues a BALR R14,R15 for linkage.

Restrictions
None.
IATXTREG (Inner Macro to Determine if Operand Is in Register Notation)

Macro definitions (the macro code in the JES3 macro library) use the IATXTREG macro to determine if a macro operand is input in register notation. This macro requires the user to define two global variable set symbols:

- A character symbol (&REG)
- An arithmetic symbol (&CNT)

If the operand passed to IATXTREG is input in register notation, the register specified is returned to the caller in the &REG global set symbol. The &CNT global set symbol is used as a return indicator and is set as follows:

- &CNT 0: The operand is not a register.
- &CNT 1: The operand is a register.
- &CNT 2: An error occurred. The macro operand passed is null.

Syntax

```
label IATXTREG macro_operand
```

Linkage

This macro does not generate executable code.

Restrictions

Only macro definitions may use this macro.

IATXTRMT (End a SNA RJP Session)

The IATXTRMT macro ends the specified session.

The routine using this macro must set register 11 according to the SNA RJP convention. In this convention, the high-order bit of register 11 is used to indicate the type of mode as follows:

- If the bit is off (0), it is running under the FCT.
- If the bit is on (1), it is running under the IRB or the SRB.

To distinguish an IRB from an SRB, test the PSATOLD word in the PSA. If this word is zero, the SRB is active. If the word is not zero, the IRB is active.

Syntax

```
label IATXTRMT LUCB=(R1)(reg) address, TYPE=(R0)(reg) Q QI
```

```
MODE=DSP RB
```

```
```

```
Parameters

LUCB=
  Specifies the address of the logical unit control block (LCB) for the session to be ended.

TYPE=
  Specifies the type of termination requested.

(R0)
(reg)
  Specifies the register containing the type indicator. Q, and QI have type indicators of 0, and 4, respectively.

Q  The session is to end by using a VTAM CLSDST macro; however, the session is allowed to reach the “between bracket” state normally.

QI (quiesce immediately) The session is to end as soon as possible by using a VTAM CLSDST macro.

MODE=

DSP
  Indicates that the user is a DSP.

RB
  Indicates that the user is a service request block (SRB) or an interruption request block (IRB).

Linkage

When the MODE=DSP parameter is specified, this macro uses ASAVE linkage.
When the MODE=RB parameter is specified, this macro issues a BALR R14,R15 instruction for linkage.

Restrictions

Use this macro only in the SNARJP environment.

IATXUBAL (Allocate User Memory Buffer)

The IATXUBAL macro allocates an unprotected buffer in the user’s address space that will be used to satisfy an I/O request.

Upon return, register 1 contains the address of the buffer.

Syntax

```
label IATXUBAL—BUSY=—address—NORMAL=—address—
```

Parameters

BUSY=
  Specifies the address to which the routine returns if no buffer was available for allocation.

NORMAL=
  This parameter specifies that after the macro processing routine executes, the
issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

Linkage

This macro issues a BALR R14,R15 for linkage.

Restrictions

Do not use this macro in any routine that might run under the JES3 auxiliary task.

IATXVAL (Validate JES3 Spool Record Address)

The IATXVAL macro performs validity tests on a JES3 file description block (FDB) in which a JES3 spool record address resides. This macro performs three tasks:
1. Validates the spool record address in field FDBSPADR of the FDB
2. Checks the settings of the FDB flags
3. For an FDB of a job or data set track allocation table (TAT), reads the TAT chain into storage and validates all track group allocations for the TAT chain.

For an FDB of a job or data set TAT (data area IATYJBT), the ERROR, NAVAIL, REPL, and HOLD returns apply to the spool data set described by the FDB and also to the track groups described by the TAT chain.

Syntax

```
IATXVAL label FDB=(R1) (reg) address, ERROR=address, NAVAIL=(reg) address, REPL=(reg) address, HOLD=(reg) address, NORMAL=address
```

Parameters

**FDB=**
Specifies the address of the FDB to be validated.

**ERROR=**
Specifies the address of the routine that receives control if the FDB is incorrect.

**NAVAIL=**
Specifies the address to which the routine returns if the FDB describes a spool record on a spool data set that is unavailable. If NAVAIL is not coded and the spool data set is unavailable, the routine returns to the address specified with the ERROR parameter.

**REPL=**
Specifies the address to which the routine returns if the FDB describes a spool record on a spool data set that has been replaced or deleted. If this parameter is not specified and the spool data set has been replaced or deleted, the routine returns to the address specified with the ERROR parameter.
HOLD=
Specifies the address to which the routine should return if the FDB describes a spool record on a spool data set that has been held. If this parameter is not specified and the spool data set has been held, the routine returns to the address specified with the NORMAL parameter.

NORMAL=
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

Linkage
This macro uses ASAVE linkage.

Restrictions
IATXVAL can be used only during JES3 initialization.

IATXVFDB (Validate the Contents of an FDB)

During JES3 job validation, the IATXVFDB macro validates the contents of a file description block (FDB) that describes a single record file (SRF) or a multi-record file (MRF). If the contents indicate that the FDB is incorrect, the macro service returns a return code indicating this.

To validate the contents, the IATXVFDB macro checks the status indicators to ensure that the FDB is in one of the following conditions:
- The FDB is for a single record file or for a chained single record file (if MRF=NO).
- The FDB is for a multi-record file (if MRF=YES).

The macro service also checks whether the FDB is properly closed before checkpointing. The FDB is incorrect if it is in one of the following conditions:
- The FDB is for a multi-record file (MRF) when MRF=NO was specified or defaulted, or for a single record file (SRF) when MRF=YES was specified.
- The FDB is for a JBT when TAT=NO was specified or defaulted.

The IATXVFDB macro records the spool record’s FDB with the diagnostic information about the job using the IATXVSERV macro. If an error is found in the FDB, or if the record is located on an extent that is unavailable, replaced, or deleted, the macro service routine will use the IATXVSRE macro to include diagnostic text about the record with the diagnostic information for the job.

Syntax

\[ \text{label} \rightarrow \text{IATXVFDB} \rightarrow \text{MF=} \rightarrow \text{(-E,-(reg)-)} \rightarrow \text{(-E,-address-)} \rightarrow \text{FDB=} \rightarrow \text{address} \rightarrow \text{(reg)} \rightarrow \text{NO} \rightarrow \text{YES} \rightarrow \text{ID=} \rightarrow \text{address} \rightarrow \text{(reg)} \]
IATXVFDB

Parameters

**MF=**
- Specifies if the list or execute form of the macro should be used.
  
  **(E,(reg))**
  - Specifies the execute form of the macro. The second positional parameter specifies the address of the parameter list that was generated by the list form of the macro.

  **L**
  - Specifies the list form of the macro. The list form generates a parameter list that will be used by each occurrence of the execute form of the IATXVFDB macro. A label must be associated with the list form so that the execute form can access the parameter list generated by the list form.

**FDB=**
- Specifies the address of the FDB that describes the single record file that needs to be validated. The address of the FDB is in a register. The address is specified explicitly on the macro.

**TAT=YES|NO**
- Indicates whether the FDB is for a job or data set TAT.

**MRF=YES|NO**
- Indicates whether the FDB is for a multi-record file (MRF).

**ID=**
- Specifies the 4-character name of the SRF that the FDB describes.

**DESC=**
- Specifies the address of a 64-byte data area that contains an EBCDIC description of the expected contents of the spool record. IBM suggests that your installation initializes the data area with text that describes the SRF that the FDB describes.
  
  **(reg)**
  - Specifies the address of the data area is in a register.
  
  **addr**
  - Specifies the address is specified explicitly on the macro.

**ROOT=**
- Specifies the master FDB that contains the address of the FDB specified on the FDB parameter. If ROOT is not specified, the root value from the previous expansion of the parameter list will remain in the parameter list.
  
  **0**
  - Indicates that the FDB specified on the FDB= parameter does not have a master FDB.
  
  **(reg)**
  - Specifies the address of the master FDB. The address will appear in the job’s spool record summary report if the FDB is incorrect.
NAVAIL=
   Specifies the address to which control returns if the FDB is located on a spool
data set that has been replaced, deleted or is in the UNAVAIL partition.

(reg)
   Specifies the address of the FDB is in a register.

addr
   Specifies the address is specified explicitly on the macro.

ERROR=
   Specifies the address to which control returns if the FDB is incorrect.

Linkage

This macro uses ASAVE linkage.

Restrictions

The IATXVFDB service routine is located in module IATDMJV and is pointed to by
the job validation/restart data csect (mapped by IATYJVD) during the job validation
phase of initialization. IATXVFDB is used in the validation modules in place of the
IATXVAL macro.

Registers 0, 1, 14, and 15 are destroyed by this macro. Register 0 cannot be
specified for the NAVAIL= or the ERROR= parameters; register 1 can only be
specified for the MF=(E,(reg)) parameter. All other parameters may specify registers
0, 2-12, 14, or 15.

Return Codes

The IATXVFDB macro can return to the caller:
   • Without any errors – a normal return
   • Without being able to access the FDB – a NAVAIL return
   • With an error – an ERROR return

For a normal return to the caller, the following return codes (in hexadecimal) are
valid:

<table>
<thead>
<tr>
<th>Return Code In Hex</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>The FDB was valid. A JESREAD macro can be used to read the SRF from spool. If the control block needs to be updated, it can be written back to spool using the AWRITE macro.</td>
</tr>
<tr>
<td>04</td>
<td>The FDB was valid. A JESREAD macro can be used to read the SRF from spool. However, the SRF resides on a data set in a HELD partition. The SRF cannot be updated until the data set becomes available.</td>
</tr>
</tbody>
</table>

For a NAVAIL return to the caller, the following return codes (in hexadecimal) are
valid:

<table>
<thead>
<tr>
<th>Return Code In Hex</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>08</td>
<td>The FDB was valid. However, the SRF the FDB describes is located on a data set in an unavailable spool partition. Data sets in an unavailable partition cannot be accessed.</td>
</tr>
</tbody>
</table>
Return Code In Hex | Meaning |
--- | --- |
0C | The FDB was valid. However, the SRF the FDB describes is located on a data set that was replaced. |
10 | The FDB was valid. However, the SRF the FDB describes is located on a data set that was deleted from JES3. |

For an *ERROR return* to the caller, the following return codes are valid:

Return Code In Hex | Meaning |
--- | --- |
14 | The FDB is incorrect. Indications of an incorrect FDB are:
   - The FDB is for a multirecord file instead of an SRF, or the job TAT FDB (not a single record file)
   - The file was in use when the FDB was checkpointed (within a spool record)
   - The spool address in the FDB is not valid (the spool extent does not exist, or the record number is outside of the range of its extent) |

---

**IATXVMSG (Write Messages About Job Validation to the JES3OUT Data Set)**

The IATXVMSG macro writes a message to the job validation JES3OUT/Console Messages section of the SNAP output for each job. The IATXVMSG macro can also issue the message to the operator.

**Syntax**

```
IATXVMSG,label,text_address
```

**Parameters**

- **text-address**
  Specifies the address of message text that will be displayed in the Spool Record Diagnostic/Error Messages portion of the SNAP output for the job. The text must be defined by the list form of the WTO macro. The address can be specified in either a register or an address label. Do not use register 0 for the address.

- **WTO=**
  Specifies whether the message should be written to the operator's console.

  - **N** Specifies the message will be written only to the JES3OUT data set.
  - **Y** Specifies that the message will be written to the JES3OUT data set and the operator's console.

**Linkage**

This macro uses ASAVE linkage.
Restrictions

The IATXVMSG service routine is located in module IATINJV and is pointed to by the JVD during the job validation phase of initialization. IATXVMSG should be used in validation modules instead of the IWASPOUT macro.

Registers 0, 1, 14, and 15 are destroyed by this macro.

IATXVSRE (Save Information About Job Validation Spool Errors)

The IATXVSRE macro saves information about error conditions that are detected for the control block being validated. Text is specified on the macro to identify the contents of the incorrect control block. The information is written to the Spool Record Diagnostic/Error section of the SNAP for the job.

Syntax

```
IATXVSRE label, text_address, FDB=(reg) address
```

Parameters

text-address

Specifies the address of diagnostic text that will be displayed in the Spool Record Diagnostic/Error Messages portion of the SNAP output for the job. The text must be defined by the list form of the WTO macro. The address can be specified in either a register or as an address label.

FDB

Specifies the address of the single record FDB that describes the incorrect record. If a register is used to specify the address, use register 0, or registers 2-15.

Linkage

This macro uses ASAVE linkage.

Restrictions

The IATXVSRE service routine is located in module IATDMJV and is pointed to by the JVD during the job validation phase of initialization.

Registers 0, 1, 14, and 15 are destroyed by this macro.

IATXVSRV (Save Information About Spool Records Associated with a Job)

The IATXVSRV macro records information for spool records associated with the job that is being validated. JES3 uses the information to generate the Summary of Spool Records Validated section when a SNAP is requested for the job.

Syntax

```
IATXVSRV label, MF=(E, (reg)) L(E, address), FDB=(reg) address
```

Chapter 2. JES3 Macros 485
Parameters

MF=
    Specifies if the list or execute form of the macro should be used. Specifies the
    execute form of the macro. The second positional parameter specifies the
    address of the parameter list that was generated by the list form of the macro.

L
    Specifies the list form of the macro. The list form generates a parameter list that
    will be used by each occurrence of the execute form of the IATXVSRV macro. A
    label must be associated with the list form so that the execute form can access
    the parameter list generated by the list form.

FDB=
    Specifies the address of the FDB that describes the single record file that needs
    to be recorded. The address of the FDB is in a register.

    The address is specified explicitly on the macro.

ID=
    Specifies the 4-character name associated with the spool record. The name
    specified is checked against the record identifier in the requested record.

DESC=
    Contains the address of a 64-byte data area that contains an EBCDIC
    description of the expected contents of the spool record. IBM suggests that your
    installation initialize the data area with text that describes the SRF that the FDB
    describes. If the FDB is incorrect, this text will appear in the job’s spool record
    summary report. The address of the data area is loaded into a register. The
    address is specified explicitly on the macro.

ROOT=
    Specifies the master FDB that contains the address of the FDB specified on the
    FDB parameter. If ROOT is not specified, the root value from the previous
    expansion of the parameter list will remain in the parameter list. Indicates that
    the FDB specified on the FDB= parameter does not have a master FDB.
    Specifies the address of the master FDB. The address will appear in the job’s
    spool record summary report if the FDB is incorrect.

Linkage

This macro uses ASAVE linkage.

Restrictions

The IATXVSRV service routine is located in module IATDMJv and is pointed to by
the job validation/restart data csect (mapped by IATYJVD) during the job validation
phase of initialization.

 Registers 0, 1, 14, and 15 are destroyed by this macro. The macro parameters may
specify registers 0, 2-12, 14, or 15.
IATXWLST (Wait for ECF Post)

The IATXWLST macro causes either an AWAIT TYPE=ON or AWAIT TYPE=OFF macro to be issued in order to wait on (1) the posting of a specified event completion flag (ECF) and (2) the posting of the cancel ECF in the device’s device entry table (DVE). The wait is satisfied if either or both ECFs are posted.

Syntax

```
label IATXWLST TYPE= ON , EDFADD= (reg) address

label IATXWLST TYPE= OFF , EDFADD= (reg) address

label IATXWLST TYPE= OFF , ECFMASK= (reg) flag
```

Parameters

**TYPE=**

**ON**

Specifies that an AWAIT TYPE=ON macro is to be issued.

**OFF**

Specifies that an AWAIT TYPE=OFF macro is to be issued.

**ECFADD=**

Specifies the address of the ECF to be posted.

**ECFMASK=**

Specifies the hexadecimal flag to be posted. This must be an absolute expression or a symbol.

**Note:** When register notation is used, ECFADD and ECFMASK must use separate registers. ECFADD cannot specify register 0 or 1, and ECFMASK cannot specify register 0. When in a register, the ECF mask must be in the low-order byte with the high-order three bytes containing zeros.

Linkage

This macro generates inline code. When this macro executes, the contents of registers 0, 14, and 15 are destroyed.

Restrictions

This macro requires addressability to the device entry table (IATYDVE). Do not use this macro in any routine that might run in a C/I FSS address space.

IATXWOPN (SNA RJP Work Station Open)

A reader, writer, or console DSP uses the IATXWOPN macro to obtain a session on which it can send or receive data in an SNA environment.

Syntax

```
label IATXWOPN SUPUNIT= (RO) address
```
IATXWOPN

Parameters

**SUPUNIT**=
Specifies the address of the SUPUNIT entry for the device for which a session is needed.

**CTAB**=
Specifies the address of the compaction table to be used for this data. This parameter is checked only when the writer DSP issues the IATXWOPN macro. If CTAB is coded, ECF cannot be coded.

**PDIR**=
Specifies the address of a table which has the format of the peripheral data set information record (PDIR). This parameter is checked only when the writer DSP issues the IATXWOPN macro. If PDIR is coded, ECF cannot be coded.

**ECF**=
Specifies the address of a doubleword containing the address of the ECF to be posted in word 1, and the ECF mask in the low-order byte of word 2. The first three bytes of word 2 must contain zeros. ECF is used only by a reader DSP.

If ECF is not coded, the reader DSP will wait on data arrival only. If ECF is coded, the reader DSP will wait on data or the ECF.

If ECF is coded, CTAB and PDIR cannot be coded.

**CONSOLE**=
Ignored unless the SUPUNIT entry specifies this is a console device. If specified, this parameter indicates inbound or outbound.

**ERROR**=
This parameter specifies the address to which the macro processing routine returns if an error occurs. When control is passed to the address, register 0
contains a dump code and register 15 contains a return code, unless otherwise
specified. The return codes are described below.

If an error occurs and ERROR was not specified, the dynamic support program
(DSP) will be ended by using the FAILDSP macro.

NORMAL=
This parameter specifies that after the macro processing routine executes, the
issuing program can resume execution at an instruction other than the
instruction that immediately follows the macro.

Return Codes
Following are the VTAM codes returned in register 15 when return is to a location
specified in the ERROR parameter:

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>Session not available.</td>
</tr>
<tr>
<td>04</td>
<td>Permanent device error.</td>
</tr>
<tr>
<td>08</td>
<td>PDIR not fully supported.</td>
</tr>
</tbody>
</table>
| 0C          | CTAB rejected. A session has been started and I/O can be sent to the device. If
data is sent, it is compacted using the default compaction table. |
| 18          | Device ended. |
| 20          | Intervention required. |

Linkage
This macro uses ASAVE linkage.

Restrictions
This macro must run under DSP control. Do not use this macro in any routine that
might run in a C/I FSS address space or under the JES3 auxiliary task.

IATXWRE (Recover From WRITE Macro Error)
The IATXWRE macro places the appropriate new FDB (as a result of a recoverable
WRITE macro error) in the anchor location. If the error is a permanent WRITE
macro error, a FAILDSP macro is issued.

Syntax

```
IATXWRE label
DMPCD=(R0) address
DYQ=(R1) address
NORMAL= address
```

Parameters

DMPCD=
Specifies the dump code received from the WRITE macro error.

DYQ=
Specifies the address of the dynamic allocation queue (DYQ) entries associated
with the WRITE macro error.
IATXWRE

NORMAL=
This parameter specifies that after the macro processing routine executes, the
issuing program can resume execution at an instruction other than the
instruction that immediately follows the macro.

Linkage

This macro uses ASAVE linkage.

Restrictions

Do not use this macro in any routine that might run in a C/I FSS address space or
under the JES3 auxiliary task.

IATXWSCL (SNA RJP Work Station Close)

A reader, writer, or console DSP uses the IATXWSCL macro to indicate the end of a
data stream on a particular session in a SNA environment. This macro results in a
call to IATSNDNC.

Syntax

```
IATXWSCL SUPUNIT=(R1) TYPE=(R0) NORMAL CANCEL ABORT TEMP FINAL
```

Parameters

SUPUNIT=
Specifies the address of the SUPUNITS table entry that indicates which device
is at the end of a data stream.

TYPE=
Specifies the type of session close requested. The registers specified should
contain 0, 4, 8, 12, or 16 for NORMAL, CANCEL, ABORT, TEMP, or FINAL,
respectively.

NORMAL
No errors. An EDS is sent.

CANCEL
An end-of-chain request/response unit (RU) is not sent. If the session is in
"chain" state, then the session is canceled.

ABORT
CANCEL is issued, followed by ADS.

TEMP
Indicates that the session is available for outbound messages.
FINAL
An EDS and an EB are sent.

CONSOLE=
Specifies whether this close request is for a workstation whose console traffic begins with a BDS or if console data was sent by interrupting an outbound flow.

Linkage
This macro uses ASAVE linkage.

Restrictions
This macro must run under DSP control. Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task.

IATZCALL (Inner Macro to Create Routine Linkage)
The IATZCALL inner macro provides entry linkage to a routine.

Syntax
```
IATZCALL label routine_label ,SAVE=YES
,SAVE=NO
,FLAG=number ,EOD=address ,EOF=address
,NAVAIL=address ,ERROR=address ,AMBIG=address
,IPL=address ,BUSY=address ,REJECT=number
,RJPCAN=address ,TYPESNA=address ,TYPEBSC=address
,TYPETCP=address ,NORMAL=address
```

Parameters
- **routine-label**
  Specifies the label of the routine to be entered. If this parameter is specified, it must be specified as the first parameter for this macro. If omitted, a branch table will be created.

- **SAVE=**
  - **YES**
    Specifies that entry is to be made through the ASAVE linkage routine ensuring the integrity of registers 2 through 13.
  - **NO**
    Specifies that entry is to be made using BALR R14, R15.
IATZCALL

FLAG=
    Specifies a flag bit to be inserted into the first bit of the entry register (register 15). Valid values are 80 (bit is on) and 00 (bit is off). If omitted, the first bit will be binary zero.

EOD=
EOF=
NAVAIL=
ERROR=
AMBIG=
IPL=
BUSY=
REJECT=
RJPCAN=
TYPETCP=
TYPESNA=
TYPEBSC=
NORMAL=
    Specifies the return points. These return points must be specified in the order shown here. Only those return points known by the called routine can be specified. Each use of an operand generates a branch instruction, in the order of its appearance as a parameter of this macro, following the branch and link (BALR) to the called routine. All required return points should default to "**", in the outer macro; when "**" is detected, an MNOTE is generated indicating the absent parameter.

Linkage

The type of entry linkage provided by this macro depends on the specification of the SAVE parameter, explained below.

Restrictions

Only macro definitions (the macro code in the JES3 macro library) may use this macro.

IATZHEX (Inner Macro for Decimal to Hexadecimal Conversion)

The IATZHEX macro accepts input as a decimal value, and returns (by using the global macro symbol HEXTERM) the equivalent self-defining hexadecimal term in character form.

This macro converts internal flag byte values into their corresponding hexadecimal representation, increasing the readability of generated code that contains these flag bytes.

Syntax

```
IATZHEX—value
```

Parameters

value
    Specifies the decimal value that is to be converted. Valid range is 0 to 255.
IATZLOAD (Inner Macro to Load Parameter Register)

The IATZLOAD inner macro loads a register with a specified parameter.

Syntax

```
IATZLOAD label (reg x),name (reg y),label number
```

Parameters

(reg x)

Specifies the register into which the parameter is to be loaded.

name

Specifies the symbolic name of the parameter on the outer macro.

(reg y)

Specifies the parameter to be loaded into the register. If omitted, an MNOTE is generated indicating the absence of the parameter.

LA

RLA

L

LLA

LG

Specifies the type of action required to load the register (see table below). If omitted, LA is assumed.

The following table indicates the instruction generated for each of the possible actions, with ‘x’ symbolizing the ‘register’ and ‘y’ symbolizing the ‘cprameter’ specifications:

<table>
<thead>
<tr>
<th>Action</th>
<th>Register</th>
<th>Label</th>
<th>Address Constant</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>LA</td>
<td>LR Rx,Ry</td>
<td>LA Rx,LABELy</td>
<td>LA Rx,LABELy</td>
<td>LA Rx,y</td>
</tr>
<tr>
<td>RLA</td>
<td>LA Rx,0(0,Ry)</td>
<td>LA Rx,LABELy</td>
<td>LA Rx,LABELy</td>
<td>LA Rx,y</td>
</tr>
<tr>
<td>L</td>
<td>LR Rx,Ry</td>
<td>L Rx,LABELy</td>
<td>L Rx,LABELy</td>
<td>LA Rx,y</td>
</tr>
<tr>
<td>LLA</td>
<td>LR Rx,Ry</td>
<td>LA Rx,LABELy</td>
<td>L Rx,LABELy</td>
<td>LA Rx,y</td>
</tr>
<tr>
<td>LG</td>
<td>LGR Rx,R7</td>
<td>LG Rx,LABELy</td>
<td>LG Rx,LABELy</td>
<td>LGHI Rx,y</td>
</tr>
</tbody>
</table>

Note: The “RLA” action also clears out the high order bit of Rx. If Ry=0, the instructions generated are LR Rx,R0, N Rx,TVTHOB0F.

Linkage

This macro expands as inline code.

Restrictions

Only macro definitions (the macro code in the JES3 macro library) may use this macro.
IATZMEXC

IATZMEXC (Inner Macro for Checking for Mutually Exclusive Parameters)

This is an inner macro that is used by other macros to do the following:

- Check whether two or more mutually exclusive parameters were specified. The parameters may be mutually exclusive depending on a value specified for one of the other parameters.
- Make sure that at least one parameter out of a list of mutually exclusive parameters was specified.

Syntax

```
IATZMEXC(label)
   PARMS=param, VALUES=value, SPEC_VALUES=value, REQ=NO
```

Parameters

PARMS=
The names of the mutually exclusive parameters to be checked.

VALUES=
The values associated with each of the mutually exclusive parameters to be checked.

SPEC_VALUES=
An optional parameter that specifies which parameter values are mutually exclusive. For example, the RQ parameter can only be specified with the VALIDATE parameter if VALIDATE=NO is specified. If VALIDATE=YES is specified, then the RQ parameter cannot be specified.

The default value for each of the parameters is *ANY, which means that if any value is specified for the parameter, it is mutually exclusive with the other parameters.

REQ=
Specifies whether at least one of the mutually exclusive parameters must be specified.

YES
At least one of the mutually exclusive parameters must be specified.

NO
None of the mutually exclusive parameters must be specified.

Linkage

This macro expands in-line.

Restrictions

None.
**IATZMNOT (Inner Macro to Generate Standard MNOTEs)**

The IATZMNOT inner macro generates a standard set of MNOTEs in the assembly listing.

**Syntax**

```
label IATZMNOT action, p1 p2 p3 p4
```

**Parameters**

**action**

Specifies the type of MNOTE to be generated. Use the appropriate number from the “ACTION” column of the table below.

**p1, p2, p3, p4**

Specifies the parameters to be included in the MNOTE generated.

The following table indicates the MNOTE generated for each possible action.

<table>
<thead>
<tr>
<th>ACTION</th>
<th>MNOTE:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>IAT#001 ERROR, ‘p1’ PARAMETER UNSPECIFIED</td>
</tr>
<tr>
<td>2</td>
<td>IAT#002 ERROR, ‘p1=p2’ ILLEGAL SPECIFICATION</td>
</tr>
<tr>
<td>3</td>
<td>IAT#003 ERROR, ILLEGAL REGISTER USAGE FOR ‘p1’</td>
</tr>
<tr>
<td>4</td>
<td>IAT#004 ERROR, ‘p1’ REQUIRES REGISTER NOTATION</td>
</tr>
<tr>
<td>5</td>
<td>IAT#005 ERROR, ‘p1’ MUST BE SPECIFIED IN ‘p2’ FORM</td>
</tr>
<tr>
<td>6</td>
<td>IAT#006 WARNING, ‘p1’ OMITTED, MUST BE SUPPLIED DURING EXECUTION</td>
</tr>
<tr>
<td>7</td>
<td>IAT#007 WARNING, ‘p1=p2’ ILLEGAL, ‘p3’ ASSUMED</td>
</tr>
<tr>
<td>8</td>
<td>IAT#008 WARNING, ILLEGAL COMBINATION OF PARAMETERS, ‘p1’ ASSUMED</td>
</tr>
<tr>
<td>9</td>
<td>IAT#009 ERROR, REGISTER RETURN INCONSISTENT WITH ‘SAVE=NO’</td>
</tr>
<tr>
<td>10</td>
<td>IAT#010 ERROR, ‘p1’ AND/OR ‘p2’ MUST BE SPECIFIED</td>
</tr>
<tr>
<td>11</td>
<td>IAT#011 ERROR, ‘p1’ OR ‘p2’ MUST BE SPECIFIED</td>
</tr>
<tr>
<td>12</td>
<td>IAT#012 ERROR, ‘p1’ IS INVALID WITH ‘p2=p3’</td>
</tr>
<tr>
<td>13</td>
<td>IAT#013 ERROR, ‘p1’ MUST FOLLOW ‘p2’</td>
</tr>
<tr>
<td>14</td>
<td>IAT#014 ERROR, EITHER ‘p1’ OR ‘p2’ MAY BE USED, NOT BOTH</td>
</tr>
<tr>
<td>15</td>
<td>IAT#015 ERROR, ‘p1’ ILLEGAL SPECIFICATION</td>
</tr>
<tr>
<td>16</td>
<td>IAT#016 ERROR, ‘p1’ REQUIRED WITH ‘p2=p3’</td>
</tr>
<tr>
<td>17</td>
<td>IAT#017 ERROR, ‘p1=p2’ INVALID WITH ‘p3=p4’</td>
</tr>
<tr>
<td>18</td>
<td>IAT#018 ERROR, ILLEGAL COMBINATION OF PARAMETERS</td>
</tr>
<tr>
<td>19</td>
<td>IAT#019 WARNING, ‘p1’ PREVIOUSLY SPECIFIED</td>
</tr>
<tr>
<td>20</td>
<td>IAT#020 ERROR, BOTH ‘p1’ AND ‘p2’ MUST BE SPECIFIED OR BOTH OMITTED</td>
</tr>
<tr>
<td>21</td>
<td>IAT#021 ERROR, ‘p1’ INVALID. VALID RANGE= ‘p2-p3’</td>
</tr>
<tr>
<td>22</td>
<td>IAT#022 WARNING, ‘p1’ INVALID, ‘p2’ ASSUMED.</td>
</tr>
<tr>
<td>23</td>
<td>IAT#023 ERROR, ‘p1’ IS INVALID WITH ‘p1.2’</td>
</tr>
<tr>
<td>24</td>
<td>IAT#024 ERROR, ‘p1’, ‘p2’, OR ‘p3’ MUST BE SPECIFIED</td>
</tr>
<tr>
<td>25</td>
<td>IAT#025 ERROR, ONLY ONE OF ‘p1’, ‘p2’, or ‘p3’ MAY BE SPECIFIED</td>
</tr>
<tr>
<td>26</td>
<td>IAT#026 ERROR, TOO MANY PARAMETERS FOR ‘p1’</td>
</tr>
<tr>
<td>27</td>
<td>IAT#027 WARNING, ‘p1’= ‘p2’ IS NO LONGER SUPPORTED</td>
</tr>
<tr>
<td>28</td>
<td>IAT#028 ERROR, ‘p1’, ‘p2’ OR ‘p3’ MUST BE SPECIFIED OR MUST BE OMITTED</td>
</tr>
</tbody>
</table>
**IATZMNOT**

**ACTION**

MNOTE:
29 IAT#029 ERROR, 'p1' SUBPARAMETER OF '&p2' MUST BE SPECIFIED
30 IAT#030 ERROR, 'p1' MUST BE SPECIFIED WITH 'p2' AND 'p3=p4'
31 IAT#031 ERROR, INCORRECT NUMBER OF PARAMETER FOR 'p1'
IAT#000 NONEXISTENT IATZMNOT MESSAGE NUMBER

Any other action specification.

**Linkage**

This macro does not generate executable code.

**Restrictions**

Only macro definitions (the macro code in the JES3 macro library) may use this macro.

**IATZMOVE (Inner Macro to Create Move Instructions)**

The IATZMOVE inner macro creates MVC, ST, and STH assembler statements.

**Syntax**

```
label IATZMOVE from-, to-, length
,displace
,fromkey
tokey
,lenkey
type
```

**Parameters**

**from**

Specifies the location of data or the data to be moved.

**to**

Specifies the location to which the data is to be moved.

**length**

Specifies the length of the data to be moved.

**displace**

Specifies the displacement, from the moved to location, where the data should be located. If omitted, binary zero is assumed.

**fromkey**

**tokey**

**lenkey**

Specifies the symbolic name of the parameter on the outer macro.

**type**

Specifies the move is either from a register or memory.

The following table indicates the assembler statements created for each possible condition; with w symbolizing the **to**, x symbolizing the **displace**, y symbolizing the **length**, and z symbolizing the **from** parameter specifications:
## Linkage

This macro generates inline code.

## Restrictions

Only macro definitions (the macro code in the JES3 macro library) may use this macro.

### IATZParm (Inner Macro for Illegal Parameter Verification)

The IATZParm macro is an inner macro that checks for illegal parameter values.

### Syntax

```plaintext
IATZParm parm_name, value, list
```

### Parameters

- **parm_name**
  Specifies the name of the parameter being checked.

- **value**
  Specifies the value supplied for the parameter. The values can be in one of three forms: label, register, or constant.

- **list**
  Specifies all incorrect values for the parameter. The values can be in one of three forms: label, register, or constant.

### IATZPCHK (Parameter Checker)

IATZPCHK is used to verify that:

- The value specified for a parameter is valid. For example, make sure that the value for parameter `FUNC=` is `ADD`, `DELETE`, or `INITIALIZE`.

- A required value was actually specified for a parameter.

- A parameter is not specified depending on the value of another parameter. For example, check that the `SYSUNITS` parameter is not specified unless the `FUNC=DELETE` was also specified.
IATZPCHK

- A required parameter is specified depending on the value of another parameter. For example, if FUNC=DELETE was specified, verify that the SYSUNITS parameter was also specified.
- If a register was specified, that a valid register value was used.
- If the parameter was specified, that it contains the correct number of subparameters.

Syntax

```
label
IATZPCHK PARM=parm, VALUE=value

REXS=(reg)

SUBPAML_COUNT=(number)

LIST=(parm)

REQ= Processing:

  NO

  YES

No REQ= Processing:

XCHECK PARM=name, XCHECK VALUE=value

XCHECK LIST Processing:

XCHECK REQ_LIST=(parm)

XCHECK OPT_LIST=(parm)

Parameters

PARM=
  Specifies the name of the parameter to check.
VALUE=
  Specifies the value set that was set for the parameter identified by PARM=. This value is used to determine whether a value was specified for the parameter identified by PARM=.

REGS=
  Specifies which registers are allowed to be used or not allowed to be used in the checked parameter. To specify a list of registers that are allowed to be used, each register should be specified in the REGS parameter. To specify registers 2, 3, 4, and 5 as the only registers allowed for a parameter, code the following:
  REGS=(R2,R3,R4,R5)

To specify a list of registers that are not allowed to be used, specify "NOT" as the first subparameter, followed by the list of registers in the subsequent subparameters. To specify registers 0 and 1 as not allowed to be used for a parameter, code the following:
  REGS=(NOT,R0,R1)

Note: The REGS parameter assumes that registers are coded in the form PARM=(Rx) and not PARM=(x).

SUBPARM_COUNT=
  Specifies the number of subparameters that the parameter must have. A single value or multiple values can be specified. Null subparameters are not allowed. To specify that a parameter must have two, non-null subparameters, code the following:
  SUBPARM_COUNT=2

To specify that a parameter must have two or three, non-null subparameters, code the following:
  SUBPARM_COUNT=(2,3)

LIST=
  Specifies a list of valid values for the parameter. The VALUE you specify on PARM= must be one of these values or an MNOTE is issued.

If LIST= is not specified, the PARM= is not checked for valid values.

REQ=YES | NO
  Specifies whether the parameter identified by PARM= is required (YES) or optional (NO).

Notes:
1. If REQ=YES is specified and LIST= is not specified, the macro only checks whether a value was specified.
2. You cannot specify REQ= if you specify XCHECK_PARM=
3. If you specify XCHECK_PARM=, JES3 uses the XCHECK_REQ_LIST= and XCHECK_OPT_LIST= parameters to determine when you must specify the parameter identified by PARM=.

Examples Using PARM, VALUE, REQ, and LIST
  To make sure that a parameter FUNC is specified without regard to the values that were specified, code the following:
  IATZPCHK PARM=FUNC,
  VALUE=&FUNC,,
  REQ=YES
To make sure that a parameter FUNC is specified and is set to one of the following values, code the following:

```
IATZPCHK PARM=FUNC,
    VALUE=&FUNC,;
    REQ=YES
    LIST=(ADD,DELETE,INIT,TERMINATE)
```

To make sure that if parameter FUNC is specified, it is set to one of the following values, code the following:

```
IATZPCHK PARM=FUNC,
    VALUE=&FUNC,;
    REQ=NO
    LIST=(ADD,DELETE,INIT,TERMINATE)
```

**XCHECK_PARM=**

Specifies the name of the parameter that is to be cross checked against the PARM= parameter. XCHECK_PARM is used when the parameter identified by PARM=:

- Can only be specified when XCHECK_PARM is a certain value.
- Must be specified when XCHECK_PARM is a certain value.

**Example 1:**

To verify that a parameter, SYSUNITS, is only specified for FUNC=DELETE or FUNC=CHANGE, code the following (assuming that the SYSUNITS parameter is optional for FUNC=DELETE and FUNC=CHANGE):

```
IATZPCHK PARM=SYSUNITS,
    VALUE=&SYSUNITS,;
    XCHECK_PARM=FUNC,
    XCHECK_VALUE=&FUNC,;
    XCHECK_OPT_LIST=(DELETE,CHANGE),
```

**Example 2:**

To verify that a parameter, SYSUNITS, is only specified for FUNC=DELETE or FUNC=CHANGE, code the following (assuming that the SYSUNITS parameter is required for FUNC=DELETE and FUNC=CHANGE):

```
IATZPCHK PARM=SYSUNITS,
    VALUE=&SYSUNITS,;
    XCHECK_PARM=FUNC,
    XCHECK_VALUE=&FUNC,;
    XCHECK_REQ_LIST=(DELETE,CHANGE),
```

**Example 3:**

To verify that a parameter, SYSUNITS, is required for FUNC=DELETE and optional for FUNC=CHANGE: code the following:

```
IATZPCHK PARM=SYSUNITS,
    VALUE=&SYSUNITS,;
    XCHECK_PARM=FUNC,
    XCHECK_VALUE=&FUNC,;
    XCHECK_REQ_LIST=DELETE,
    XCHECK_OPT_LIST=CHANGE
```

**Notes:**

1. XCHECK_PARM= is mutually exclusive with REQ=.
2. If you code this parameter, XCHECK_VALUE= is required, and you must also specify at least one of XCHECK_REQ_LIST= or XCHECK_OPT_LIST=. 
**XCHECK_VALUE=**

Specifies the XCHECK_PARM value that is to be cross checked against the value associated with the PARM= parameter.

If you code this parameter, XCHECK_PARM= is required, and you must also specify at least one of XCHECK_REQ_LIST= or XCHECK_OPT_LIST=.

**XCHECK_REQ_LIST=**

Specifies a list of valid values for the parameter represented by XCHECK_PARM. When the parameter represented by XCHECK_PARM is one of the values in this list, then the parameter represented by PARM= must be specified.

For example, to verify that a parameter, INDEX, is specified when FUNC=DIRECT_ACCESS is specified, code the following:

```
IATZPCHK PARM=INDEX,
  VALUE=&INDEX,;
  XCHECK_PARM=FUNC, 
  XCHECK_VALUE=&FUNC,;
  XCHECK_REQ_LIST=DIRECT_ACCESS
```

**XCHECK_OPT_LIST**

Specifies a list of valid values for the parameter represented by XCHECK_PARM=. When the parameter represented by XCHECK_PARM= is one of the values in this list, then the parameter represented by PARM= is allowed to be specified but is optional.

For example, to allow a parameter, INDEX, to be optionally specified when FUNC=DIRECT_ACCESS is specified, code the following:

```
IATZPCHK PARM=INDEX,
  VALUE=&INDEX,;
  XCHECK_PARM=FUNC, 
  XCHECK_VALUE=&FUNC,;
  XCHECK_OPT_LIST=DIRECT_ACCESS
```

**Linkage**

This macro generates inline code.

**Restrictions**

Only macro definitions (the macro code in the JES3 macro library) may use this macro.

**Error Conditions**

If any errors occur, JES3 sets the following global variables, which can be checked by your macro. These variables should be declared as global bit variables.

- **PCHKERR**: This variable is used by the calling macro to determine whether any errors were detected by IATZPCHK for the current parameter. The calling macro does not have to initialize PCHKERR to ‘0’B before invoking the IATZPCHK macro.
- **PCHKANY**: This variable is used to determine whether any IATZPCHK errors occurred in this assembly for the calling macro. This allows the calling macro to check for errors on any of the parameters that use the IATZPCHK without having to manage its own locally defined error indicator. Before a macro uses IATZPCHK for the first time, it should set this variable to ‘0’B.
IATZTYPE

**IATZTYPE (Inner Macro to Generate Data Areas)**

The IATZTYPE inner macro generates data areas.

**Syntax**

```
label  IATZTYPE type, name
```

**Parameters**

- **type**
  Specifies the type of data area to be generated by indicating the type of boundary for the data area. Valid types are DSECT, CSECT, X, H, F, and D, where X means a character boundary, H means halfword, F means fullword, and D means doubleword. If you do not specify a type, DSECT is assumed.

  If you specify CSECT for the type, be sure to remember to include the appropriate AMODE and RMODE statements related to the CSECT name.

- **name**
  Specifies the label for the generated instruction.

An example of this macro is: IATZTYPE H,LABEL3

which generates the statement: LABEL3 DC 0H'0'

**Linkage**

This macro does not generate executable code.

**Restrictions**

Only macro definitions (the macro code in the JES3 macro library) may use this macro.

---

ICARDRD (Read Initialization Statement)

The JES3 initialization modules use the ICARDRD macro to read each initialization statement.

Upon return, the record is contained in field INAREA, and field INPOINT is set to the address of the first non-blank character within field INAREA.

**Syntax**

```
label  ICARDRD address
```

**Parameters**

- **NORMAL=**
  This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.
**Linkage**

This macro uses ASAVE linkage.

**Restrictions**

Only the JES3 initialization modules may use this macro.

---

**ICONVBIN (Convert to Binary)**

The JES3 initialization modules use the ICONVBIN macro to convert a data field from decimal to binary format. The data to be converted must be in field SOPER and its length in field LOPER of data CSECT IATYINT.

Upon return, register 1 contains the converted number.

**Syntax**

```
ICONVBIN label address
```

**Parameters**

**NORMAL**

This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

---

**Linkage**

This macro uses ASAVE linkage.

**Restrictions**

Only the JES3 initialization modules may use this macro.

---

**ICONVHEX (Convert to Hexadecimal Format)**

The JES3 initialization modules use the ICONVHEX macro to convert a data field from decimal to hexadecimal format. The data to be converted must be in field SOPER and its length in field LOPER of data CSECT IATYINT.

Upon return, register 1 contains the converted number.

**Syntax**

```
ICONVHEX label address
```

**Parameters**

**NORMAL**

This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.
INITMWLE (Scan for Ending Right Parenthesis)

The JES3 initialization modules use the INITMWLE macro to scan for the ending right parenthesis of a control statement parameter. The starting address of the scan is in field INPOINT. These fields are maintained by the ICARDRD, ISCAN1, and ISCAN2 macro routines.

INPOINT is updated to contain the address of the byte following the right parenthesis, unless that byte contains a comma, in which event the address is incremented past the comma.

Syntax

```
/SM590000/SM590000
```

Parameters

**NORMAL=**

This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

Linkage

This macro uses ASAVE linkage.

Restrictions

Only the JES3 initialization modules may use this macro.

INTERCOM (Simulate Input of Operator Message)

The INTERCOM macro simulates operator input of a console message or command.

Syntax

```
/SM590000/SM590000
```

ICONVHEX
Parameters

**CNDB=**
Specifies the address of console destination block which contains the command issuer data (console identifier, console name, command and response token (CART)). When you specify a console identifier of 0 in the CNDB, the issuer receives master command authority. Entries in the hardcopy log for the command have the name INTERNAL associated with them. IF CNDB= is not specified, the default dummy CNDB (with console id 0) is used. R0 is used for the CNDB address.

**TEXT=**
Specifies the address of the message. The first byte of the message must contain the text length, in the range from 1 to 126. R0 may not be used to specify TEXT=.

**PREFIX=**

**YES**
Specifies that an 8-byte prefix is to be displayed before the message. The prefix must immediately precede the message. (The one-byte message length field that precedes the message text is considered part of the message.)

**NO**
Specifies that the name of the DSP issuing the macro will be displayed at the beginning of the message.

**MSG=**
Specifies whether the message is to be displayed on the specified console. In either case, the message will appear on the log.

**ASYNCH=**
Specifies whether the command is to be processed asynchronously or synchronously. If ASYNCH=NO is specified, the caller will not receive control back until the command is issued. If ASYNCH=YES is specified (or allowed to default) the caller will receive control back immediately. ASYNCH=NO is required to ensure that a group of commands. (for example: Commands submitted through a DISKRDR member are processed in FIFO order.)

**CHK=**
Specifies whether console authority checking is required for the message. If you specify CHK=YES, the TOKEN= parameter must also be coded.

**TOKEN=**
Specifies the address of the operator command security token associated with the specified console id. One way you can obtain this token's value is by issuing the IATXSEC macro, specifying REQUEST=VERIFYX. See IATXSEC.
INTECORM

(Security Authorization Services)* on page 429 for more information about coding the IATXSEC macro. R0 and R1 may not be used to specify TOKEN=.

**NORMA**
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

**Linkage**
This macro uses ASAVE linkage.

**Restrictions**
Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task.

**IPURGE (Recover Spool Space)**
JES3 spool initialization modules use the IPURGE macro to reallocate spool space that was previously allocated to JES3 or to a job. Specifically, this macro reallocates either:

- A single track table (STT) single record file
- All track groups previously assigned to a job or spin-off data set

The routine that issues the IPURGE macro must first issue the IATXVAL macro, to ensure that the FDB specified by the FDB parameter is valid. For the FDB of a job or data set TAT, the IATXVAL service routine leaves the TAT chain in storage. If the IATXVAL service routine takes an error return (that is, the return indicated by the ERROR or REPL parameter of the IATXVAL macro), the calling routine must not issue the IPURGE macro.

**Syntax**

```
IPURGE label, FDB=(R1)(reg), ERROR=address
```

**Parameters**

**FDB**
Specifies the address of the FDB.

**ERROR**
Specifies the address to which control returns if the spool space described by the FDB has already been recovered.

**NORMAL**
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

**Linkage**
This macro uses ASAVE linkage.
Restrictions

This macro may be used only by the JES3 spool initialization modules.

ISCAN1 (Locate Fields on Initialization Statement)

The JES3 initialization modules use the ISCAN1 macro to isolate the fields while scanning an initialization statement. The starting address for the scan is in field INPOINT of data CSECT IATYINT.

Upon return, INPOINT is incremented to the next scan address.

Syntax

```
ISCAN1--EOD=address
```

```
,TYPE=KEY,CARD
,NORMAL=address
```

```
,EXTOPER=: NO,YES
```

Parameters

EOD=

Specifies the address to which the routine returns if the end of the statement has been reached.

TYPE=

- **KEY**
  Specifies that the scan is for a keyword and either a single operand or the first operand of a string. The keyword is returned in field SPARAM, the first or only operand in field SOPER, and the length of the operand in field LOPER.

- **CARD**
  Specifies that the scan is for the positional parameter that identifies the statement. The statement name appears in field SPARAM. SPARAM and SOPER are 16-byte fields in which the data is left-justified and padded with blanks.

EXTOPER=

This parameter allows you to scan for extended operands in initialization parameters. Extended parameters are those longer than 15 bytes. NO indicates that JES3 will not scan for extended parameters, and will issue an error message if such a parameter is found. YES allows you to scan for parameters that are longer than 15 bytes, but not longer than 50 bytes.
If you allow extended operands, the first operand is returned in a 50-byte field called EXTOPER instead of in field SOPER. EXTOPER=YES is not allowed with TYPE=CARD.

NORMAL=
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

Linkage
This macro uses ASAVE linkage.

Restrictions
Only the JES3 initialization modules can use this macro.

ISCAN2 (Locate Subfield of Keyword Parameter)

The JES3 initialization modules use the ISCAN2 macro to locate the second or subsequent subfield of a keyword parameter within an initialization statement. The starting address for the scan is in field INPOINT of data CSECT IATYINT.

Upon return, INPOINT is incremented to the next scan address. The subfield is in field SOPER and its length in field LOPER. SOPER is a 16-byte field in which the data is left-justified and padded with blanks.

Syntax

```
ISCAN2 LABEL EOD=ADDRESS, NORMAL=ADDRESS
```

Parameters

EOD=
Specifies the address to which the routine returns if the end of the keyword parameter has been reached.

NORMAL=
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

Linkage
This macro uses ASAVE linkage.

Restrictions
Only the JES3 initialization modules can use this macro.

ISORT (Sort Internal Table Entries)

The JES3 initialization modules use the ISORT macro to sort a group of internal table entries.
Syntax

\[
\text{ISORT-\textit{LIST}= (R1)}
\]

Parameters

\text{LIST=}  
Specifies the address of the list describing the entries to be sorted. The list is mapped by the IATYSOR macro.

\text{NORMAL=}  
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

Linkage

This macro uses ASAVE linkage.

Restrictions

Only the JES3 initialization modules may use this macro.

ITREAD (Read the Intermediate Text for JES3 Initialization)

The JES3 initialization modules use the ITREAD macro to read the intermediate text from the spool data set. The intermediate text results from the first phase of translating the initialization stream and is used as input to the second phase.

Syntax

\[
\text{ITREAD-\textit{SPOOL}= (R1)}
\]

Parameters

\text{SPOOL=}  
Specifies the address of a list (in the form generated by the IATYSPL mapping macro) defining the set of table entries to be read.

\text{EOD=}  
Specifies the address to which the routine returns when the end of the table has been reached.

\text{ERROR=}  
This parameter specifies the address to which the macro processing routine returns if an error occurs. When control is passed to the address, register 0 contains a dump code and register 15 contains a return code, unless otherwise specified.

If an error occurs and ERROR was not specified, the dynamic support program (DSP) will be ended by using the FAILDSP macro.
ITREAD

NORMAL=
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

Linkage
This macro uses ASAVE linkage.

Restrictions
Only the JES3 initialization modules may use this macro.

ITWRITE (Write the Intermediate Text for JES3 Initialization)

The JES3 initialization modules use the ITWRITE macro to write the intermediate text to the spool data set. The intermediate text results from the first phase of translating the initialization stream and is used as input to the second phase.

Syntax

```
[label] ITWRITE SPOOL= (R1) address, LENGTH= (R0) address
```

```
,ADDR= address, ,NORMAL= address
```

Parameters

SPOOL=
Specifies the address of a list (in the form generated by the IATYSPL mapping macro) defining the set of table entries to be written.

LENGTH=
Specifies the length of each entry in the table. This length field must be supplied to the ITWRITE routine, either by use of this parameter, by an assembled value in the spool list, or by storing the value in the length field before this macro is issued.

ADDR=
Specifies the address of the entry to be written. This address field must be supplied to the ITWRITE routine, either by use of this parameter, by an assembled value in the spool list, or by storing the value in the address field before this macro is issued.

NORMAL=
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

Linkage
This macro uses ASAVE linkage.

Restrictions
Only the JES3 initialization modules may use this macro.
IWASPOUT (Write Initialization Error Message)

The JES3 initialization modules use the IWASPOUT macro to write messages to the JES3OUT data set and, optionally, to the operator console, describing errors in an initialization statement.

For a summary of JES3 macros involved in message processing, see the section "Choosing the Appropriate Macro to Issue a Message" on page 42.

Note: JES3 insures that it sends all critical error messages (that is, for unrecoverable errors) issued by the IWASPOUT macro to the operator’s console by defaulting the WTO= parameter to Y when the ERRLVL= parameter specifies C.

Syntax

```
label
IWASPOUT—ERRMSG=(R1)
address,ERRLVL=(C,E,W,N),WTO=(Y,N),COMPRESS=(Y,N),NORMAL=address
```

Parameters

**ERRMSG=**

Specifies the address of the error message to be issued.

**ERRLVL=**

Specifies the error level for the message:

- **C** Unrecoverable – Indicates that execution beyond initialization is impossible. JES3 will end at the end of initialization. C is the default for ERRLVL=.
- **E** Error – Indicates that a failure is likely within JES3, but JES3 will attempt to continue.
- **W** Warning – Indicates that a JES3 function may be impacted.
- **N** No error – Indicates a message is to be issued without any error level flag being set.

**WTO=**

Specifies whether a WTO is to be issued for this message.

- **Y** A WTO is to be issued. This is the default parameter when ERRLVL= is omitted or specified with C. When JES3 creates the WTO macro, it assigns the SUBSMOD=NO parameter to the WTO.
- **N** A WTO is not to be issued. This is the default parameter when ERRLVL= is specified without C.

**COMPRESS=**

Specifies whether duplicate blanks in the message should be compressed (deleted) before the message is written to JES3OUT, the operator console, or both.
Duplicate blanks should not be compressed.
Y  Duplicate blanks should be compressed.

NORMAL=
Specifies the address to which the routine returns upon normal completion of macro processing. If you omit the NORMAL= parameter, control returns to the next sequential instruction after the macro.

Linkage
This macro uses ASAVE linkage.

Restrictions
Only the JES3 initialization modules may use this macro.

JDSADD (Add New Entry to JDS)
The JDSADD macro adds a new entry to the job data set block (JDS).

On return, register 0 contains information that may be used with a JDSPOINT macro. The entire JDS is held upon return from the JDSADD macro. The JDS cannot be accessed by another DSP until a JDSREL macro is issued.

Syntax

Parameters
ENTRY=
Specifies the address of the entry to be added.

RQ=
Specifies the address of the resident job queue (RESQUEUE) entry for the JDS to be accessed. If omitted, the FCT pointer to the RESQUEUE is used.

BUSY=
Specifies the address to which the routine returns if another function is accessing the JDS. This macro returns an ECF mask and address which may be used to wait until the JDS becomes available. The 31-bit ECF address is returned in register 0. The ECF mask needed is contained in the low-order byte of register 1, with the other three bytes containing zeros.

If this parameter is omitted, the routine will wait until the JDS may be accessed.

INSERT=
YES
Specifies that the new entry is to be added to the JDS in the first unused entry.
NO

Specifies the new entry is to be added at the end of the last JDS buffer.

NORMAL=

This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

Linkage

This macro uses ASAVE linkage.

Restrictions

Do not use this macro in any routine that might run under the JES3 auxiliary task.

JDSGET (Access Entry in JDS)

The JDSGET macro accesses the next (or first) entry in the job data set block (JDS) of the type and ddname or dsname specified.

On return, register 1 contains the address of the JDS entry and register 0 contains information which may be used with a JDSPOINT macro.

The entire JDS is held upon return from the JDSGET macro. The JDS cannot be accessed by another DSP until a JDSREL macro is issued.

Syntax

```
/JDSGET label EOF=address,NAVAIL=address,BUSY=address,RQ=(R0)address,TYP"(type,type)" SYSIN,TYPE=(type,typ"),DDNAME=(R1)address,DDNAME=(R1)address,DSNAME=(R1)address,DSNAME=(R1)address,DSNUM=(R1)address,DSNUM=(R1)address,SCANSTR=TOP,NEXT,SPINSEL=no,SPINSEL=yes
```

Parameters

EOF=

Specifies the address to which the routine returns when the end of the JDS is reached.
NAVAIL=
  Specifies the address to which the routine returns when the JDS entry
  requested is held by another DSP. If omitted, the routine will wait until the entry
  is available.

BUSY=
  Specifies the address to which control returns if another function is accessing
  the JDS. This macro returns an ECF mask and address which may be used to
  wait until the JDS becomes available. The 31-bit ECF address is returned in
  register 0. The ECF mask needed is contained in the low-order byte of register
  1, with the other three bytes containing zeros.

  If this parameter is omitted, the routine will wait until the JDS may be accessed.

RQ=
  Specifies the address of the resident job queue (RESQUEUE) entry for the JDS
  to be accessed. If omitted, the address of the RESQUEUE contained in the
  FCT is used.

TYPE=
  (type [,type]...)
  Specifies the data set type (PRINT, PUNCH, OUTPUT, TAT, USER1,
  USER2, RSVD, or HOLD) of the required entry. One or more types may be
  coded.

SYSIN
  Specifies any input data set.

DDNAME=
  Specifies the address of a 24-byte field containing the fully qualified ddname of
  the data set required.

DSNAME=
  Specifies the address of a 40-byte field containing the fully qualified data set
  identifier (dsname) of the data set required. The data set identifier consists of
  the following five 8-character qualifiers:
  
  user id
  jobname
  job id
  data set number
  data set name

  You can specify either DDNAME= or DSNAME=, but not both.

DSNUM=
  Specifies the address of an 8-byte field containing the data set number field in
  EBCDIC.

SCANSTR=
  NEXT
    Specifies that the next JDS entry is to be obtained.
  TOP
    Specifies that the first JDS entry is to be obtained.

NORMAL=
  This parameter specifies that after the macro processing routine executes, the
  issuing program can resume execution at an instruction other than the
  instruction that immediately follows the macro.

SPINSEL=

JDSGET
YES
Specifies that only unprocessed SPINOFF JDS entries are returned.

NO
All SPINOFF JDS entries will be returned regardless of whether they have
been processed. No is the default.

Linkage
This macro uses ASAVE linkage.

Restrictions
Do not use this macro in any routine that might run under the JES3 auxiliary task.
This macro uses register 2.

JDSHOLD (Hold JDS Entry)
The JDSHOLD macro holds the entire JDS or one specific entry.

On return, register 0 contains information which may be used with a JDSPOINT
macro. The JDS cannot be accessed by another DSP until a JDSREL macro is
issued.

Syntax

Parameters

EOF=
 Specifies the address to which the routine returns when the end of the JDS is
reached. The EOF parameter is required if TYPE or DDNAME is used. EOF
and NAVAIL are ignored if TYPE and DDNAME are omitted.
JDSHOLD

NAVAIL=
   Specifies the address to which the routine returns when the requested JDS entry is held by another DSP. If omitted, the routine will wait until the entry is available.

BUSY=
   Specifies the address to which control returns if another function is accessing the JDS. This macro returns an ECF mask and address which may be used to wait until the JDS becomes available. The 31-bit ECF address is returned in register 0. The ECF mask needed is contained in the low-order byte of register 1, with the other three bytes containing zeros.

   If this parameter is omitted, the routine will wait until the JDS may be accessed.

RQ=
   Specifies the address of the resident job queue (RESQUEUE) entry for the JDS to be accessed. If omitted, the address of the RESQUEUE contained in the FCT is used. If this parameter is specified, the EOF parameter must be specified.

TYPE=
   type[,type]
      The data set type (PRINT, PUNCH, OUTPUT, TAT, USER1, USER2, RSVD, or HOLD) of the required entry. One or more types may be coded.

SYSIN
   Specifies any input data set.

DDNAME=
   Specifies the address of the 24-byte field containing the fully-qualified ddname of the data set required. If this parameter is specified, the EOF parameter must be specified.

NORMAL=
   This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

Linkage

This macro uses ASAVE linkage.

Restrictions

Do not use this macro in any routine that might run under the JES3 auxiliary task. This macro uses register 2.

JDSPOINT (Reobtain JDS Entry)

The JDSPOINT macro reobtains one specific entry in the JDS.

On return, register 1 contains the address of the JDS entry and register 0 contains the note value which may be used with a subsequent JDSPOINT macro.

The entire JDS is held upon return from the JDSPOINT macro. The JDS cannot be accessed by another DSP until a JDSREL macro is issued.
JDSPOINT

Syntax

```
JDSPOINT label, NAVAIL=address, BUSY=address
/FCTRQAD/, RQ=(R0) address
/POINTER=(R1) address
/ERROR=address
/NORMAL=address
```

Parameters

**NAVAIL=**
Specifies the address to which control returns when the requested JDS entry is held by another DSP. If omitted, the routine will wait until the entry is available.

**BUSY=**
Specifies the address to which control returns if another function is accessing the JDS. This macro returns an ECF mask and address which may be used to wait until the JDS becomes available. The 31-bit ECF address is returned in register 0. The ECF mask needed is contained in the low-order byte of register 1, with the other three bytes containing zeros.

If this parameter is omitted, the routine will wait until the JDS may be accessed.

**RQ=**
Specifies the address of the resident job queue (RESQUEUE) entry for the JDS to be accessed. If omitted, the address of the RESQUEUE contained in the FCTRQAD field of the FCT is used.

**POINTER=**
Specifies the note value returned by one of the JDS-access macros.

**ERROR=**
Specifies the address to which control returns if POINTER is incorrect.

**NORMAL=**
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

Linkage

This macro uses ASAVE linkage.

Restrictions

Do not use this macro in any routine that might run under the JES3 auxiliary task.

JDSPUT (Checkpoint Change in Current JDS Entry)

The JDSPUT macro checkpoints a change in the current JDS entry. The entry to be checkpointed must have been accessed by using a JDSGET or JDSPOINT macro.
JDSPUT

On return, register 0 contains information which may be used with a JDSPORT
macro. The entire JDS is held upon return from this macro. The JDS cannot be
accessed by another DSP until a JDSREL macro is issued.

To reduce the amount of I/O activity that occurs when JES3 must process multiple
JDSPUT macros for the same JDS, specify the TYPE=DEFER parameter on this
macro. The TYPE=DEFER parameter causes JES3 not to write the entire JDS each
time a single entry changes. Instead, the entry will be flagged, and the entire JDS
will be written only once – when a JDSREL signifies that JDS processing is
finished.

Syntax

```
JDSPUT
   label
      RQ= (R0)
      address
      ,TYPE= DEFER
   ,NORMAL= address
```

Parameters

RQ=
   Specifies the address of the resident job queue (RESQUEUE) entry for the JDS
to be checkpointed. If omitted, the address of the RESQUEUE contained in the
FCTRQAD field of the FCT is used.

TYPE=
   IMMED
      Specifies that the current JDS entry is to be checkpointed (written to spool).
      This is the default value.
   DEFER
      Specifies that the current JDS entry is to be flagged for checkpoint when a
      JDSREL macro is issued to write the JDS.

NORMAL=
   This parameter specifies that after the macro processing routine executes, the
   issuing program can resume execution at an instruction other than the
   instruction that immediately follows the macro.

Linkage

This macro uses ASAVE linkage.

Restrictions

Do not use this macro in any routine that might run under the JES3 auxiliary task.

JDSREL (Release Control of JDS or Negate Entry Hold)

The JDSREL macro releases control of the JDS or negates the hold of a specific
entry.

On return, register 1 contains the address of the JDS entry and register 0 contains
information which may be used with a JDSPORT macro.
Syntax

Parameters

RQ=
Specifies the address of the resident job queue (RESQUEUE) entry for the JDS to be accessed. If omitted, the address of the RESQUEUE contained in the FCTRQAD field of the FCT is used.

DDNAME=
Specifies the address of a 24-byte field containing the fully-qualified ddname of the data set required. If DDNAME is coded, the current entry is released if the ddname matches; otherwise, the first occurrence of the ddname which is held by the caller is released. If DDNAME and TYPE are specified, TYPE is ignored. If DDNAME is specified, then the EOF parameter must be specified. If the DDNAME parameter is specified, the entire JDS is held upon return from this macro. A JDSREL macro without the DDNAME parameter specified must be used to release the entire JDS.

EOF=
Specifies the address to which control returns when the end of the JDS is reached.

TYPE=

DSP
ALL

Specifies that all entries held by the caller are to be released. If TYPE=ALL is specified, the BUSY parameter must be specified.

BUSY=
Specifies the address to which control returns if one or more JDS entries are held by another function. This parameter is required when TYPE=ALL is specified.

NORMAL=
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

Linkage

This macro uses ASAVE linkage.
Restrictions

Do not use this macro in any routine that might run under the JES3 auxiliary task. This macro uses register 2.

JESCKPNT (Save JES3 Checkpoint Data)

The JESCKPNT macro saves basic JES3 data (for example, job queue pointers) whenever any part of that data changes.

Syntax

```
label JESCKPNT ,NORMAL=address
```

Parameters

**NORMAL=**

This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

Linkage

This macro uses ASAVE linkage.

Restrictions

Do not use this macro in any routine that might run in a C/I FSS address space, or under the JES3 auxiliary task, or in user code during initialization.

JESCLOSE (Close Unit Record or Tape Device)

The JESCLOSE macro closes a unit record, tape or BSC RJP device. If the device is a tape, no writing of tape marks or tape positioning is performed.

Syntax

```
label JESCLOSE--IOB=(R1)--address ,NORMAL=TYPE=NORMAL
```

Parameters

**IOB=**

Specifies the address of the IOB of the device to be closed. The IOB must reside below 16 megabytes in virtual storage.

**TYPE=**

- **NORMAL**
  - Specifies a normal close.
- **T**
  - Specifies a temporary close.
NORMAL=
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

Note: contains an example of the JESCLOSE macro.

Linkage

This macro uses ASAVE linkage.

Restrictions

Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task.

JESEXCP (Request I/O for Device)

The JESEXCP macro requests I/O for a specific tape, unit record, or BSC RJP device.

Data chaining is supported only for print devices, while command chaining is supported for all output devices.

Syntax

```
JESEXCP  label  IOB=(R1) address  IOBE=(R0) address
 assume RJPCAN=address, TYPE=V, NORMAL=address
```

Parameters

IOB=
Specifies the address of the IOB associated with the I/O request. The IOB must reside below 16 megabytes.

IOBE=
Specifies the address of the IOB extension (IOBE) associated with the I/O request. The IOBE must reside below 16 megabytes.

RJPCAN=
Specifies the address to which the routine returns if a cancel command is issued from a remote work station.

TYPE=

```
  R  Specifies that the EXCPVR macro is to be used. All CCW addresses must be real and all data areas must be fixed. All channel programs (CCWs) and data areas must reside below 16 megabytes in real and virtual storage.
    V  Specifies that the EXCP macro is to be used. All channel programs (CCWs) and data areas must reside below 16 megabytes in virtual storage.
```

NORMAL=
This parameter specifies that after the macro processing routine executes, the
issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

Note: 30 shows an example of the JESEXCP macro.

Linkage

This macro uses ASAVE linkage.

Restrictions

Prior to this request, the device must have been opened using the JESOPEN macro. Do not use this macro in any routine that might run in a C/I FSS address space.

JESMSG (Place Message in JESMSG Data Set)

When your system detects an error in JCL or a JES3 control card, you can use the JESMSG macro in the appropriate installation exit routine to time stamp a message and write it to a job’s JESMSG data set.

The JESMSG macro uses a job’s RESQUEUE to locate the appropriate JESMSG data set entry. However, because the RESQUEUE for a job is not created until that job has completed input service and has been scheduled by JSS, the JESMSG macro cannot be used by installation exits during input service processing. This limitation of the JESMSG macro affects users who need to save error messages related to jobs that get flushed during input processing because of JCL or JES3 control card errors detected in installation exits.

To write a message to the JESMSG data set during input processing, you can use the IATXISMG macro. See "IATXISMG (Write a Message to the JESMSG Data Set During Input Service)" on page 317.

Note: You do not need to code the JESMSG macro if you have coded the MESSAGE macro with the JOBNAME= parameter set to a value other than JES3. Coding the MESSAGE macro in this form ensures that the message is written to the JESMSG data set.

For a summary of JES3 macros involved in message processing, see the section "Choosing the Appropriate Macro to Issue a Message" on page 42.

Syntax

```
/s/SM590000/SM590000
JESMSG label=RQ=(R0) address
```

```
/s/SM590000
STD TYPE=TEXT=NAVAIL=address
```

```
MULT TEXT=NAVAIL=address
```

```
CLOSE NAVAIL=address
```

```
ASYNCH TEXT=BUSY=address
```
Parameters

**RQ=**
Specifies the address of the RESQUEUE entry for the job.

**TEXT=**
Specifies the address of the message. The first byte of the message must contain the length of the message text. The length must not exceed 120 characters.

**TYPE=**

- **STD**
  Specifies that only one message is to be written into JESMSG. After the message is written, the JESMSG data set is closed. If several messages are to be added in succession, the MULT and CLOSE forms should be used to improve performance.

- **MULT**
  Specifies that this is one in a series of messages being added to the JESMSG data set. You must issue a TYPE=CLOSE after the last TYPE=MULT.

  Be sure to close the data set with TYPE=CLOSE or TYPE=STD after you have completed adding messages.

- **CLOSE**
  Specifies that the JESMSG data set is to be closed. If you specify TYPE=CLOSE, you can not specify the TEXT= parameter.

- **ASYNCH**
  Specifies that the message is to be queued for the CONSDM FCT.

**NAVAIL**
Specifies the address to which the routine returns if the JESMSGLG data set is locked. NAVAIL is only for the following requests:
- TYPE=STD
- TYPE=MULT
- TYPE=CLOSE
- TYPE=MSG_CLEANUP

**BUSY=**
Specifies the address to which the routine returns if the job data set block (JDS) is in use. This macro returns an ECF mask and address which may be used to wait until the JDS becomes available. The 31-bit ECF address is returned in
register 0. The ECF mask needed is contained in the low-order byte of register 1, with the other three bytes containing zeros.

If this parameter is omitted, the routine will wait until the JDS may be accessed.

**NORMAL=**
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

**ERROR=**
Specifies the address to which the routine returns if an error is detected. If you omit the ERROR= parameter, the calling DSP fails. Register 15 contains a return code indicating the type of error:
- **X'04'** – The job has no JDS.
- **X'08'** – EOF returned from JDSGET or JDSHOLD.
- **X'0C'** – JESMSG text length is incorrect.
- **X'10'** – JSAM error.
- **X'14'** – An error occurred during ASYNCH processing.
- **X'18'** – Request is from an FSS address space.
- **X'1C'** – Input control block is not a RQ or JCT.
- **X'20'** – MPC not found.

**LOG=**
- **YES**
  Specifies that messages should be written to JESMSGLG for this job.
- **NO**
  Specifies that messages should be suppressed from JESMSGLG for this job.

**JCT=**
Specifies the address of the JCT for the job.

## Linkage

This macro uses ASAVE linkage.

## Restrictions

Do not use this macro in any routine that might run under the JES3 auxiliary task. This macro uses register 2. RQ and JCT are mutually exclusive.

## JESOPEN (Open a Unit Record or Tape Device)

The JESOPEN macro opens a unit record, tape, or RJP device.

### Syntax

```
label JESOPEN.IOB=(RO),SUPUNIT=address,NORMAL=address
```
Parameters

**IOB**=
Specifies the address of the IOB of the device to be opened. The IOB must be below 16 megabytes.

**SUPUNIT**=
Specifies the address of the SUPUNIT entry for the device to be opened. The address is found in the GETUNIT list entry for the device.

**NORMAL**=
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

**Note:** See shows an example of the JESOPEN macro.

Linkage

This macro uses ASAVE linkage.

Restrictions

Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task.

JESREAD (Read Single Record File)

The JESREAD macro reads a single record file (SRF).

After accessing a control block, such as a JDAB, the ARELEASE or IATXRELCE macro should be used when a control block is read into storage, and has not been modified, and is no longer needed in storage. If a control block has been modified, the AWRITE or WRTCHAIN macro should be used to update the control block on the spool.

Syntax

```
JESREAD FDB=address, ID=name, (R1), (R0), reg, (label)
```

```
AWAIT=NO, BUSY=address, ERROR=address, NORMAL=address
```

Parameters

**FDB**=
Specifies the address of the FDB of the SRF to be read. The file must reside on a spool data set when the JESREAD macro is issued; otherwise, the action specified by the AWAIT parameter is taken.

**ID**=
Specifies the name to be associated with the SRF. The ID length is limited to
JESREAD

four characters. The name you specify is checked against the record identifier in the requested record. If it does not match, the request fails. The ID= parameter is required.

AWAIT=

   YES
   Specifies that JESREAD should wait for the I/O to complete.

   NO
   Specifies that JESREAD should not wait for I/O to complete. If AWAIT=NO is specified, then the BUSY parameter is required.

BUSY=

   Specifies the address to which the routine returns if I/O is not possible. The 31-bit ECF address is returned in register 0. Register 1 contains the ECF mask in the low-order byte. The other three bytes contain zeros. If you specify the BUSY parameter, then you must also specify the AWAIT=NO parameter.

ERROR=

   This parameter specifies the address to which the macro processing routine returns if an error occurs. When control is passed to the address, register 0 contains a dump code and register 15 contains a return code, unless otherwise specified. The return codes and dump codes are described below. For a description of every dump code, see z/OS JES3 Diagnosis.

   If an error occurs and ERROR was not specified, the dynamic support program (DSP) will be ended by using the FAILDSP macro.

NORMAL=

   This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

Return Codes

Following are the return codes and dump codes returned in registers 15 and 0, respectively:

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Dump Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>08</td>
<td>721</td>
<td>Incorrect spool address</td>
</tr>
<tr>
<td>0C</td>
<td>725</td>
<td>Unrecoverable I/O error</td>
</tr>
<tr>
<td>10</td>
<td>704</td>
<td>Zero FDB</td>
</tr>
<tr>
<td>14</td>
<td>704</td>
<td>Record IDs did not match</td>
</tr>
<tr>
<td>18</td>
<td>726</td>
<td>File unavailable during initialization</td>
</tr>
<tr>
<td>1C</td>
<td>704</td>
<td>Request made for a multirecord file (MRF)</td>
</tr>
<tr>
<td>2C</td>
<td>736</td>
<td>CSBT identifier is incorrect</td>
</tr>
<tr>
<td>28</td>
<td>736</td>
<td>SRF identifier is incorrect</td>
</tr>
<tr>
<td>48</td>
<td>736</td>
<td>RCE entry not found</td>
</tr>
</tbody>
</table>

Linkage

This macro uses ASAVE linkage.

Restrictions

Do not use this macro in any routine that might run under the JES3 auxiliary task.
JESTAE (Define Abnormal Exit Routine)

The JESTAE macro defines a DSP abnormal exit routine. The JESTAE exit routine must be resident throughout the life of the JESTAE request. A DSP can issue more than one JESTAE macro. All JESTAE requests issued by programs running under the same FCT are queued so that the exit established by the most recent JESTAE request will be the first to get control. If this exit fails or requests that the abnormal termination continue, the exit established by the previous JESTAE request will get control. This process is called JESTAE percolation.

A DSP can cancel the current JESTAE request by issuing the JESTAE macro with the JESTAE exit routine address specified as zero. The JESTAE that is canceled is the one most recently created. If no JESTAE requests are active for the FCT at the time a cancel is issued, the caller will be informed that its request is incorrect by a return code of 4 in register 15.

You can specify the address of a symptom recording exit for the JESTAE exit. The symptom recording exit gets control before failsoft processing. The symptom recording exit records failure symptoms in the system diagnostic work area (SDWA) using the variable recording area (VRA). The record of JES3 failure symptoms can be helpful when trying to diagnose problems with JES3.

Syntax

```
/JESTAE label exit_address 0,PARAM=(R1)address/SYMEX=(R2)address,NORMAL=address/
```

Parameters

**exit-address**

Specifies the address of the routine to be entered if the DSP issuing this macro ends abnormally.

**0**

Specifies that the most recent JESTAE exit is to be canceled.

**PARAM=**

Specifies the address of a parameter list to be passed to the exit routine when it receives control.

**SYMEX=**

Specifies the JESTAE system recording exit. The JES3 ESTAE routine gives this exit control to add symptoms to the system diagnostic work area (SDWA), using the variable recording area (VRA).

**Linkage:** The symptom exits use BALR 14,15 linkage.

**Information supplied on entry:**

**Register 0**

JESTAE parameter. This value is specified using the PARAM= keyword on the JESTAE macro.

**Register 1**

Address of the SDWA that is being used for this failure.

**Register 14**

Return address to IATABMN
Register 16  Entry point to system recording exit

Return code in register 15 on exit:

Register 15=0  Continue processing JESTAE calls to symptom recording exit routines. A return code of 0 indicates normal completion of the symptom recording exit routine.

Register 15=4  The symptom recording exit routine determined that the data in the VRA is complete. Therefore, no additional symptom recording exit routines are called.

To make this determination, your routine needs to use a recovery design that includes footprinting of error paths and production of other key symptoms to allow the routine to know what stage of processing it is in.

Register 15=8  The data in the VRA is incorrect or incomplete. JES3 writes a message to the VRA indicating that the VRA symptom data may be incomplete or may contain errors. No additional JESTAE symptom recording exit routines will be called. The JES3 ESTAE continues recovery processing.

NORMAL=  
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

Linkage

This macro uses BALR R14,R15 linkage.

Restrictions

None.

JNADD (Add Entry to JNCB Chain)

The JNADD macro adds a job net control block (JNCB) entry to the JNCB chain. Only synchronous access is allowed to the JNCB chain.

Syntax

```
JNADD ENTRY=(R1) address \n```

Parameters

ENTRY=  
Specifies the address of the JNCB to be added to the chain.

NORMAL=  
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

Linkage

This macro uses ASAVE linkage.
Restrictions
Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task.

JNCBHLD (Hold Specified JNCB Within JNCB Chain)
The JNCBHLD macro holds a specific job net control block (JNCB) within the JNCB chain. Only synchronous access is allowed to a specific JNCB.

Upon return, register 1 contains the address of the JNCB or zero, if the JNCB was not found.

Syntax

Parameters

ID=
Specifies the address of the net identification of the JNCB to be placed in hold status.

NORMAL=
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

Linkage
This macro uses ASAVE linkage.

Restrictions
Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task.

JNCBREL (Release a JNCB from Hold)
The JNCBREL macro releases a previously held job net control block (JNCB).

Syntax

Parameters

ID=
Specifies the address of the net identification of the JNCB to be released.

NORMAL=
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.
JNDEL (Delete a JNCB from the JNCB Chain)

The JNDEL macro deletes a job net control block (JNCB) from the JNCB chain. Only synchronous access is allowed to the JNCB chain.

Syntax

```
JNDEL label ID=(R1) address, NORMAL= address
```

Parameters

- **ID=**
  Specifies the address of the net identification of the JNCB to be deleted.

- **NORMAL=**
  This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

Linkage

This macro uses ASAVE linkage.

Restrictions

Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task.

JNGET (Access Next JNCB on the Chain)

The JNGET macro accesses the next job net control block (JNCB) on the JNCB chain. Only synchronous access is allowed to the JNCB chain.

Upon return, register 1 contains the address of the JNCB.

Syntax

```
JNGET label EOF= address, NORMAL= address
```

Parameters

- **EOF=**
  Specifies the address to which the routine returns when the JNCB chain is empty.
NORMAL=
This parameter specifies that after the macro processing routine executes, the
issuing program can resume execution at an instruction other than the
instruction that immediately follows the macro.

Linkage
This macro uses ASAVE linkage.

Restrictions
Do not use this macro in any routine that might run in a C/I FSS address space or
under the JES3 auxiliary task.

JSERV (Issue SSISERV Request)
The JSERV macro provides communication between JES3 and a user or functional
subsystem (FSS) address space, between JES3 on one processor and JES3 on
another processor, and between FCTs in the same JES3 address space.

Syntax

```
JSERV

label

,TYPE=COMM

MPC=| TEXT1

,TYPE=ACK,DEST=name

,TYPE=RESP,SA=|

,TYPE=PURGE,TEXT=

,NORMAL=address

MOD=:

,MOD=hex-

name

FSS:

,FSID=(reg),FSSNAME=(reg)

address

address

SA=:

,SA=(RO)

address
```
Parameters

MPC=
  Specifies the address of the main processor control table (IATYMPC) for the processor to which the data is being sent.

TEXT=
  (R1)
  address
  Specifies the address of the data to be sent. The first two bytes must contain the length of the data. The maximum length is 3800 bytes.

SA
  Specifies that the data to be sent is in the staging area.

TYPE=
  COMM
    Specifies that the request is for communications only; no response will be received. If you specify TYPE=COMM, you must specify the MPC and TEXT parameters but you may not specify TEXT=SA or the SA parameter.

  RESP
    Specifies that this request is in response to a previously received REPLY, WAIT, or ACK SSISERV request or ACK JSERV request. The length of the response data cannot exceed the data length of the original request. If you specify TYPE=RESP, you must specify the SA and TEXT parameters and you must not specify the MPC, DEST, and MOD parameters.

  PURGE
    Must be used to return a previously routed staging area if it was not responded to. If you specify TYPE=PURGE, you must specify the SA and TEXT parameters and you must not specify the MPC, DEST, and MOD parameters.

  ACK
    Specifies that the request is for communication and that the request requires acknowledgment. The RESP subparameter is used to make that acknowledgment. If you specify TYPE=ACK, you must specify the DEST, MPC and TEXT parameters but you may not specify TEXT=SA or the SA parameter.
DEST=
Specifies the name of a destination class. The register form may not be coded.

MOD=
Specifies a modifier which further defines the request.

FSSNAME=

    address
Specifies the name of a field containing the 8-character FSS name for the FSS address space to which this JSERV request is to be routed.

    (reg)
Specifies the register containing the address of the 8-character FSS name for the FSS address space to which this JSERV request is to be routed.

This parameter is valid only with the TYPE=COMM parameter. If you specify this parameter, you must also specify the FSID parameter and you may not specify the DEST and MOD parameters.

FSID=

    address
Specifies the name of a field containing the fullword functional subsystem (FSS) identifier for the FSS to which this JSERV request is to be routed.

    (reg)
Specifies the address of the fullword FSID of the FSS to which this JSERV request is to be routed.

This parameter is valid only with the TYPE=COMM parameter. If you specify this parameter, you must also specify the FSSNAME parameter and you may not specify the DEST and MOD parameters.

SA=
Specifies the address of the staging area.

NORMAL=
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

Linkage
This macro uses ASAVE linkage.

Restrictions
Do not use this macro in any routine that might run under the JES3 auxiliary task. This macro uses register 2.

LOGIN (Establish Console Communication Linkage)
The LOGIN macro establishes communication and transfer of data between console service and the dynamic support program (DSP) using the macro. This macro must be executed by each DSP that allows the receipt of messages and responses from the consoles.

Syntax

\[
\text{label} \quad \text{LOGIN} \quad \text{ENTER} = \quad (\text{RI}) \quad \text{address} \quad \text{FAIL} = \quad \text{NO} \quad \text{YES}
\]
LOGIN

Parameters

ENTER= Specifies the address to which console service passes control when an operator command is entered for this DSP. The DSP, at this time, accepts or rejects the command and returns control to console service. The DSP should not take any action on the command at this time.

FAIL= Specifies whether console service or the DSP’s console appendage should process *FAIL operator commands.

YES Specifies that the DSP’s console appendage should process *FAIL operator commands.

NO Specifies that console service should process *FAIL operator commands.

NORMAL= This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

Linkage

This macro uses ASAVE linkage.

Restrictions

Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task.

LOGOUT (End Console Communication Linkage)

The LOGOUT macro ends the console communication linkage established by the LOGIN macro.

Syntax

\[ label \rightarrow \text{LOGOUT} \rightarrow \text{SE=}(\text{RO}) \rightarrow (\text{address}) \rightarrow \text{,AWRITE=} \rightarrow \text{,YES=NO} \rightarrow \text{,NORMAL=address} \]

Parameters

SE= Specifies the address of the job description and accounting block (JDAB) scheduler element.

AWRITE=
YES
-Specifies that the routine is to write the JDAB before returning.

NO
-Specifies that the DSP may elect to have LOGOUT read and locate the scheduler element. On return, the DSP may post any additional accounting information in that entry. The DSP is then required to write the JDAB. Register 0 contains the address of the JDAB scheduler element and register 1 contains the JDAB FDB address to be supplied to the AWRITE macro.

NORMAL=
-This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

Linkage
-This macro uses ASAVE linkage.

Restrictions
-Do not use this macro in any routine that might run under the JES3 auxiliary task.

MDSALLOC (Allocate MDS Resources)
-The MDSALLOC macro requests main device scheduling (MDS) resource allocation.

Syntax

```
MDSALLOC TYPE=label

ALLOCATE Options2
  MPC= | MAINMSK= | ,RQ=(R1)

CATALOG Options1

,NORMAL=address

,OPTIONS=SETUP,ERROR=address

,MPC=(R0) address
```

Options2:

```
,OPTIONS=SETUP,ERROR=address

,BREAKDOWN
```

MPC=:
MDSALLOC

MAINMSK=:

\[ \text{XIсложнь} \]

Options 1:

\[ \text{XIсложнь} \]

Parameters

\text{TYPE=}

Specifies the action required to satisfy this request.

\text{ALLOCATE}

Interfaces directly with the MDS allocate function to allocate volumes, data sets, and devices. (If TYPE=ALLOCATE is specified, this macro will use register 2).

\text{CATALOG}

Initiates MDS processing to set up a private catalog request for the converter/interpreter function. If you specify this subparameter, register 13 must point to the CI DSP data area (IATYIDD).

\text{RQ=}

Specifies the location of the RESQUEUE entry related to this request.

\text{ERROR=}

Specifies the address to which control returns if an error occurs. This parameter must not be specified if OPTIONS=BREAKDWN is coded, but is required in all other cases.

\text{OPTIONS=}

Specifies unique requirements for a request.

\text{MAINTEST}

Tests if this request can ever be satisfied on the specified main processor. This subparameter is valid for TYPE=ALLOCATE only.

\text{RESERVE}

Reserves all obtainable resources. This subparameter is valid for TYPE=ALLOCATE only.

\text{NORESERV}

Reserves resources only if all resources are obtainable. This subparameter is valid for TYPE=ALLOCATE only.

\text{SETUP}

Initiates the volume fetching, mounting, and verification of a private catalog. This subparameter is valid for TYPE=CATALOG only.

\text{BREAKDWN}

Deallocation of a private catalog. This subparameter is valid for TYPE=CATALOG only.
MPC=
  Specifies the address of the main processor control table related to this request. Valid and required for TYPE=ALLOCATE only.

MAINMSK=
  Specifies the address of the fullword main task which represents the processor execution eligibility. This parameter is valid only for TYPE=ALLOCATE.

NORMAL=
  This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

Linkage
  This macro uses ASAVE linkage.

Restrictions
  This macro requires addressability to the MDS table (IATYMDS). Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task. (If TYPE=ALLOCATE is specified, this macro will use register 2).

MDSDSN (Build or Delete a SETDSN Entry)
  The MDSDSN macro builds or deletes a data set name entry (SETDSN).

Syntax

```
/MDSDSN VLM=(R0) DSN=(R1),TYPE=BUILD,NORMAL=(R1),BUSY=(R1),TYPE=DELETE
```

Parameters

VLM=
  Specifies the address of the resident volume allocation table (SETVOL) entry which is associated with the SETDSN entry.

DSN=
  For TYPE=BUILD, specifies the address of the fully qualified data set name. The first byte of the field must contain the length of the data set name.
  For TYPE=DELETE, specifies the address of the SETDSN entry.

TYPE=
  Specifies whether the routine is to build or delete the specified SETDSN entry. For TYPE=DELETE, register 13 must contain a pointer to the MDSDATA control block.

BUSY=
  Specifies the address to which the routine returns if the SETDSN already exists. This parameter applies only when TYPE=BUILD is coded.
MDSRSN

**NORMAL=**

This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

**Linkage**

This macro uses ASAVE linkage.

**Restrictions**

Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task.

**MDSERRQ (Invoke Installation Exit 61)**

The MDSERRQ macro is used by MDS modules to invoke a subroutine that will invoke installation exit IATUX61. Installation exit 61 will enable you to cancel jobs before they are placed on the MDS error queue, thereby reducing actions needed by the system operator.

On return from this macro, register 0 contains the RQINDEX value of the queue the RESQUEUE entry will be put on. Register 1 still has the address of the RESQUEUE entry. These are the values needed to issue a RQTAPUT macro to put the entry on the queue.

**Syntax**

```
MDSERRQ
RQ = R1
REASON = FETCH

 ViewGroup

Parameters

**RQ=**

Specifies a register (either register 1 or registers 2-9) that contains the address of the RESQUEUE entry for the job to be placed on the MDS error queue.

**REASON=**

Specifies the reason that MDS is placing the job on the MDS error queue. REASON can be specified as a keyword value or as a register (register 0 or 2-9) that contains the error code. The defined values are:

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Code</th>
<th>Label</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FETCH</td>
<td>0</td>
<td>RSNFETC</td>
<td>The job had an error during fetch</td>
</tr>
<tr>
<td>ALLOC</td>
<td>4</td>
<td>RSNALOC</td>
<td>The job had an allocation error</td>
</tr>
<tr>
<td>VERIFY</td>
<td>8</td>
<td>RSNVERF</td>
<td>The job had a verification error</td>
</tr>
<tr>
<td>RESTART</td>
<td>12</td>
<td>RSNREST</td>
<td>The job had an error during MDS restart</td>
</tr>
<tr>
<td>FAILSOFT</td>
<td>16</td>
<td>RSNFAIL</td>
<td>Error because of MDS failsoft processing</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>RSNPRMIO</td>
<td>The job had a permanent JSAM I/O error</td>
</tr>
<tr>
<td>Keyword</td>
<td>Code</td>
<td>Label</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>------</td>
<td>---------------</td>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td>24</td>
<td></td>
<td>RSNNOBUF</td>
<td>No JSAM buffers available for JESREAD</td>
</tr>
<tr>
<td>28</td>
<td></td>
<td>RSNNOJST</td>
<td>The job had a zero JST FDB</td>
</tr>
<tr>
<td>SYSSEL</td>
<td>32</td>
<td>RSNSYSSL</td>
<td>Error during MDS system select processing</td>
</tr>
<tr>
<td>SYSVER</td>
<td>36</td>
<td>RSNSYVER</td>
<td>Error during MDS system verify processing</td>
</tr>
<tr>
<td>SRSFAIL</td>
<td>40</td>
<td>RSNSRSFL</td>
<td>Error during MDSSRS FAILSOFT processing</td>
</tr>
</tbody>
</table>

(The labels are equates for the values as defined in macro IATYMDS.)

**Linkage**

This macro uses ASAVE linkage.

**Restrictions**

This macro requires addressability to the MDS table (IATYMDS). Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task.

**MDSID (Create MDS Trace Table Entry)**

The MDSID macro creates a main device scheduler (MDS) trace table entry.

**Syntax**

```
MDSID TYPE=JOBTRACE,SSIDYARC,BYPASS,RQ=(reg),MODULE=cc
```

**Parameters**

**TYPE=**

Specifies the type of trace entry being created.

**JOBTRACE**

Specifies that the movement of the resident job queue (RESQUEUE) entry is from one MDS function to another.

**SSIDYARC**

Specifies the return code passed from MDS to the dynamic allocation SSI.

**BYPASS**

Specifies the reason a RESQUEUE entry is bypassed by MDS allocation.

**RQ=**

Specifies the address of the RESQUEUE entry of the job being traced.

**MODULE=**

Specifies the last 2 characters of the name of the MDS module making the trace entry; the module name is IATMDcc. Valid characters are:

- AL – MDS allocation
- BK – MDS breakdown
- DR – MDS driver
MDSID

- FE – MDS fetch
- IQ – MDS inquiry
- MO – MDS modify
- OP – MDS operator commands
- RS – MDS restart
- SB – MDS subroutines
- SL – MDS select
- SR – MDS system resource scheduling
- VE – MDS verify

NEWINDX=
Specifies the RESQUEUE index of the function to which the job is being passed. This parameter applies only when TYPE=JOBTRACE is coded.

Linkage
This macro issues the IATZCALL macro without using ASAVE linkage.

Restrictions
Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task.

MDSJERR (Analyze Allocation Error or Locate Error Message for JST Entry)
The MDSJERR macro, with the RQ, MPC, and ERROR options, is used when allocation has failed. The failure is analyzed and appropriate messages are issued. If the job summary table (JST) shows an error for an entry, the MDSJERR macro, with the JST option, is used to access a predefined error message which corresponds to an error field in the JST entry.

Syntax

```
MDSJERR
  label

JST= (R1) address
RQ= (R7) address
  ,MPC= (R9) address

Returns:
  ERROR= address
  ,NORMAL= address
```

Parameters

JST=
Specifies the address of a JST entry that contains the error code to be interpreted into a message whose address is in register 0.

RQ=
Specifies the address of the resident job queue (RESQUEUE) entry.
MPC=
    Specifies the address of the main processor control table (MPC) of the processor on which the job is ineligible for setup.

ERROR=
    This parameter specifies the address to which the macro processing routine returns if an error occurs. When control is passed to the address, register 0 contains a dump code and register 15 contains a return code, unless otherwise specified.

    If an error occurs and ERROR was not specified, the dynamic support program (DSP) will be ended by using the FAILDSP macro.

NORMAL=
    This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

Linkage
    This macro uses ASAVE linkage.

Restrictions
    Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task.

MDSJGET (Get JST Entry)
    The main device scheduler (MDS) uses the MDSJGET macro to access a job’s job summary table (JST) entry. The MDSJST macro must be issued before MDSJGET (see the EOD parameter description below).

    On return, the address of the JST entry is in register 1.

Syntax

    
    \texttt{MDSJGET\quad label\quad RQ= (reg), JST= (R1), EOD= address}
    \texttt{\quad STEP= address, ERROR= address, NORMAL= address}

Parameters

    RQ=
        Specifies the address of the resident job queue (RESQUEUE) entry.

    JST=
        Specifies the address of the last JST entry. If the register contains 0, it indicates the first JST entry access.

    EOD=
        Specifies the address to which the routine returns when the end of data (EOD) has been reached. The EOD condition is defined by the JGETEOD parameter of the MDSJST macro.

    STEP=
        Specifies the address to which the MDSJGET routine returns when it finds a JST step entry.
MDSJGET

ERROR=
Specifies the address to which the routine returns when a permanent JSAM I/O error occurs. No further processing should be done on this job because the RESQUEUE has already been moved to the MDS error queue.

NORMAL=
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

Linkage
This macro issues a BALR R14,R15 for linkage.

Restrictions
Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task.

MDSJST (Read, Write, and Release JST Buffers)
Use the MDSJST macro to read, write, release, or checkpoint a job's job summary table (JST) in code that runs under the MDS (SETUP) FCT. This macro must be issued before each MDSJGET macro. On return, the address of the JST header is in register 1. JST header refers to the first JST buffer, unless JGETEOD=DYNALOC was specified. In this case, JST header refers to the first buffer for the last dynamic allocation of the requested step.

Syntax

Options1:

MULTIPLE
SINGLE
CHECKPNT
STEP
Options2:

\[ \text{OPTION=} \begin{array}{c}
\text{MULTIPLE} \\
\text{SINGLE} \\
\text{STEP}
\end{array} \]

Options3:

\[ \text{OPTION=} \text{MULTIPLE} \]

Options4:

\[ \text{OPTION=} \text{CHECKPNT} \]

Parameters

**TYPE=**
Specifies the type of I/O operation to be performed.

**RQ=**
Specifies the address of the resident job queue (RESQUEUE) entry associated with the JST records.

**JGETEOD=**
Specifies the definition of the end-of-data (EOD) condition for the MDSJGET macro. This parameter is valid only if TYPE=READ is specified. This parameter influences the number of buffers that are read.

**DYNALOC**
EOF is to occur at the end of the last dynamic allocation of the requested step (RQDYNSTP). All batch JST buffers will be read up to and including the one containing the step entry for the current step. Also, all dynamic allocation buffers for the last dynamic allocation of the current step are read (except if OPTION=SINGLE is specified).

**STEP**
EOF is to occur at the JST entry that ends the requested step. (The steps correspond to the number in MDRQSTPN.) The JST will be positioned (that is, MDSJGET will start accessing) to the step entry of the requested step. All batch JST buffers will be read up to and including the one containing the step entry for the step following the requested step. Also, all dynamic allocation buffers for that step will be read. JGETEOD=STEP is not valid with OPTION=SINGLE or OPTION=STEP.

**JOB**
EOF is to occur at the end of the JST.

**OPTION=**
This parameter further qualifies the TYPE= parameter.

**MULTIPLE**
Causes I/O processing to be performed for the entire JST unless JGETEOD=STEP or JGETEOD=DYNALOC is specified.
SINGLE
Causes I/O processing to be performed for the JST header record. For
OPTION=SINGLE and JGETEOD=DYNALLOC, I/O is performed for all batch
(that is, not dynamic allocation) JST buffers up to and including the buffer
containing the entry for the requested step, but only for the first header of
the last dynamic allocation of the requested step.
OPTION=SINGLE is valid only when TYPE=READ is specified.

CHECKPNT
Causes a checkpoint WRITE to be performed. This option is valid only with
TYPE=WRITE.

STEP
Causes I/O to be performed for all batch JST buffers until the entry for the
step following the requested step has been found. (No I/O is performed for
dynamic allocation JST buffers.) In addition, the JST is positioned (that is,
MDSJGET will start accessing entries) to the step entry corresponding to
the step requested in the RQDYNSTP field. This option is valid only with
TYPE=READ.

MODONLY=
Specifies whether only modified buffers should be written. This parameter is
valid only if TYPE=WRITE is specified. The default value is NO.

YES
Write only the buffers marked “modified.”

NO
Write the whole chain.

NORMAL=
Specifies the address to which the routine returns upon normal completion of
macro processing. If you omit the NORMAL= parameter, control returns to the
next sequential instruction after the macro.

ERROR=
Specifies the address to which the routine returns if an uncorrectable error
occurs. No further RESQUEUE processing should be done because the job will
have been removed from the MDS processing queues.

Linkage
This macro uses ASAVE linkage.

Restrictions
Do not use this macro in any routine that might run under FCTs other than SETUP
or DDR, or any routine that might run in a C/I FSS address, or under the JES3
auxiliary task.

MDSMSG (Construct MDS Output Messages)
The MDSMSG macro constructs main device scheduler (MDS) output messages.
Console service also uses it to route device-related messages to MDS. The calling
routine must issue the IATYMDS macro.

Syntax
Parameters

To construct messages, the TEXT parameter is used with either the DSN, ERRID, or APPEND parameter. The address of the combined text is returned in register 1.

To route a message, only the ROUTE parameter is used.

**TEXT=**

Specifies the address of the message. The first byte of the message must contain the length of the text. If coded, then the ROUTE parameter may not be coded. (If the TEXT= parameter is specified, this macro will use register 2.)

**ROUTE=**

Specifies the address of a 2-word parameter list, which is defined as follows:
The first word contains the address of the unit, and the second word contains the address of the volume serial number. If ROUTE is coded, then no parameters, except NORMAL, may be coded.

**DSN=**

Specifies the address of a data set name. The first byte of the data set name must contain the length of the data set name. The DSN parameter on the SETPARAM initialization statement will be used to determine how much of the data set name should be appended. If coded, then the ERRID and APPEND parameters may not be coded.

**ERRID=**

Specifies the address of an 8-byte (2-word) area. This area contains the length of the error message to be added in the first byte of the first word, and the address of the error message in the second word. The remainder of the first word must be zeros.

If ERRID is specified, the DSN and APPEND parameters may not be specified.

**APPEND=**

Specifies the address of the text to be appended to TEXT. The first byte of the text must contain the length of the text. If APPEND is coded, then the DSN and ERRID parameters may not be coded.

**NORMAL=**

This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

Linkage

If the TEXT parameter is specified, this macro issues the IATZCALL macro without using ASAVE linkage. If the ROUTE parameter is specified, this macro uses ASAVE linkage.
Restrictions
Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task. (If the TEXT= parameter is specified, this macro will use register 2.)

MDSVGET (Get JVT Entry)
The main device scheduler (MDS) uses the MDSVGET macro to access a job's job volume table (JVT) entry.

On return, register 1 will contain the address of the JVT entry.

Syntax

```
MDSVGET RQ=label (R0) (reg), JVT=(R1) (reg), JST=(reg)
```

Parameters

**RQ=**
Specifies the address of the resident job queue (RESQUEUE) entry associated with the JVT records.

**JVT=**
Specifies either the address of the last JVT entry or zero, to indicate access of the first JVT entry.

**JST=**
Specifies the address of the JST entry associated with the JVT records.

**EOD=**
Specifies the address to which the routine returns when the end of the JVT is reached.

**ERROR=**
Specifies the address to which the routine returns if an uncorrectable error occurred. No further RESQUEUE processing should be done because the job will have been removed from the MDS processing queues.

**NORMAL=**
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

Linkage
This macro issues the IATZCALL macro without using ASAVE linkage.

Restrictions
Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task.
MDSVLM (Build, Scan, or Delete SETVOL Entry)

The MDSVLM macro builds, scans, or deletes a resident volume allocation table (SETVOL) entry.

If TYPE=BUILD or TYPE=SCAN is specified, register 1 contains the address of the SETVOL entry upon return.

Syntax

```
MDSVLM label, TYPE=BUILD
MDSVLM label, TYPE=SCAN
MDSVLM label, EOD= (reg)
MDSVLM label, TYPE=DELETE
MDSVLM label, DSNTYPE=NONSMS
MDSVLM label, VLM= (R1)
MDSVLM label, NORMAL= address
```

Parameters

**TYPE=**

Will BUILD, SCAN or DELETE the specified SETVOL entry.

**DSNTYPE=**

Specifies the type of entry being processed. NONSMS indicates that the entry being processed is for a non-SMS managed data set. SMS indicates that the entry is for an SMS-managed data set. NONSMS is the default.

**VLM=**

If TYPE=BUILD or TYPE=SCAN and DSNTYPE=NONSMS is coded, this parameter specifies the address of the 6-byte volume serial number.

If TYPE=BUILD or TYPE=SCAN and DSNTYPE=SMS is coded, this parameter specifies the address of one byte specifying the length followed by a 44-byte data set name.

If TYPE=DELETE is coded, this parameter specifies the address of the SETVOL entry.

**EOD=**

Specifies the address to which the routine returns if the SETVOL entry does not exist. This parameter applies only when TYPE=SCAN is coded.

**BUSY=**

Specifies the address to which the routine returns if the SETVOL entry already exists. Register 1 contains the address of the SETVOL entry. This parameter applies only when TYPE=BUILD is coded.

**NORMAL=**

This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.
Linkage

This macro uses ASAVE linkage.

Restrictions

Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task.

MESSAGE (Standard Form: Issue a Message from a JES3 DSP)

The standard form of the MESSAGE macro issues an operator message. This macro also allows you to specify the type of message being sent: immediate action, informational, failure, or eventual action.

Action messages must be dequeued using the DEQMSG macro when the requested action has been completed. The MESSAGE macro will return a DOM ID in register 0, so that you can use it as the value for the BUFFER= parameter on the DEQMSG macro.

If you use this macro specifying ACTION=YES or ACTION=CRITICAL, the default WTO generated uses descriptor code 7. These messages are retained until JES3 termination.

If any action messages need to be retained across address space termination, you should specify the ACTION=FAIL parameter on the MESSAGE macro.

If you specify the MESSAGE macro incorrectly, a DM132 ABEND may be issued by the MESSAGE service routine, or if JES3 does not detect the error, a D23 ABEND could occur when the message is converted to a WTO.

If you use installation exit 18 and you choose to code the MESSAGE macro in the exit routine, you must specify the BUSY= parameter so that the CONCMD DSP (under which IATUX18 runs) does not enter a wait state.

JES3 converts the parameters on the MESSAGE macro to the corresponding parameters on the WTO macro, and then issues the message as a WTO. Table 20 on page 555 shows the relationship between the parameters on the two macros.

Note: If your program is running outside of the JES3 address space (or under a writer FSS), use the WTO macro instead of the MESSAGE macro.

Syntax
Parameters

TEXT=
Specifies the address of the message to be issued. The maximum length of the message area is 120 bytes. The length is given in the first byte and the remaining 119 bytes contain the message text. TEXT= can be specified as an address or as a register. When the list form is used, TEXT= must use the address form.

Note: The TEXT= and MLWOLST= keywords are mutually exclusive.

MLWOLST=
Specifies the token value returned from the first IATXMLWO invocation for a multi-line message.

This parameter specifies a register which contains the token value, or a label of a 4 byte field which contains the token value.

Note: The MLWOLST= and TEXT= keywords are mutually exclusive. Prior to issuing the MESSAGE macro, set up the multi-line message through
calls to the IATXMLWO REQUEST=BUILD service. After issuing the message, use IATXMLWO REQUEST=CLEANUP to release resources held for the message.

CLASS=
Specifies the JES3 destination class to which the message is to be directed. Both CLASS= and CONS= parameters may be coded. If so, the message goes to the places defined to the class and to the indicated console.

ALL
All consoles. (JES3 messages with CLASS=ALL are treated as broadcast messages.)

D1-D22
Extra destination classes for customer use.

ERR
System error console.

LOG
System log console. (JES3 messages with CLASS=LOG are also displayed on any MCS consoles that are monitoring job names.)

MLG
Master log console. (JES3 messages with CLASS=MLG are treated as WTOs issued with the “hardcopy only” function.)

M1-M32
Classes related to the mains in the complex.

JES
Global processor.

S1-S32
Classes related to the devices in the complex.

TAP
Tape consoles.

TP
Teleprocessing consoles.

UR
Unit record consoles.

SEC
Specifies security messages.

(reg)
If register notation is used, the lower 2 bytes of the register must contain a halfword (1-byte displacement and 1-byte mask) converted from the class name by using the CONCNVRT macro.

CONS=
Specifies the address of the 4 byte console ID of the console that is to receive the message. If the register form is used, the register must contain the console ID. Both CLASS= and CONS= parameters may be coded. If so, the message goes to the places defined to the class and to the indicated console.

CNDB=
Specifies the Console Destination Block(s) associated with the message. The CNDB contains destination information that determines where the message is displayed. A single CNDB address or multiple CNDB addresses may be
specified with this keyword. A maximum of 5 CNDB addresses is allowed. This parameter is not allowed on the list form of the MESSAGE macro.

If multiple CNDBs are specified, the information in the individual CNDBs is extracted or merged to form the routing information used on the message. The following rules determine how the CNDB information is merged:

- **Console ID** - The console name/console ID pair in the first CNDB with a non-zero and non-blank console name is used for the MESSAGE request. The CNDBs are scanned starting with the leftmost parameter specified on the CNDB= keyword.

  **Note:** An exception to this rule is the ‘dummy’ CNDB built by JES3 and used internally on certain MESSAGE invocations. The console name in this CNDB is non-zero and non-blank, but it is subject to be overridden by subsequent CNDBs in the list.

- **Command Response Indicator** - The message is treated as a command response if any CNDB in the list has the command response indicator set. The CART (Command and Response Token) will be taken from the first CNDB encountered with the command response indicator set (starting from the leftmost CNDB parameter).

- **Routing Code Information** - The routing code mask in all CNDBs is merged to form a single routing code mask.

The IATXCNDB service must be used to initialize a CNDB and to update the information in an existing CNDB.

**Note:** Message destination and routing information in the CNDB(s) is subject to be overridden by an explicit Console ID (through the CONS= keyword), or explicit routing code information (through the CLASS=, ROUT=, or ROUTCDE= keywords).

**ACTION=**

Specifies the type of message and whether it should be entered into an action queue. Action messages must be dequeued by the DEQMSG macro when the requested action has been done.

**YES**

System or subsystem immediate action message. This specification corresponds to MCS descriptor codes 2 and 7.

**NO**

System or subsystem informational (non-action) message. This specification corresponds to MCS descriptor code 7.

**FAIL**

System or subsystem failure message. This specification corresponds to MCS descriptor code 1.

**CRITICAL**

Eventual critical action message. This specification corresponds to MCS descriptor codes 7 and 11.

**PRTY=**

Specifies the priority of the message. The default is 5. Only the decimal form of PRTY= is allowed in the list form of this macro. If PRTY= is specified as a decimal number, the range for the number is 00 to 11. If PRTY= is specified as an address, the address must point to a 1-byte field
containing the priority. If PRTY= is specified as a register, the low-order byte must contain the priority. The following priorities have special meanings:

9-11 The message is exempt from JES3 constraint checking which may cause the issuance of the message to be delayed.

MF=
Specifies if the MESSAGE request is in the list or execute form. The address specifies the location of the list form expansion of the macro.

MLG=
Specifies whether the message is to be written to the system log.

DESC=
Specifies additional MCS descriptor codes associated with this message. This 2-byte field is ORed with another 2-byte field containing the descriptor code equivalents implied by the ACTION= parameter. The result yields the final set of descriptor codes for the message. Certain combinations of descriptor codes are not allowed. For definitions of the descriptor codes in an MVS/ESA™ system, see z/OS MVS Programming: Assembler Services Guide.

The DESC= value can be either the address of a 2-byte field containing the MCS descriptor codes or a register pointing to a 2-byte field. If the register form is used, register 1 cannot be specified.

In this 2-byte field, each of the 16 bits corresponds to a single descriptor code. For example, the first bit of byte 1 indicates descriptor code 1, and the first bit of byte 2 indicates descriptor code 9.

ROUT=
Specifies the MVS routing codes associated with this message.

For definitions of the MVS routing codes in an z/OS system, see z/OS MVS Programming: Assembler Services Guide.

The ROUT= value can be either the address of a 17-byte field, or a register pointing to a 17-byte field.

The 17-byte field is a bit mask where each of the 128 bits in the first 16 bytes represent a single MVS routing code. The last byte of the 17-byte field represents the JES3 ALL and MLG destination classes because these classes do not have a routing code equivalent. The first bit of the 17th byte represents destination class ALL, while the 2nd bit represents destination class MLG.

If the register form is used, register 1 cannot be specified.

Note: The ROUT= and ROUTCDE= keywords are mutually exclusive.

ROUTCDE=
Specifies MVS routing code(s) in decimal number format. With this keyword you can specify one or more codes from 1 to 128 separated by commas, or a hyphen to indicate a range.

Certain routing codes are reserved for future use by IBM and will be ignored by the system if specified. JES3 does not prevent their use, nor does it provide any warning if specified.

Note: The ROUT= and ROUTCDE= keywords are mutually exclusive.
MESSAGE

JOBNAME=
Specifies a job name. This field is 8 bytes long, and must be left-justified and padded with trailing blanks. If the register form is used, it must contain a pointer to an 8-byte job name field.

If you specify JOBNAME=, you must also specify JOBID=.

Note: If you do not specify the JOBNAME= parameter or if you specify JOBNAME=JES3, the message is not written to the JESMSG data set. However, if you specify any value other than JES3 on the JOBNAME= parameter, the message is written to the JESMSG data set.

JOBID=
Specifies a JES3 job id. This field is 8 bytes long. If the register form is used, it must contain a pointer to an 8-byte job number field.

If you specify JOBID=, you must also specify JOBNAME=.

KEY=
Specifies the address of a key (such as the DSP name) used with D,R,L,KEY= keyname command to retrieve this action message.

SAVE=
Specifies whether the macro is to use ASAVE linkage or not.

BUSY=
Specifies the address to which the routine returns if the following condition exists:
  • There is a shortage of staging areas

If the routine returns to this address, an ECF mask and address are returned which may be used to wait until the shortage is relieved. Register 0 will contain the 31-bit ECF address. The ECF mask needed is contained in the low-order byte of register 1, with the other three bytes containing zeros.

If you omit this parameter and a busy condition occurs, the routine issuing this macro will wait until the condition is resolved.

NORMAL=
Specifies the address to which the routine returns upon normal completion of macro processing. JES3 stores this address in register 14. If you omit the NORMAL= parameter, the issuing module receives control again at the next sequential instruction following the MESSAGE macro.

On return from this macro, the registers contain:
For a normal return:
  Register 0 contains the DOM ID.
  Register 11 contains the address of the FCT.
For a busy return, the BUSY= exit is taken and the registers contain:
  Register 0 contains the ECF address.
  Register 1 contains the ECF mask.
  Register 11 contains the address of the FCT.

For an error return, a FAILDSP with ABEND code DM132 will be issued if the input message length is 0, or the destination class specified is incorrect.
Register 2 contains a return code indicating the cause of the failure:
**MESSAGE**

<table>
<thead>
<tr>
<th>Return Code In Hex</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>04</td>
<td>The length of the message was 0</td>
</tr>
<tr>
<td>08</td>
<td>The destination class was incorrect</td>
</tr>
<tr>
<td>0C</td>
<td>Error return from IATXCNDB TRANSFER service.</td>
</tr>
<tr>
<td>10</td>
<td>Error return from IATXCNDB VERIFY service.</td>
</tr>
<tr>
<td>14</td>
<td>Multi-line message list token specified with the MLWOLST keyword is not valid.</td>
</tr>
<tr>
<td>18</td>
<td>The length of a line of a multi-line message is zero.</td>
</tr>
</tbody>
</table>

**Supplying Routing Information on the MESSAGE Macro**

Routing information for a message may be specified with the following MESSAGE parameters:
- **CONS** - Specifies the console ID of the console to receive the message
- **CLASS** - Specifies a single JES3 Destination Class to which the message is sent
- **ROUT** - Specifies a bit mask which represents the MVS routing code(s) to be applied to the message.
- **ROUTCDE** - Specifies MVS routing code(s) in decimal number format. \( \text{ROUT=} \) and \( \text{ROUTCDE=} \) are mutually exclusive.
- **CNDB** - Specifies the address of a Console Destination Block. Use of the CNDB is the preferred method of supplying routing information to the MESSAGE service.

The following rules apply to the specification of these parameters:
- Any combination of CONS, CLASS, and ROUT/ROUTCDE may be specified with CNDB=. In this case, the routing information in the CNDB is overridden by the CONS, CLASS, and ROUT/ROUTCDE values. If both CLASS and ROUT/ROUTCDE are specified, then the CLASS value is converted into its routing code equivalent (if one exists) and merged with the ROUT/ROUTCDE information to form a single routing code mask.
- If CLASS and ROUT/ROUTCDE are specified without CNDB, then the CLASS value is converted into its routing code equivalent and merged with the ROUT/ROUTCDE information to form a single routing code mask.
- If CONS is specified with routing codes (CLASS and/or ROUT/ROUTCDE and/or CNDB), then the message is sent to the specified console in addition to consoles receiving the routing codes.
- If CNDB is specified without CLASS, CONS, and ROUT/ROUTCDE, then it is the callers responsibility to initialize the CNDB with the proper destination information through use of the IATXCNDB macro. See [z/OS JES3 Customization](z/OS JES3 Customization) for a complete description on the use of the IATXCNDB macro.

**Linkage**

This macro uses ASAVE linkage.
### Restrictions

Do not use this macro in any routine that might run under the JES3 auxiliary task.

#### Table 20. Relationships Between MESSAGE Parameters and WTO Parameters

<table>
<thead>
<tr>
<th>MESSAGE Parameter</th>
<th>WTO Parameter</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TEXT=</td>
<td>text</td>
<td>JES3 uses the positional form in inline message text instead of the TEXT= parameter on the WTO macro.</td>
</tr>
<tr>
<td>CLASS=</td>
<td>ROUTCDE=</td>
<td>Note: If CLASS=MLG is specified in combination with other routing information (for example: CONS=, ROUT=, ROUTCDE=, CNDB=), MLG is overridden by the other routing information. Reserved for customer use. Particular processor messages Particular devices messages.</td>
</tr>
<tr>
<td>TAP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UR</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>TP</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>SEC</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>ERR</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>MLG</td>
<td>MCSFLAG=HRDCPY</td>
<td></td>
</tr>
<tr>
<td>LOG</td>
<td>41</td>
<td></td>
</tr>
<tr>
<td>JES</td>
<td>42</td>
<td></td>
</tr>
<tr>
<td>D1-D22</td>
<td>43-64</td>
<td></td>
</tr>
<tr>
<td>M1-M32</td>
<td>65-96</td>
<td></td>
</tr>
<tr>
<td>S1-S32</td>
<td>97-128</td>
<td></td>
</tr>
<tr>
<td>ALL</td>
<td>MCSFLAG=BRDCST</td>
<td></td>
</tr>
<tr>
<td>CNDB=</td>
<td>n/a</td>
<td>For use by the MESSAGE macro service routine. Information from the CNDB will be used for various WTO parameters (for example: Routing code information, console ID, command response indicator, command and response token (CART)).</td>
</tr>
<tr>
<td>ROUT=</td>
<td>ROUTCDE=</td>
<td>Contains the address of data used as routing codes by WTO.</td>
</tr>
<tr>
<td>ROUTCDE=</td>
<td>ROUTCDE=</td>
<td>Specifies the MVS routing code(s) to be assigned to the message.</td>
</tr>
<tr>
<td>DESC=</td>
<td>DESC=</td>
<td>Contains the address of data used as descriptor codes by WTO.</td>
</tr>
<tr>
<td>CONS=</td>
<td>CONSID=</td>
<td>The parameter value is used as is.</td>
</tr>
<tr>
<td>ACTION=</td>
<td>DESC=(2,7)</td>
<td>The ACTION= values are combined with the DESC= values to form a complete set of descriptor codes when JES3 converts the MESSAGE macro to a WTO.</td>
</tr>
<tr>
<td>YES</td>
<td>DESC=1</td>
<td></td>
</tr>
<tr>
<td>FAIL</td>
<td>DESC=(7,11)</td>
<td></td>
</tr>
<tr>
<td>CRITICAL</td>
<td>DESC=7</td>
<td></td>
</tr>
<tr>
<td>NO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PRTY=</td>
<td>PRTY=</td>
<td>The value is used internally by the MESSAGE macro service routine.</td>
</tr>
<tr>
<td>MLG=NO</td>
<td>MCSFLAG=NOCPY</td>
<td>Equivalent function</td>
</tr>
<tr>
<td>BUSY=</td>
<td>MCSFLAG=BUSYEXIT</td>
<td>Equivalent function</td>
</tr>
<tr>
<td>KEY=</td>
<td>KEY=</td>
<td>Specifies the address of the key used with the D,R,L,KEY= keyname command to retrieve this action message.</td>
</tr>
<tr>
<td>JOBNAME=</td>
<td>JOBNAME=</td>
<td>Name of the job. An 8-character field, left-justified.</td>
</tr>
</tbody>
</table>
Table 20. Relationships Between MESSAGE Parameters and WTO Parameters (continued)

<table>
<thead>
<tr>
<th>MESSAGE</th>
<th>WTO</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLWOLST=</td>
<td>N/A</td>
<td>Specifies the token value returned from the first IATXMLWO invocation for a multi-line message.</td>
</tr>
<tr>
<td>N/A</td>
<td>TOKEN=</td>
<td>A unique token for each FCT</td>
</tr>
<tr>
<td>NORMAL=</td>
<td>N/A</td>
<td>Processed by the MESSAGE service routine</td>
</tr>
<tr>
<td>SAVE=</td>
<td>N/A</td>
<td>Processed within the MESSAGE macro</td>
</tr>
<tr>
<td>JOBID=</td>
<td>JOBID=</td>
<td>The job number (in the form JOB0nnnn) associated with the message. This is an 8-byte field.</td>
</tr>
<tr>
<td>N/A</td>
<td>SUBSMOD=NO</td>
<td>The subsystem issuing the WTO cannot modify the message, which indicates that JES3 message route processing is not to be done.</td>
</tr>
<tr>
<td>N/A</td>
<td>CONNECT=</td>
<td>Used for issuing minor lines for a multiline WTO.</td>
</tr>
<tr>
<td>N/A</td>
<td>LINKAGE=SVC</td>
<td>JES3 uses the default WTO linkage value.</td>
</tr>
</tbody>
</table>

MESSAGE (Execute Form: Issue a Message)

The execute form of the MESSAGE macro refers to, and can modify, the list form of the MESSAGE macro. The description of the standard form provides the explanation of the function of each operand.

Syntax

```
MESSAGE MF=E, address TEXT=address, CLASS=ALL, D1-D22, ERR, LOG, MLG, M1-M32, JES, S1-S32, TAP, TP, UR, SEC, ACTION=NO, YES, FAIL, CRITICAL, PRTY=number, MLG=NO
```
Parameters

**MF=**
Specifies the execute form. The second subparameter specifies the address of
the list form expansion of the macro.

Linkage

This macro uses ASAVE linkage.

Restrictions

Do not use this macro in any routine that might run under the JES3 auxiliary task.

**MESSAGE (List Form: Issue a Message)**

The list form of the MESSAGE macro constructs a MESSAGE parameter list. The
description of the standard form provides the explanation of the function of each
operand.

Note that the following parameters are not valid with the list form of the MESSAGE
macro: DESC, ROUT, ROUTCDE, CNDB, MLWOLST, JOBID, JOBNAME, KEY.

Syntax
MESSAGE

Parameters

MF=
Specifies that a parameter list is to be created that will be referenced by an execute-form MESSAGE macro.

Linkage
This macro generates only a parameter list.

Restrictions
Do not use this macro in any routine that might run under the JES3 auxiliary task.

MNTRDEF (Create Monitor Definition Entry)

The MNTRDEF macro allows you to create a monitor definition entry that defines a resource or a queue to be monitored.

Syntax

```
\( id \) \( MNTRDEF \) \( DESC=\) \( ,INTERVAL=n\) \( nn \) \( ,THRESHOLD=n\) \( nn \) \( ,ROUTINE=label\) \( \) \( ,COUNT=ALL\) \( nnn \) \( ,SUMMARY=YES\) \( NO\) \( ONLY\) \( ,LAST=YES\) \( NO\) \n```

Parameters

id  Specifies a 1–to-8-character alphanumeric name given to the monitor definition. This name is used on modify commands to indicate which monitor definition is being changed. The first character must be an alphabetic character.

The following are the JES3-supplied id values:

**RESOURCE**
 Specifies monitoring of an unavailable JES3 resource. A JES3 resource is anything that can be used by an FCT and that can become unavailable.
The monitor DSP allows you to monitor the following JES3 resources:
• Generalized subtasks
• AENQ resources

LOCATE
   Specifies monitoring of jobs in catalog locate.

CIFSS
   Specifies monitoring of jobs scheduled to a C/I FSS.

SYSSELQ
   Specifies monitoring of jobs waiting in the MDS system select queue for DFSMS™-managed resources to become available.

ALLOCQ
   Specifies monitoring of jobs waiting in the MDS allocate queue for a resource to become available.

VERIFYQ
   Specifies monitoring of a job waiting in the MDS verify queue for one or more volume mounts to complete.

SYSVERQ
   Specifies monitoring of jobs waiting in the system verify queue.

DSPWAIT
   Specifies monitoring of jobs waiting for a particular DSP to become available.

JSSWAIT
   Specifies monitoring of a job on a job segment scheduler (JSS) wait queue waiting for an event to occur of a resource to become available. For example:
   • Jobs waiting for a PROCLIB to become available.
   • Jobs waiting for a main processor, class, or group to become available.
   • Jobs waiting for DFSMS-managed user catalogs to become available.
   • Jobs waiting for a main processor to become available for catalog locate.

IOWAIT
   Specifies monitoring of the ending function waiting for I/O to complete.

DESC=
   Specifies a 40 byte description of the queue or resource being monitored.

INTERVAL=nnn
   Specifies the amount of time in minutes, that elapses before the monitor DSP checks the queue or resource to determine if processing has exceeded the maximum time limit. The interval value must be in the range of 0 to 999.

   If you specify INTERVAL=0, no monitoring of the queue or resource is performed.

THRESHOLD=nnn
   Specifies the minimum amount of time, in minutes, that a job or functional control table (FCT) must have been waiting on the specified queue or for the specified resource before the monitor DSP will display it. The threshold value must be in the range of 0 to 999.

   If you specify THRESHOLD=0, the monitor DSP displays the job or FCT regardless of the amount of time.
ROUTINE=label
Specifies the routine to get control to collect the data when the monitoring
interval expires.

The following are the JES3-supplied label values:

MNJS3RSC
The routine to receive control when you specify an id of RESOURCE.

MNLOCATE
The routine to receive control when you specify an id of LOCATE.

MNCIFSS
The routine to receive control when you specify an id of CIFSS.

MNSYSSLQ
The routine to receive control when you specify an id of SYSSELQ.

MNALLOCQ
The routine to receive control when you specify an id of ALLOCQ.

MNVERFYQ
The routine to receive control when you specify an id of VERIFYQ.

MNSYSVRQ
The routine to receive control when you specify an id of SYSVERQ.

MNDSPWT
The routine to receive control when you specify an id of DSPWAIT.

MNJSSWT
The routine to receive control when you specify an id of JSSWAIT.

MINIOWAIT
The routine to receive control when you specify an id of IOWAIT.

Note: This parameter should only be modified if the installation decides to add
its own monitor functions.

COUNT={ALL|nnn}
Specifies the number of FCTs or jobs that are over the threshold value to be
displayed. The count value must be in the range of 0 to 999, or ALL.

If you specify COUNT=nnn, nnn is the number of FCTs or jobs that are over the
threshold value and are to be displayed. If you specify COUNT=ALL, all jobs or
FCTs that are over the threshold value are displayed. If you specify COUNT=0,
no jobs or FCTs are displayed.

The default count value is ALL.

SUMMARY={YES|NO|ONLY}
Specifies whether a summary message is to be issued.

YES
Specifies that the monitor DSP issues a summary message in addition to
the messages that are issued for each job or FCT.

NO
Specifies that the monitor DSP does not issue a summary message, but
messages are issued for each job or FCT.

ONLY
Specifies that the monitor DSP issues a summary message, but no
messages are issued for each job or FCT.
The default is SUMMARY=YES.

Notes:
1. If COUNT=0 and SUMMARY=NO, no messages are issued.
2. If you specify SUMMARY=ONLY, the COUNT= parameter is ignored.

LAST={YES|NO}
Specifies whether this is the last monitor definition entry. NO is the default.

Linkage

This macro does not generate executable code.

Restrictions
If you modify the MNTRDEF macro expansions in IATYMNTR, you must assemble and link-edit module IATGRMN as non-reentrant. To use the assembled and link-edited version of the monitor, IATGRMN, you must refresh LLA and HOT START JES3.

Examples of Use
The following examples list specifications of the MNTRDEF macro for the situations described.

Example 1
The installation has modified the MNTRDEF macro to have JES3 check the queue of jobs in catalog locate every ten minutes. A summary message is to be displayed describing the number of jobs that have been waiting more than three minutes. Detailed information is to be displayed only for the eight longest waiting jobs.

LOCATE MNTRDEF INTERVAL=10,
   THRESHOLD=3,
   COUNT=8,
   SUMMARY=YES,
   DESC='WAITING FOR CATALOG LOCATE',
   ROUTINE=MNLOCATE

Example 2
The installation has modified the MNTRDEF macro to have JES3 check the queue of jobs waiting for DSPs every 15 minutes. If a job has been waiting for a DSP for more than 30 minutes, the installation considers that to be too long. Only a summary message is to be issued, displaying the number of jobs waiting for a particular DSP. Detailed information is not requested because it is likely that many jobs are backlogged for a particular DSP such as C/I.

DSPWAIT MNTRDEF INTERVAL=15,
   THRESHOLD=30,
   SUMMARY=ONLY,
   DESC='WAITING FOR A DSP',
   ROUTINE=MNDSPWT

MOVEDATA (Move Data Exceeding 256 Bytes)
The MOVEDATA macro moves an area longer than 256 bytes. However, the MVCL assembler instruction performs the same function more efficiently and should be used instead.
MOVEDATA

Syntax

```
MOVEDATA

   label
   TO=  R2
       address
   FROM= R3
       address
   COUNT= R4
       number
   NORMAL= address
```

Parameters

TO=
   Specifies the address to which the area is to be moved.

FROM=
   Specifies the address of the area to be moved.

COUNT=
   Specifies the number of bytes to be moved.

NORMAL=
   This parameter specifies that after the macro processing routine executes, the
   issuing program can resume execution at an instruction other than the
   instruction that immediately follows the macro.

Linkage

This macro issues a BALR R14,R15 for linkage.

Restrictions

Do not use this macro in any routine that might run under the JES3 auxiliary task.

MTBL (Build a Main Service Search Table Entry)

Main service uses the MTBL macro to build a variable-length table entry to search
for a supplied 1- to 8-character value.

Syntax

```
MTBL

   label
   chars
   ,address
```

Parameters

chars
   Specifies a 1- to 8-byte character string. If the character string contains special
   characters or blanks, it must be enclosed in apostrophes.

address
   The name of a routine or address within the CSECT containing the MTBL
   macro call that will be assembled as a halfword offset from the CSECT start. If
   the second operand is not supplied, ‘chars’ will be used as the routine address.
Linkage

This macro generates inline code.

Restrictions

Do not use this macro in any routine that might run under the JES3 auxiliary task.

**NCBTAADD (Add a Net Control Block to the JNCB)**

The NCBTAADD macro adds a net control block (NCB) to a job network control block (JNCB). Only synchronous access is allowed.

Upon return, register 1 contains the address of the NCB.

**Syntax**

```
NCBTAADD
NCSECT= label (R1) address, NORMAL= address
```

**Parameters**

- **NCSECT=**
  Specifies the address of the job net control CSECT which contains the calling parameter list.

- **NORMAL=**
  This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

Linkage

This macro uses ASAVE linkage.

Restrictions

Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task.

**NCBTAFND (Locate a Net Control Block Within the JNCB)**

The NCBTAFND macro locates a specific net control block (NCB) within a job net control block (JNCB). Only synchronous access is allowed.

Upon return, register 1 contains the address of the requested NCB or zero, if the NCB is not found.

**Syntax**

```
NCBTAFND
ENTRY= (R0) address, NCB= (R1) address, JOBNUM= (reg) address
```
NCBTAGET (Locate Next Net Control Block Within the JNCB)

The NCBTAGET macro accesses the next net control block (NCB) within a job net control block (JNCB). Only synchronous access is allowed.

Upon return, register 1 contains the address of the NCB.
Parameters

ENTRY=
  Specifies the address of the job net control block (JNCB) which contains the NCB.

EOF=
  Specifies the address returned to when the end of the JNCB is reached.

NORMAL=
  This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

ERROR=
  This parameter specifies the address to which the macro processing routine returns if an error occurs. When control is passed to the address, register 0 contains a dump code and register 15 contains a return code, unless otherwise specified.

  If an error occurs and ERROR was not specified, the dynamic support program (DSP) will be ended by using the FAILDSP macro.

Linkage

This macro uses ASAVE linkage.

Restrictions

Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task.

NCBTAPUT (Write NCB Buffer)

The NCBTAPUT macro writes an updated net control block (NCB) buffer in the spool data set.

Syntax

\[
\text{label} \rightarrow \text{NCBTAPUT} \rightarrow \text{ENTRY} = \text{(R0)} \rightarrow \text{address} \rightarrow \text{NORMAL} = \text{address}
\]

Parameters

ENTRY=
  Specifies the address of the job net control block (JNCB) which contains the NCB.

NORMAL=
  This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.
NCBTAPUT

Linkage

This macro uses ASAVE linkage.

Restrictions

Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task.

NCBTAREL (Release NCB Buffer)

The NCBTAREL macro releases a previously read net control block (NCB) buffer.

Syntax

```plaintext
NCBTAREL label ENTRY=(R1) address NORMAL=address
```

Parameters

**ENTRY**=

Specifies the address of the job net control block (JNCB) which contains the NCB.

**NORMAL**=

This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

Linkage

This macro uses ASAVE linkage.

Restrictions

Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task.

NCKADD (Checkpoint a DJC JNCB)

The NCKADD macro adds a net control block (NCB) FDB and network ID to the dependent job control (DJC) checkpoint record.

Syntax

```plaintext
NCKADD label ENTRY=(R1) address NORMAL=address
```

Parameters

**ENTRY**=

Specifies the address of the job net control block (JNCB) for the DJC network to be checkpointed.

**NORMAL**=

This parameter specifies that after the macro processing routine executes, the
issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

Linkage

This macro uses ASAVE linkage.

Restrictions

Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task.

NCKDEL (Delete DJCJNCB Checkpoint Record)

The NCKDEL macro deletes a net control block (NCB) FDB and network ID from the dependent job control (DJC) checkpoint record.

Syntax

```
NCKDEL ENTRY=(R1) address, NORMAL=address
```

Parameters

**ENTRY**=
Specifies the address of the job net control block (JNCB) for the DJC network to be removed from the DJC checkpoint record.

**NORMAL**=
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

Restrictions

Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task.

PURCHAIN (Purge Chained Single Record Files)

The PURCHAIN macro purges chained single record files (SRFs).

Syntax

```
PURCHAIN FDB=(R1) address, AREA=(R0) address, DISP=number, ID=(reg) name, NORMAL=address
```
PURCHAIN

Parameters

FDB=
   Specifies the address of the FDB for the SRF being purged.

AREA=
   Specifies the address of a 6-fullword work area.

DISP=
   Specifies the displacement into the SRF of the chain FDB.

ID=
   Specifies the name to be associated with the single record file being purged. The ID length is limited to four characters. The ID= parameter is required.

NORMAL=
   This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

ERROR=
   This parameter specifies the address to which the macro processing routine returns if an error occurs. When control is passed to the address, register 0 contains a dump code and register 15 contains a return code, unless otherwise specified. The return codes and dump codes are described below. For a description of every dump code, see z/OS JES3 Diagnosis.

   If an error occurs and ERROR was not specified, the dynamic support program (DSP) will be ended by using the FAILDSP macro.

Return Codes

Following are the return codes and dump codes returned in registers 15 and 0, respectively:

<table>
<thead>
<tr>
<th>Return Code in Hex</th>
<th>Dump Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>04</td>
<td>713</td>
<td>Bad buffer address or spool address</td>
</tr>
<tr>
<td>08</td>
<td>nnn</td>
<td>Failure on internal call (APUTBUF, ARELEASE, JESREAD)</td>
</tr>
<tr>
<td>0C</td>
<td>710</td>
<td>Duplicate spool address</td>
</tr>
<tr>
<td>10</td>
<td>714</td>
<td>Spool address not in the single track table (STT)</td>
</tr>
</tbody>
</table>

Complete descriptions of the dump codes are found in z/OS JES3 Diagnosis.

Linkage

This macro uses ASAVE linkage.

Restrictions

Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task.

This macro uses register 2.
PUTUNIT (Return One or More JES3 Devices for Reallocation)

The PUTUNIT macro returns one or more JES3 devices to availability for allocation to other functions. The devices being returned were received by using the GETUNIT macro.

Syntax

```
PUTUNIT label LIST=(R1) address, NORMAL=address
```

Parameters

LIST= Specifies the address of the list which was used by the GETUNIT macro for allocation. If omitted, the sublist whose address is contained in the FCT is used.

NORMAL= This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

Linkage

This macro uses ASAVE linkage.

Restrictions

Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task.

RQTAADD (Add a RESQUEUE to the RESQUEUE Chain and Subchain)

The RQTAADD macro adds a resident job queue table (RESQUEUE) to the RESQUEUE chain and subchain.

Syntax

```
RQTAADD label ENTRY=(R1) address, INDEX=(reg) address, PARM=(reg) address, NORMAL=address
```

Parameters

ENTRY= Specifies the address of the entry to be added.

PARM= Specifies the register containing the address of a user-defined parameter list. The parameter list, which is mapped by the macro RQTAGEN specifying TYPE=DSECT, must contain (in this order):
RQTAADD

- The address of the entry to be added to the RESQUEUE
- The index, which must be a fullword containing one of the terms defined for field RQINDEX in the IATYRSQ macro
- The base address of the previous RESQUEUE subchain, which should be zero
- The base address of the new RESQUEUE subchain

INDEX=
Specifies the chain where the entry is placed. If “name” is coded, the name specified must be one of the terms defined for field RQINDEX in the IATYRSQ macro. If omitted, the index in the entry is used. The INDEX parameter parameter and the PARM parameter are mutually exclusive.

NORMAL=
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

Linkage
This macro uses ASAVE linkage.

Restrictions
Do not use this macro in any routine that might run under the JES3 auxiliary task.

RQTDELETE (Delete a RESQUEUE from a RESQUEUE Chain and Subchain)

The RQTDELETE macro deletes a resident job queue table (RESQUEUE) from a RESQUEUE chain and subchain.

Syntax

\[
\text{ENTRY=}[\text{address}], \text{NORMAL=}[\text{address}]
\]

Parameters

ENTRY=
Specifies the address of the RESQUEUE entry to be deleted.

NORMAL=
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

Linkage
This macro uses ASAVE linkage.

Restrictions
Do not use this macro in any routine that might run under the JES3 auxiliary task.
RQTAGEN (Inner Macro to Manipulate RESQUEUE Table)

The RQTAGEN inner macro is used within RQTAADD, RQTADEL, and RQTAPUT to manipulate resident job queue tables (RESQUEUEs). This macro can also map a parameter list used with the RQTAADD and RQTAPUT macros.

Syntax

```
ENTRY=(R1), TYPE=ADD, INDEX=name
```

Parameters

**ENTRY=**

Specifies the address of the RESQUEUE entry to be manipulated.

**TYPE=**

Specifies whether to add, delete, or move (PUT) a RESQUEUE in a RESQUEUE chain and subchain. DSECT specifies that a mapping of the parameter list used with the RQTAADD, RQTAGEN, and RQTAPUT macros should be generated. This parameter must be used alone (without any other parameters).

**PARM=**

Specifies the register containing the address of a user-defined parameter list. The parameter list, which is mapped by this macro specifying TYPE=DSECT, must contain (in the following order):

- The address of the entry to be added to the RESQUEUE
- The index, which must be a fullword containing one of the terms defined for field RQINDEX in the IATYRSQ macro
- The base address of the previous RESQUEUE subchain
- The base address of the new RESQUEUE subchain

**INDEX=**

Specifies the name of the index indicating the status of the entry. See the IATYRSQ mapping macro for the list of index names. If this parameter is omitted, the index in the entry itself is used.

**NORMAL=**

This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

Linkage

This macro uses ASAVE linkage.
RQTAGEN

Restrictions

Only the RQTAADD, RQTADEL, and RQTAPUT macro expansions can use this macro. The RQTADEL macro can use only the TYPE, ENTRY and NORMAL parameters.

RQTAPUT (Move a RESQUEUE to Another Chain)

The RQTAPUT macro moves a resident job queue table (RESQUEUE) from one chain to another or changes the priority within a chain.

Syntax

```
RQTAPUT
  ENTRY=(R1)
  INDEX=(reg)
  PARM=(reg)
  NORMAL=address
  NEWPRTY=(reg)
```

Parameters

ENTRY=
  Specifies the address of the RESQUEUE entry.

INDEX=
  Specifies the chain into which the entry is to be placed. If you code a name, the name specified must be one of the terms defined for field RQINDEX in the IATYRSQ macro. If you specify a register, the register must contain one of the values defined for field RQINDEX in the IATYRSQ macro.

If this keyword is omitted, the index in the IATYRSQ is used. This parameter is not valid with the PARM= keyword.

PARM=
  Specifies the register containing the address of a user-defined parameter list. The parameter list, which is mapped by the RQTAGEN macro specifying TYPE=DSECT, must contain (in this order):
  • the address of the RESQUEUE to be moved
  • the index, which must be a fullword containing one of the terms defined for field RQINDEX in the IATYRSQ macro
  • the base address of the previous RESQUEUE subchain
  • the base address of the new RESQUEUE subchain

NORMAL=
  This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

NEWPRTY=
  This parameter specifies the new priority in the RQ chain.

Linkage

This macro uses ASAVE linkage.
Restrictions

Do not use this macro in any routine that might run under the JES3 auxiliary task.

SPINOFF (Process Output Prior to Job Completion)

The SPINOFF macro requests output service processing of an output data set before termination of the originating job.

Syntax

```
label SPINOFF(R1) address,
RQ=(R1) address,
JDS=(R1) address,
FDB=(R1) address,
TATPTR=(R0) address,
NORMAL=address
```

Parameters

RQ=

Specifies the address of the resident job queue (RESQUEUE) for the executing job whose data sets are to be processed. All spin-off type job data set block (JDS) entries for this job will be processed. If coded, then the JDS and FDB parameters may not be coded. If all parameters are omitted then RQ is assumed and the RESQUEUE pointer in the FCT is used.

JDS=

Specifies the address of the job data set block entry to be processed. This type of request requires that a data set track allocation table (TAT) has been allocated for this JDS. Once this type of request has been issued, the original JDS is cleared and the data set is under the exclusive control of output service. If coded, the RQ and FDB parameters may not be coded.

FDB=

Specifies the address of the FDB for the data set to be processed. This type of request creates a JDS using the system default print attributes and the data in the FDB; the data set is under the exclusive control of output service. The data set name used is JES3.SYSTEM.OUTPUT. If coded, then the RQ and JDS parameters may not be coded.

TATPTR=

Specifies the address of the associated track allocation table (TAT) FDB. Valid and required with the FDB parameter only.

NORMAL=

This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

Linkage

This macro uses ASAVE linkage.
Restrictions
Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task.

SSISERV (Provide Interface for SSI and JES3)
The subsystem interface (SSI) function routines and the JES3 subtasks use the SSISERV macro to communicate with JES3.

Syntax

Parameters

FUNC=

(name
hexexp
Specifies either the subsystems options block (SSOB) function code of the requester or, if the originator is JES3, a JES3 destination code.

(Rx)
Specifies the address at which either the SSOB function code or the JES3 destination code can be obtained.

Do not specify the FUNC= parameter if you specify TYPE=RESP or TYPE=PURGE.

PARM=

Specifies the address of service entrance list (SEL) control block. The following addresses must be contained in the SEL:

- Data area address – The first 2 bytes of the data area must contain the data length (maximum length of 3800 bytes). The corresponding response can be no longer than the original data.
- ECB/ECF address
- Exit routine address – The address of a routine to be given control when a response is received. If no buffer was supplied, register 1 contains upon entry the address of the response buffer or the staging area containing the response.
- Response buffer address – The address of a buffer for the response. The first 2 bytes must specify the buffer data length.
- Staging area address.

**TYPE=**

**COMM**

Specifies that the request is for communication only. The data area address must be contained in the SEL.

**WAIT**

Specifies that a response is required and that the routine is to wait until the response is received. The data area, ECB, and buffer addresses must be in the SEL.

**REPLY**

Specifies that a response is required but the routine is not to wait for the response. The data area address must be in the SEL. If the response buffer address is specified without the exit routine address, the ECB/ECF address is required. If both the response buffer address and the exit routine address are specified, then the ECB/ECF address must not be specified. Also, if only the exit routine address is specified, then the ECB/ECF address must not be specified.

**ACK**

Specifies that the request is for communication and that the request requires an acknowledgment. The RESP subparameter is used to make the acknowledgment.

**RESP**

Specifies the answer to a WAIT or REPLY request, or the acknowledgment to an ACK request. The staging area address is required. If you specify TYPE=RESP, do not specify the FUNC parameter.

**PURGE**

Specifies that a staging area that had been routed to the requester is to be deleted. The staging area address in the SEL is required. If you specify TYPE=PURGE, do not specify the FUNC parameter.

**EOMT**

Specifies the special interface from EOM/EOT SSI for staging area cleanup. The data area address in the SEL must contain the address of the SSIB/SSOB control block pair.

**MOD=**

Specifies a modifier in the SEL which further defines the function code.

**SYSID=**

Specifies the identification of the processor to which the request is being sent. This parameter is required if the receiver is not the global processor. The SYSID is contained in the applicable MPC.

**PRIORITY=**

Specifies the priority of the request. The priority will determine where the request is queued on the destination queue. PRIORITY=0 is the default.
NORMAL=
This parameter specifies that after the macro processing routine executes, the
issuing program can resume execution at an instruction other than the
instruction that immediately follows the macro.

Linkage
This macro issues a BALR R14,R15 for linkage.

Restrictions
Do not use this macro in any routine that might run under the JES3 auxiliary task.

Considerations for TYPE=REPLY with ECB/ECF
When you specify TYPE=REPLY and provide the address of an ECB/ECF, the
following restrictions apply to the storage location and structure of the response
buffer:
  • The response buffer must reside in a common area subpool (CSA or ECSA) with
    a storage protect key of 1.
  • The beginning of the response buffer must include space for the JESXCF
    acknowledgment message header and the JES3 staging area header, because
    both of these data areas and the response data itself are returned.
    The size of the JESXCF acknowledgment message header (YIXAC_LENGTH)
    can be obtained by including the IXZYIXAC mapping macro in your program. The
    size of the JES3 staging area header (STAHDLEN) can be obtained by including
    the IATYSTA mapping macro in your program.
    For information about JESXCF and the IXZYIXAC macro, see [z/OS MVS
    Programming: JES Common Coupling Services]
  • Before processing the response data, check the return code in the JESXCF
    acknowledgment header (YIXAC_APPL_RETURN_CODE). A non-zero return
    code indicates the SSISERV request did not complete properly, and the response
    data should not be processed.

The following code sample demonstrates how to define and obtain the storage for
the response buffer:
Define and Obtain Response Buffer Storage

* Include STA and YIXAC mappings
* IATYSTA Mapping of Staging Area
  IXZYIXAC Mapping of Acknowledgement

* Definition of SSISERV response data
* MYRESP DSECT Response data
MYRLNGTH DC H'0' Length of response data
MYRID DC CL4'MYR' Area identifier
.
  Other response data fields go here
.
MYRENDEQU* End of response data
MYRSIZE EQU MYRENDEMYRESP Size of response data

* Declares for response buffer sections
* MYBUFLEN EQU YIXAC_LENGTH+STAHDLEN+MYRSIZE Size of the
  Response buffer needed for
  SSISERV TYPE=REPLY
HEADLEN EQU YIXAC_LENGTH+STAHDLEN Size of the header
  information returned in
  the response buffer
.
* Obtain storage for response buffer
* STORAGE OBTAIN,
  Get storage for the SSISERV
  LENGTH=MYBUFLEN, Response Buffer
  SP=241, Must reside in CSA
  LOC=ANY, Above or below 16M
  KEY=1 Key 1 storage
  ST R1,MYBUFFER Save buffer address

When the ECB/ECF is posted, indicating that the SSISERV response has been received, JES3 can process the response buffer. The following example demonstrates how JES3 checks the return code in the JESXCF acknowledgment header and how JES3 addresses the actual response data:
JES3 Checks Return Code

* Check JSERV return code
*  
  L  R14,MYBUFFER       Address of response buffer
USING IXYIXAC,R14       Assembler addressability
ICM  R15,B'1111',YIXAC.APPL_RETURN_CODE Get return
     code from JSERV
BC  NZERO,NOTOK         Branch if not OK
DROP  R14               IXYIXAC
LA  R3,HEADLEN(R14)     Point to response data
USING MYRESP,R3          Addressability for response

* The return code is 0. The response data can
  be processed.
*  
  .
  .
  
  DROP  R3               MYRESP

SUPFLAG (Set or Reset Flag in SUPUNITS)

The SUPFLAG macro sets or resets flags in the SUPUNITS table.

Syntax

SUPFLAG-bit-name

-label-

Parameters

bit-name

Specifies the bit (defined by using the IATYSUP macro) to be modified.

OFF

ON

Specifies whether the bit is to be turned on or off.

SUPFLAGX
SUPFLAG1
SUPFLAG2

Specifies the name of the flag byte to be modified.

Linkage

This macro generates inline code.

Restrictions

Do not use this macro in any routine that might run in a C/I FSS address space or under the JES3 auxiliary task.
**VIOLATE (Execute Instructions Violating Storage Protection)**

The VIOLATE macro executes instructions which violate storage protection. It may also be used to alter the setting of the system mask and protection key.

**Syntax**

```
VIOLATE

label

ENTER= (R1)

KEY=ZERO

address

KEY=JES

KEY=ZERO

NORMAL= address
```

**Parameters**

**ENTER=**
规定的地址 appendage 包含违反存储保护的指令。appendage 不应包含任何 supervisor call 或其他状态切换指令，并且必须通过在寄存器 14 上返回的方式终止，以恢复寄存器 1。如果省略，只会使用 protect 功能。

**KEY=**

规定保护密钥应设置为 JES3 地址空间的密钥或零。如果 ENTER 指定，KEY=ZERO 假定，KEY=JES 是不正确的。如果 ENTER 省略，KEY=JES 假定，但两种指定都是有效的。

**NORMAL=**

此参数指明在宏处理程序执行后，发出程序可以恢复执行在指令的其他指令之后，立即紧随宏之后执行。

**Linkage**

此宏会发出 BALR R14,R15 进行连接。

**Restrictions**

不要在任何可能在 JES3 辅助任务下运行的程序中使用此宏。

**WRITEOSE (Write an OSE to spool)**

使用 WRITEOSE 宏在任何写入 OSE 到 spool 的程序中。

**Syntax**

```
WRITEOSE

label

TYPE= AWRITE

TATPTR= WRTCHAIN

FDB= TATPTR
```

Chapter 2. JES3 Macros  579
WRITEOSE

Parameters

TYPE=
Specifies the type of OSE your routine is writing to spool.

AWRITE
Specifies that your routine is writing a single OSE to spool.

WRTCHAIN
Specifies that your routine is writing a chain of OSEs to spool.

FDB=
Specifies the address of the FDB for the OSE to be written.
You may not specify register 0 or a register specified for TATPTR=.

TATPTR=
Specifies the address of the track allocation table FDB. This parameter is required when TYPE=WRTCHAIN.
You may not specify register 1 or a register specified for FDB=.

RQ
Specifies the address of the job's resident job queue (RESQUEUE) element.

PUTBUF=
Specifies whether to return the buffer to the buffer pool.

YES
Specifies that the buffer is to be returned.

NO
Specifies that the buffer is not to be returned.

NORMAL=
Specifies the return address for the macro processing routine. If NORMAL= is not specified, the routine returns to the instruction immediately following the macro.

ERROR=
Specifies the return address for the macro processing routine if an error occurs. If ERROR= is not specified, the dynamic support program (DSP) will be ended by using the FAILDSP macro.

Note: If you specify the ERROR= parameter with PUTBUF=YES, the routine you specify on the ERROR= parameter will only receive control if the error occurs in AWRITE or WRTCHAIN processing.
**Linkage**

This macro uses ASAVE linkage.

**Restrictions**

Do not use this macro in any routine that might run under the JES3 auxiliary task.

**WRTCHAIN (Write Chained Single Record Files)**

The WRTCHAIN macro is used to write a group of chained single record files (SRFs).

**Syntax**

```
/SM590000/SM590000
WRTCHAIN
label
FDB=(R1)address
TATPTR=(R0)address
DISP=(reg)number
PUTBUF=YES
ID=(reg)name
OPTION=MODONLY
ERROR=address
NORMAL=address
/SM590000/SM590000
```

**Parameters**

- **FDB=**
  Specifies the address of the FDB of the first SRF in the chain.

- **TATPTR=**
  Specifies the address of the job or data set track allocation table (TAT) FDB to be used. If the SRFs obtain their tracks from the single track table (STT), FDB=MNTRKFDB should be coded. When the SRF chain consists entirely of records belonging to the job’s TAT, the address of the job TAT FDB should be coded.

- **DISP=**
  Specifies the displacement into the SRF of the chain FDB, or a register containing the displacement in the low-order byte.

- **PUTBUF=**
  Specifies whether buffers are to be returned.

- **ID=**
  Specifies the name to be associated with the SRF. The ID length is limited to 4 characters. The ID is compared to the ID in the buffer before write.

- **OPTION=**
  Specifies that only modified buffers are to be written. If the PUTBUF=NO parameter is also specified, unmodified buffers are left unchanged. If the PUTBUF=YES parameter is specified with this parameter, unmodified buffers are released. The requester marks modified buffers with flag DMCDATMD in flag byte DMCFLAG4.

  If the MODONLY option is specified, WRTCHAIN issues an IATXSIO macro for each record marked by the caller or by WRTCHAIN when new records were
WRTCHAIN

added. If the MODONLY option is omitted, WRTCHAIN flags all old records as modified and rewrites the entire chain by issuing IATXSIO macros.

After issuing the last IATXSIO, WRTCHAIN waits for the “I/O completed bit” in the SRF File Directory Entry.

ERROR=
This parameter specifies the address to which the macro processing routine returns if an error occurs. When control is passed to the address, register 0 contains a dump code and register 15 contains a return code, unless otherwise specified. The return codes and dump codes are described below. For a description of every dump code, see z/OS JES3 Diagnosis.

If an error occurs and ERROR was not specified, the dynamic support program (DSP) will be ended by using the FAILDSP macro.

NORMAL=
This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

Return Codes
Following are the return codes and dump codes returned in registers 15 and 0, respectively:

<table>
<thead>
<tr>
<th>Return Code in Hex</th>
<th>Dump Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>08</td>
<td>703</td>
<td>Root FDB containing a spool address</td>
</tr>
<tr>
<td>0C</td>
<td>721</td>
<td>Incorrect buffer or spool address (DISK)</td>
</tr>
<tr>
<td>10</td>
<td>724</td>
<td>Recovered I/O error, should checkpoint FDB</td>
</tr>
<tr>
<td>14</td>
<td>725</td>
<td>Unrecoverable I/O error</td>
</tr>
<tr>
<td>18</td>
<td>725</td>
<td>Error during ROOT I/O error recovery</td>
</tr>
<tr>
<td>1C</td>
<td>703</td>
<td>Zero FDB</td>
</tr>
<tr>
<td>20</td>
<td>703</td>
<td>IDs did not match</td>
</tr>
<tr>
<td>28</td>
<td>729</td>
<td>Spool space could not be allocated because of a job or data set TAT error</td>
</tr>
<tr>
<td>30</td>
<td>736</td>
<td>SRF identifier is incorrect</td>
</tr>
<tr>
<td>34</td>
<td>703</td>
<td>Request for a multirecord file</td>
</tr>
<tr>
<td>34</td>
<td>736</td>
<td>CSBT identifier is incorrect</td>
</tr>
<tr>
<td>3C</td>
<td>736</td>
<td>CSBT used with a previous JESREAD</td>
</tr>
<tr>
<td>38</td>
<td>736</td>
<td>CSBT not used with a previous JESREAD</td>
</tr>
<tr>
<td>40</td>
<td>736</td>
<td>Unrecoverable I/O error</td>
</tr>
</tbody>
</table>

Linkage
This macro uses ASAVE linkage.

Restrictions
Do not use this macro in any routine that might run under the JES3 auxiliary task.

Do not use this macro in any routine that writes OSEs to spool. Use the WRITEOSE macro instead.
**ZEROCORE (Clear Area to Binary Zeros)**

The ZEROCORE macro clears an area to binary zeros. However, the MVCL assembler instruction performs this function more efficiently and should be used instead.

**Syntax**

```
ZEROCORE label
   AREA=address
   (R3)
   COUNT=address
   (R4)
   NORMAL=address
```

**Parameters**

- **AREA=**
  Specifies the address of the area to be cleared.

- **COUNT=**
  Specifies the number of bytes to be cleared.

- **NORMAL=**
  This parameter specifies that after the macro processing routine executes, the issuing program can resume execution at an instruction other than the instruction that immediately follows the macro.

**Linkage**

This macro issues a BALR R14,R15 for linkage.

**Restrictions**

Do not use this macro in any routine that might run under the JES3 auxiliary task.

**Table Generating and Other Macro Instructions**

This section describes the JES3 macros that are used to generate control blocks and to map data areas. The macros that are necessary to build the JES3 system are described with all their operands.

**IATYASM (Identify Level of MVS Macros)**

The IATYASM macro is used to identify the default level of MVS macros used by JES3. IATYASM sets the default SP level for all JES3 modules to make use of MVS/ESA, OS/390, or z/OS MVS capabilities. IATYASM generates an SPLEVEL SET=5 macro in JES3 modules.

In MVS/ESA, OS/390 or both, some macros had to be changed to support MVS/ESA and OS/390 MVS capabilities. To provide compatibility for existing code, both the old and the new forms of the macro are available in MVS/ESA and OS/390 MVS. The SPLEVEL macro is the way to specify which level of the macros to use when assembling a module. The SPLEVEL macro can be used in a module so that different paths in a module can generate different levels of a macro. New releases
of MVS will add SPELEVEL values if they create any downward incompatible macros. The current default level is 6 which is the OS/390 Version 1 Release 2 level and is also in effect for z/OS.

The IATYASM macro must be included in a module before any mapping macros or executable code. This allows the assembler global symbol set by SPELEVEL to be established before any code invokes an MVS downward-incompatible service. The JES3 convention is to code the IATYASM macro immediately after the AMODE and RMODE statements being added to every JES3 module. By convention, these statements must be placed at the beginning of a module before the prologue.

Syntax

```
IATYASM
   SPELEVEL=
```

Parameters

**SPELEVEL=**

Specifies a value to override the current JES3 default value generated as the SET= parameter on the MVS SPELEVEL macro. The only valid values that can be specified are 1, 2, 3, 4, 5, and 6. If you omit the SPELEVEL= parameter, the default value used by IATYASM is 6. If you specify any other value the module issuing this macro will fail to assemble correctly.

With no parameter specified, the IATYASM macro generates an SPELEVEL SET=6 statement. JES3 modules use the IATYASM macro instead of the MVS SPELEVEL macro to set the JES3 default value. If a module needs to change the level for some path in the module, it uses the MVS SPELEVEL macro to change and restore the level.

**IATYAWDR (Generate an AWAIT Reason Code)**

The IATYAWDR macro generates AWAIT reason codes within the IATYAWR macro.

You can define installation AWAIT reason codes to distinguish between different AWAITs by using the IATYAWDR macro. The AWDCODE parameter and the label uniquely identify the AWAIT reason code.

The IATYAWDR macro is located within the IATYAWR macro.

Syntax

```
IATYAWDR
   label
   OPTION=OPTION
   AWDCODE='nnnn'
   AWDTEXT='text'
   BOTTLENECK=NO
```

584  z/OS V1R11.0 JES3 Customization
Parameters

**label**
Specifies the 1- to 8-character name of the AWAIT reason code whose entry is being created by this macro.

The label is required.

**OPTION=&OPTION**
Specifies that the value specified on the OPTION= parameter of the IATYAWR macro is used by the IATYAWDR macro.

**AWDCODE='nnnn'**
Specifies a four-digit reason code (in hexadecimal). Be sure that the reason code you specify is not already in use.

Installation AWAIT reason codes should be between X'F000' and X'FFFF'. AWDCODE values X'0000' through X'EFFF' are reserved for IBM.

**AWDTEXT='text'**
Explanation text for the AWAIT reason code. The text character length must be no more than 100 bytes long.

The text must be in all capitals so that IPCS can output it correctly.

**BOTTLENECK=**
An optional parameter that specifies whether the AWAIT is a potential bottleneck. This is used by JMF to determine which AWAITs associated with an FCT are bottlenecks. For example, an AWAIT that involves waiting for resources to become available, such as job numbers, buffers, can be considered a bottleneck.

Example

The following is an example of how to code an installation AWAIT reason code in IATYAWR using IATYAWDR.

```
Example
AWRMYRSN IATYAWDR OPTION=&OPTION, +
     AWDCODE='F001', +
     AWDTEXT='REASON MY CODE IS WAITING'
```

IATYAWR (JES3 AWAIT Reason Codes)
The IATYAWR macro defines the AWAIT reason codes that are used when the AWAIT macro is issued with the REASON= parameter. The AWAIT reason code uniquely identifies why a module is waiting.

Notes:
1. Similar AWAIT reasons can be grouped under one AWAIT reason code. For example, almost every FCT has an AWAIT for waiting for work. These AWAITs can use the AWRWT4WK reason code.
2. If you include the IATYFCT macro in your module, you do not have to include IATYAWR because IATYAWR is automatically included.
Parameters

OPTION={DEFNTABLE|USETABLE|NONE}
You can generate IATYAWR in three different forms. The OPTION= parameter specifies which form you want to generate.

DEFNTABL
Specifies that you want to generate an AWAIT reason code table that contains AWAIT reason codes and corresponding descriptive text.

You can search this table to obtain the descriptive text for an AWAIT reason code to clarify why a function is waiting.

USETABLE
Specifies that you want to search the AWAIT reason code table. USETABLE generates the mapping for a table entry.

Note: In order to search the table, you must have the table generated by the DEFNTABL option.

NONE
Specifies that you want to generate the JES3 AWAIT reason code equates defined in IATYAWR.
NONE is the default.

Examples of Use

Example 1
The following is an example of searching the AWAIT reason code table using IATYAWR:

IATYAWR OPTION=USETABLE Create table entry mapping

*-------------------------------------------------------------*
* Set up to search the AWAIT reason code table.               *
* R5 keeps track of the current position in the table.         *
* R4 contains the reason code for which we are searching.      *
*-------------------------------------------------------------*
LA R5,AWRTABLE R5 starts at table begin
USING DSAWRWRK,R5 Use mapping of table entry

*-------------------------------------------------------------*
* Loop through each entry in the table until we find the       *
* reason code we're searching for or the end of the table.      *
*-------------------------------------------------------------*
LOOP DS 0H
CLM R4,'B'0011','DSAWRCOD Check if this entry matches
BC EQ,FOUND If match, go process it

*-------------------------------------------------------------*
* The current entry did not match, check if we have reached    *
* the end of the table. If not, loop back and continue.        *
*-------------------------------------------------------------*
CLC MAXRSNCD,DSAWRCOD Check max against current
BC EQ,NOTFOUND At table end, not found
LA R5,DSAWRNXT Increment to the next entry
Loop back and continue

*-------------------------------------------------------------*
* Found a matching code, can now access reason text using  *
* field DSAWRTXT.                                         *
*-------------------------------------------------------------*
FOUND  DS  OH

* Reason code not found, handle this condition.              *
*-------------------------------------------------------------*
NOTFOUND DS  OH

* Constants and work areas                                  *
*-------------------------------------------------------------*
MAXRSNCD DC  AL2(AWRUSMAX) Maximum reason code
             IATYAWR OPTION=DEFNTABL Create reason code table in
             *                                                 
             *                                                 
             *                                                 
             storage

Example 2
The following is an example IATYAWR generating JES3 AWAIT reason
code equates. If you want a function to issue an AWAIT to “wait for work”, code the
following:

IATYAWR OPTION=None Generate AWAIT reason code equates

: 

: 

AWAIT ECFADD=MYECF, Wait for a post +
ECFMASK=MYWT4WK, Waiting for work +
TYPE=ON, +
REASON=AWRWT4WK Include reason for wait

IATYCNS (Generate Console Service Tables)
The IATYCNS macro generates various tables used to communicate information
between console service and other functions within JES3.

Syntax

```
IATYCNS

label CODES=YES

-CODES=NO

TYPE=:

INPUT

-SPOOL

-DLOG

-FCTQ

```

Parameters

**TYPE=**

Defines the tables to be generated. To generate multiple tables, group the
parameters within parentheses; for example, TYPE=(INPUT,SPOOL). The first
occurrence of IATYCNS within a CSECT also causes a set of buffer control equate statements to be generated. If CODES=NO is specified, the TYPE parameter is required.

**INPUT**
Input message buffer

**DLOG**
JES3 format hardcopy log mapping

**FCTQ**
Action message buffer

**SPOOL**
Console spool message buffer

**CODES=**
If TYPE=INPUT is specified, CODES=YES causes the generation of a group of equate statements defining the console input command action codes (for example, START, CALL, VARY). CODES=NO may be used to suppress generation of these statements with TYPE=INPUT; however, it is incorrect if the TYPE parameter is omitted.

---

**IATYDJF (Define a Field Within a JES3 Spool-Resident Data Area)**

The IATYDJF macro defines a field within a JES3 spool resident data area to the dump job facility. The dump job facility uses the field definitions when translating a job’s control blocks from one release level to another.

This macro must be preceded by an IATYDJR macro that defines the JES3 spool-resident data area. Several IATYDJF macros may follow a single IATYDJR macro, however. See the IATYDJR macro for an example of a correct macro sequence.

To obtain a DSECT mapping of the control block generated by the IATYDJF macro, do not specify the ‘fieldnm’ label or any parameters.

**Syntax**

```
IATYDJF (fieldnm, fmid, change, action, address)
```

```
OFFSET=nnn, LEN=nnn, TYPE=, EXIT=, COND=, END
```

**TYPE=:**
EXIT=:

(EXIT=(address, address))

COND=:

(COND=(fieldnm, EQ, operand, BYPASS, EXIT))

Parameters

**fieldnm**
Specifies the name of the data area field being defined. This name must match the label name defined for the field in the JES3 mapping macro for the data area.

**fmid**
Specifies the FMID of the JES3 release to which these modifications apply. A SETC assembler macro instruction must be included in the IATDJCR module to set the FMID into one of the previous release name variables:

&OLDREL1
&OLDREL7

**change**
Specifies the type of change in the field. Any of the following constants can be specified:

**ADDED**
Specifies that the field was added in the current JES3 release. The field did not exist or was defined as TYPE=DUMMY in the JES3 release identified by the ‘fmid’ parameter. Only the EXIT, BLANK, ZERO, or BYPASS ‘action’ parameter option is valid with this ‘change’ option.

**DELETED**
Specifies that the field existed in the JES3 release identified by the ‘fmid’ parameter. The field was deleted or defined as TYPE=DUMMY in the
current JES3 release. Only the EXIT, BLANK, ZERO, or BYPASS ‘action’ parameter option is valid with this ‘change’ option.

**RENAME**
Specifies that the field was renamed in the current JES3 release. Following this option, the name of the field in the JES3 release identified by the ‘fmid’ parameter must be specified; that is, specify RENAMED,oldname. The ‘action’ parameter is not valid with this option.

**LENGTH**
Specifies that the length of the field was modified in the current JES3 release. Only the LEFT0, LEFTBL, RIGHT0, RIGHTBL, BYPASS, or EXIT ‘action’ parameter option is valid with this ‘change’ option.

**MASK**
Specifies that the field was defined as TYPE=B (binary data) and that the binary bit mask was modified in the current release. Only the EXIT, AND, or OR ‘action’ parameter option is valid with this ‘change’ option.

**TYPE**
Specifies that the field type was modified in the current release. Only the EXIT ‘action’ parameter option is valid with this option.

**EXIT**
Specifies that translation of the field is to be bypassed. Following this option, the address of an exit routine that will perform the translation for the field must be specified; that is, specify EXIT,address. The ‘action’ parameter is not valid with this option.

**action**
Specifies the type of action that should be taken as a result of the change indicated by the ‘change’ parameter. Any of the following options can be specified:

**EXIT**
Specifies that the following positional parameter is the address of a field modification exit routine in module IATDJCR that is to perform the translation for the field.

**BLANK**
Specifies that, for the dump job facility input processing, the field is to be initialized to blanks. For the dump job facility output processing, the field is to be ignored.

**ZERO**
Specifies that, for the dump job facility input processing, the field is to be initialized to binary zeros. For the dump job facility output processing, the field is to be ignored.

**LEFT0**
If the data in the field being translated (the input field) is longer than the output field, the data is to be truncated on the left. If the data in the input field is shorter than the output field, the data is to be right-justified and left-filled with binary zeros.

**LEFTBL**
If the data in the field being translated (the input field) is longer than the output field, the data is to be truncated on the left. If the data in the input field is shorter than the output field, the data is to be right-justified and left-filled with blanks.
RIGHT0
If the data in the field being translated (the input field) is longer than the output field, the data is to be truncated on the right. If the data in the input field is shorter than the output field, the data is to be left-justified and right-filled with binary zeros.

RIGHTBL
If the data in the field being translated (the input field) is longer than the output field, the data is to be truncated on the right. If the data in the input field is shorter than the output field, the data is to be left-justified and right-filled with blanks.

BYPASS
Specifies that translation of the field is to be bypassed. The output field will contain binary zeros.

AND
Specifies that all incorrect bits in the bit mask (defined by the MASK parameter) are to be turned off.

OR
Specifies that all incorrect bits in the bit mask (defined by the MASK parameter) are to be turned off. Also, any bits not defined in the bit mask for the definition of the input field are to be turned on.

address
Specifies the address of a field modification exit routine in module IATDJCR that will perform the translation for the field. This parameter is a positional parameter that is required with the EXIT 'action' parameter option.

OFFSET=
Specifies the offset from the start of the data area to the start of the field being defined. The value may be a label, an arithmetic expression, or a decimal number. If this parameter is omitted, the offset is calculated as the offset difference between the data area field defined by the 'fieldnm' label and the data area 'base' parameter that was defined on the preceding IATYDJR macro statement.

LEN=
Specifies the length of the field being defined. The value may be a label, an arithmetic expression, or a decimal number. If this parameter is omitted, the length is calculated using the assembler language length attribute expression: L'fieldnm.

TYPE=
Specifies the type of data contained in the field. If this parameter is omitted, the data type is taken from the data area mapping macro that corresponds to the 'acronym' label specification on the prior IATYDJR macro.

C The field contains character data.
X The field contains hexadecimal data.
P The field contains packed decimal data.
B The field contains binary data used as single bit indicators. The field must be a single byte long and the MASK parameter must be specified.

FDB
The field contains an FDB for a single record file (SRF).

MRF
The field contains an FDB for a multirecord file (MRF).
The field does not actually exist or the field is not to be processed during data translation.

MASK=
Specifies, for fields defined as TYPE=B, the bits within the flag byte that are valid.

   ALL
   All bits in the byte are valid.

   0
   No bits in the byte are valid.

bitmask
The indicated bits are valid. To generate the bit mask using the equated bit names, the MASK parameter should be specified as
MASK=bitname1+bitname2+bitname3....

EXIT=
Specifies the addresses of the exit routines within module IATDJCR to be called during translation of the field.

The first address is the address of the exit routine to be called before translation of the field. For input processing, the exit routine may examine or modify the location of the input field, the length of the input field, or the contents of the input field. For output processing, the exit routine may examine or modify the location of the output field, the length of the output field, or the contents of the output field.

The second address is the address of the exit routine to be called after translation of the field. For either input or output processing, the exit routine may examine or modify the contents of the input or output fields.

COND=
 Specifies the conditions under which translation of the field is to occur. This parameter consists of four positional parameters:
 1.   fieldnm, the name of the field to be tested
 2.   The operation (EQ, NE, ALLON, or ALLOFF), the type of test to be performed
 3.   The operand or mask, the data to be used in the test
 4.   The action (BYPASS, EXIT, or ERROR) to be performed if the test is successful.

fieldnm
Specifies the name of the field within the data area upon which the operation is to be performed. This field name must have been defined by an IATYDJF macro within the scope of the current IATYDJR macro.

EQ
Specifies that the field specified by the ‘fieldnm’ subparameter is to be compared against the value specified by the ‘operand’ subparameter. If the values are equal, then the action specified by the fourth subparameter is to be taken.

NE
Specifies that the field specified by the ‘fieldnm’ subparameter is to be compared against the value specified by the ‘operand’ subparameter. If the values are not equal, then the action specified by the fourth subparameter is to be taken.

ALLON
Specifies that a test-under-mask is to be made on the first byte of the field
specified by the ‘fieldnm’ subparameter, using the bit mask value specified by the ‘operand’ subparameter. If all the bits tested by the mask are on, then the action specified by the fourth subparameter is to be taken.

**ALLOFF**

Specifies that a test-under-mask is to be made on the first byte of the field specified by the ‘fieldnm’ subparameter, using the bit mask value specified by the ‘operand’ subparameter. If all the bits tested by the mask are off, then the action specified by the fourth subparameter is to be taken.

**operand**

Defines the data that is to be compared to the field specified by the ‘fieldnm’ subparameter. The operand must be specified as the data would appear in an assembler DC instruction, for example, CL3’ABC’ or F’2’. This subparameter can be specified only with the EQ and NE subparameters.

**mask**

Defines the binary mask that is to be tested against the field specified by the ‘fieldnm’ subparameter. This subparameter must be specified as it would appear in an assembler DC instruction for an address constant, for example, BIT1+BIT2 or 128. This subparameter can be specified only with the ALLON and ALLOFF subparameters.

**BYPASS**

Specifies that further processing of the field specified by the ‘fieldnm’ subparameter is to be bypassed. The field is not to be moved from the input data area to the output data area. If an output field exit routine is defined by the EXIT parameter, it will not be called.

**EXIT**

Specifies that further processing of the field specified by the ‘fieldnm’ subparameter is to be bypassed. The field is not to be moved from the input data area to the output data area. If an output field exit routine is defined by the EXIT parameter, it will be called.

**ERROR**

Specifies that the dump job DSP is to be abnormally ended.

**END**

Specifies that the previous IATYDJF macro was the last IATYDJF macro within the current data area to define a field to the dump job data area translator. No other parameters are valid with this parameter.

---

**IATYDJR (Define a JES3 Spool-Resident Data Area)**

The IATYDJR macro defines a JES3 spool-resident data area to the dump job facility. The dump job facility uses the data area definition when translating a job’s control blocks from one release level to another.

Each IATYDJR macro (except the IATYDJR macro specifying the END parameter) must be followed by one or more IATYDJF macros describing the fields with the data area. When all the data areas to be supported by the dump job facility control block translator have been defined, a final IATYDJR macro specifying the END parameter must be issued.

Following is an example of a correct sequence of IATYDJR and IATYDJF macros:
To obtain a DSECT mapping of the control block generated by the IATYDJR macro, do not specify the 'acronym' label or any parameters.

**Syntax**

```
acronym IATYDJR base,.....
fieldnm IATYDJF (fmid,change,action,address),.....
fieldnm IATYDJF (fmid,change,action,address),.....
IATYDJF END
acronym IATYDJR base,.....
fieldnm IATYDJF (fmid,change,action,address),.....
fieldnm IATYDJF (fmid,change,action,address),.....
IATYDJF END
IATYDJR END
```

**Parameters**

- **acronym**
  Specifies the common acronym for the spool data area being defined. (For example, the common acronym for the job control table, IATYJCT, is JCT.) This acronym is passed by the dump job DSP as a parameter to the dump job translator, to identify the data area to be processed. The acronym specified on this macro must match the acronym used by the dump job DSP to refer to that data area.

- **base**
  Specifies the assembly label within the spool data area specified by the 'acronym' label that is to be used as the reference point for all fields. This label is used by the IATYDJF macro for calculation of the offset to the start of each field within the data area, unless the IATYDJF macro specifies the OFFSET parameter.

- **EXIT=**
  Specifies the address of the exit routines in module IATDJCR to be called during output translation of the data area.

  The first address specifies the address of the exit routine to be called before translation of the data area into the output buffer. The exit routine may initialize
constant fields in the output buffer that are not handled during translation and may optionally reject translation of the data area.

The second address specifies the address of the exit routine to be called after translation of the data area into the output buffer. This exit routine may examine or modify the translated fields within the output buffer and may optionally reject translation of the data area.

VAREXIT=
Specifies the addresses of the exit routines in module IATDJCR to be called during translation of spool records which have multiple variable length data area segments.

The first address specifies the address of the exit routine to be called after translation of this data area. The exit routine must determine if another data area exists within the same spool record. The exit routine is called for the input data area only.

The second address specifies the address of the exit routine to be called before translation of this data area, if the data area will not fit within the current output buffer. The exit routine for the current release is called.

LEN=
Specifies the length of the data area being processed. This value, if specified, is used during input and output translation processing of the data area.

LENFLD=
Specifies the name of a field within the data area that contains the actual length of the data area. This parameter is used only during input translation processing of variable length data areas. If this parameter is omitted, the length specified by the LEN parameter is used.

MAXLEN=
Specifies the maximum possible length of the data area being defined. This parameter is used only during output translation processing of variable data areas to determine if there is sufficient space in the current output buffer to hold the translated data area.

END
Specifies that the previous IATYDJR macro defined the last data area supported by the dump job data area translator. No other parameters are valid with this parameter.

IATYDSD (Generate a DSP Dictionary Entry)
The IATYDSD macro generates an entry for a dynamic support program (DSP) in the DSP dictionary (module IATGRPT or, in a C/I FSS address space, module IATGRPTF). An entry in the table is required for each DSP in order for it to be recognized as part of JES3.

Syntax

```
label IATYDSD PRTY=nnn,REENT=YES
```

Chapter 2. JES3 Macros 595
Parameters

**label**
Specifies the 1- to 8-character name of the DSP whose entry is being created by this macro. If the DSP is to be callable, this name will appear as the argument of an *X,dspname command. If the DSP is to be processable, this name will appear in the /*PROCESS dspname JCL statements. The label is required.

**PRTY=**
Specifies the priority to be assigned to the DSP, in the range from 1 through 255. For the WAIT DSP, the priority may be 0.

**REENT=**
**YES**
Specifies that the DSP is reenterable.

**NO**
Specifies that the DSP is not reenterable.

**XABLE=**
**YES**
Specifies that the DSP can be called by using the *CALL command.

**NO**
Specifies that the DSP cannot be called by using the *CALL command.

**NOREQ=**
Specifies the maximum number of devices required by the DSP, in the range from 0 to 5. The number of entries in the GETUNIT list associated with the function control table (FCT) will be equal to this value. This parameter should be specified when known (see note below).

**REQ=**
Specifies the types of devices known to be required by the DSP. The valid device types are specified on the DEVICE initialization statement. This parameter should be specified whenever the DSP unit requirements are known.
(see note below). If more than one device is required, the subparameter list must be enclosed in parentheses. For example, a DSP requiring two 9-track tape devices and a printer could specify NOREQ=3 and REQ=(TA9,TA9,PRT).

**DRVR=**
Specifies the name of the DSP driver module to be loaded, if necessary, for each use of the DSP.

**CSECT=**
Specifies the name of the data CSECT to be loaded by the job segment scheduler driver (module IATGRJR) for each use of the DSP.

**GETUNV=**
Specifies the required disposition of the DSP if devices are not available when the DSP issues the GETUNIT macro.

- **C** Indicates the DSP is to be canceled.
- **R** Indicates the DSP is to return for specialized rescheduling.

The DSPRESCH bit in the DSP dictionary may be tested during DSP execution to determine which option is required. All callable DSPs in the distributed JES3 system also provide an operator override by using a keyword parameter to alter the basic disposition for an individual job.

**PABLE=**
Indicates whether this DSP may be scheduled by using the JES3 //*PROCESS control statement.

**MAXCT=**
Specifies the maximum number of copies of this DSP that may be concurrently active. The number must be within the range 1 through 6,5534. If this parameter is not specified for a DSP specifying REENT=YES, no MAXCT limit is imposed. This parameter is dynamically alterable by using the *MODIFY command unless you specify MUCC=NO.

**MUCC=**

- **YES**
  Allows the maximum use count specified by MAXCT to be altered by using the *MODIFY command.

- **NO**
  Disallows maximum use count modification.

**HABLE=**

- **YES**
  Allows the DSP’s work to be held and released by the *F,X,D=dsp,HOLD and *F,X,D=dsp,RELEASE commands. For DSPs that have an FCT, the command takes effect immediately upon being issued. For DSPs having no FCT, the command prevents the job segment scheduler (JSS) from scheduling the DSP.

- **NO**
  Disallows the DSP’s work to be held and released by the *F,X,D=dsp,HOLD and *F,X,D=dsp,RELEASE commands.

**JOBVAL=**
Allows you to specify the name of the job validation routine for the defined DSP (that is, the scheduler element) to use for validation or restart processing.

**CRITICAL=**
YES
Specifies that this DSP is critical to the function of JES3. If this DSP
abnormally ends and cannot be recovered, the entire JES3 address space
will automatically restart.

NO
Specifies that this DSP is not critical to the function of JES3. JES3 can
continue to operate even if this DSP abnormally ends and cannot be
recovered.

Note: Parameters NOREQ and REQ should always be specified when NOREQ
and REQ are known because these values are used to determine the size
of the FCT used for job segment scheduling. In the distributed JES3 system,
DSPs are scheduled without devices and obtain them by using the GETUNIT
macro. The distributed JES3 DSP attributes, as defined above, combine to
make the JES3 DSP dictionary, which is contained in module IATGRPT or, in
a C/I FSS address space, module IATGRPTF.

Example of Use
The following is an example of IATYDSD that identifies a validation routine named
IATNDJV to JES3. The name of the DSP is NEWDSP.

NEWDSP  IATYDSD  PRTY=22,  FCT PRIORITY
DRVR=IATNDSP,  DRIVER MODULE NAME
CSECT=IATNDTA,  DATA CSECT NAME
JOBVAL=IATNDJV  JOB VALIDATION MODULE NAME

IATYFCD (Generate Function Control Table)
The IATYFCD macro generates the permanent function control table (FCT) entries
in the DSP dictionary (module IATGRPT or, in a C/I FSS address space, module
IATGRPTF) for JES3 functions such as: console service, the JES3 disk input/output
routines (JSAM), the main device scheduler (MDS), and the job segment scheduler
(JSS).

Syntax

```
IATYFCD.label
ECFADD=address, R10=--label--
R11=--label--
R12=--label--
R13=--label--
R14=--label--
R15=--label--
ANAME=--name--
SUCCEED=address
LOGIN=address
INISH=NO
PABLE=YES
REENT=NO
NOREQ=n
```
Parameters

ECFMASK=
Specifies the mask that indicates the bits (which represent events) to be tested in the event completion flag (ECF). The test is satisfied if any or all of the events are complete.

COND=
Specifies the hexadecimal value of the mask field of the branch-on-condition instruction used to test ECFMASK. For example, X'50' specifies AWAIT TYPE=ON (branch one or mixed); X'80' specifies AWAIT TYPE=OFF (branch zero).

PRTY=
Specifies the function priority. The value may be any decimal value from 001 to 254; 000 is reserved for WAIT and 255 is reserved for console service.

NAME=
Specifies the name by which the operator may communicate with the function.

ECFADD=
Specifies the label of the 1-byte ECF. If ECFADD is omitted, the field must be filled in dynamically before its use.

R10=
Specifies the external label of an instruction whose address will be in register 10 when the function is entered for the first time. This address is the entry point to the module.

R13=
Specifies the external label of an instruction whose address will be in register 13 when the function is entered for the first time. This address is that of the module’s data CSECT.

R14=
Specifies the external label of an instruction whose address will be in register 14 when the function is entered for the first time. This parameter is required if the FCT is dispatchable. This is the address of the multifunction monitor (MFM).
R15=
  Specifies the external label of an instruction whose address will be in register 15 when the function is entered for the first time. This address is the module’s entry point.

ANAME=
  Specifies the 1- to 4-character alternate name by which the operator may communicate with the function.

SUCC=
  Specifies the label of the succeeding IATYFCD macro, which must have an equal or lower priority.

LOGIN=
  Specifies the label of the function’s console appendage if the function is to be permanently logged in.

INISH=
  YES
    Specifies that the FCT is always dispatchable.
  NO
    Specifies that the FCT is not dispatchable until JES3 initialization is complete.

PABLE=
  YES
    Specifies that the DSP may be scheduled by using the //*PROCESS job control statement.
  NO
    Specifies that the DSP may not be scheduled by using the //*PROCESS job control statement.

REENT=
  YES
    Specifies that the DSP is reenterable.
  NO
    Specifies that the DSP is not reenterable.

NOREQ=
  Specifies the maximum number of devices required by the DSP. The number of entries in the GETUNIT list associated with the function control table will be equal to this value. This parameter should be specified when known.

REQ=
  Specifies the types of devices known to be required by the DSP. The valid device types are specified on the DEVICE initialization statement. This parameter should be specified whenever the DSP unit requirements are known.
  If more than one device is required, the subparameter list must be enclosed in parentheses. For example, a DSP requiring two 9-track tape devices and a printer could specify NOREQ=3 and REQ=(TA9,TA9,PRT).

CSECT=
  For DSPs, this is the name of the data CSECT to be loaded for each use of the DSP.
GETUNV=
Specifies the required disposition of the function if devices are not available when the macro GETUNIT is issued.

C Indicates the function is to be canceled.
R Indicates the function is to return for specialized rescheduling.

The DSPRESCH bit in the DSP dictionary may be tested during DSP execution to determine which option is required. All callable DSPs in the distributed JES3 system also provide an operator override by using a keyword parameter to alter the basic disposition for an individual job.

MAXCT=
Specifies the maximum number of copies of this DSP that may be concurrently active. The number must be within the range 1 through 65534. If this parameter is not specified for a DSP specifying REENT=YES, no MAXCT limit is imposed. This parameter is dynamically alterable by using the *MODIFY command.

MUCC=
YES Allows the maximum use count specified by MAXCT to be altered by using the *MODIFY command.
NO Disallows maximum use count modification.

DRVR=
The name of the DSP driver module to be loaded, if necessary, for use of the DSP.

XABLE=
YES Specifies that the DSP can be called by using the *CALL command.
NO Specifies that the DSP cannot be called by using the *CALL command.

HABLE=
YES Allows the DSP’s work to be held and released by the *F,X,D=dsp,HOLD and *F,X,D=dsp,RELEASE commands. The command takes affect immediately upon being issued.

NO Disallows the DSP’s work to be held and released by the *F,X,D=dsp,HOLD and *F,X,D=dsp,RELEASE commands.

JOBVAL=
Allows you to specify the name of the job validation routine associated with the defined resident FCT.

PREV=
Specifies the label of the previous IATYFCD macro. This previous macro must have an equal or greater function priority (PRTY=) than the current invocation of the macro.

CRITICAL=
YES
Specifies that this resident FCT is critical to the function of JES3. If this FCT abnormally ends and cannot be recovered, the entire JES3 address space will automatically restart.

NO
Specifies that this resident FCT is not critical to the function of JES3. JES3 can continue to operate even if this FCT abnormally ends and cannot be recovered.

Example of Use
The following is an example of IATYFCD that identifies a validation routine named IATNFJV to JES3.

```
NEWFCT IATYFCD PRTY=45,
      NAME=NEWFUNC,
      SUCC=OLDFCT,
      PREV=PRIORFCT,
      ECFADD=NFCTECF,
      ECFMASK=FF,
      COND=50,
      JOBVAL=IATNFJV
```

IATYMOD (Identify Module)

The IATYMOD macro generates module identification information. Use of this macro is recommended, especially in installation exit routines.

When you use the IATYMOD macro in your routine, you must also include a field labelled APARNUM (and also a PTFNUM field) in your routine. These fields must be seven bytes each. They will be displayed on the operator’s console and in a dump header if the module fails in a JES3 environment. These fields help identify the level of maintenance in your code. If you use IATYMOD and omit APARNUM, your module will get an assembly error.

Syntax

```
/lSM590000/IATYMOD label
   ,BRANCH=YES, ,DSECT=YES
```

```
/lSM590000/IATYMOD label
   ,BRANCH=NO, ,DSECT=NO
```

```
/lSM630000/IATYMOD label
   ,LEVEL=name, ,ID=name
```

Parameters

BRANCH=
YES specifies that code generated by expansion of this macro will include a branch around the identification statements. NO specifies that no branch is to be included.

LEVEL=
Specifies the release, the feature, or the SU of the module.
ID=
   Specifies the name of the module.

DSECT=
   YES generates a DSECT for symbolic reference to the generated identification
   constants. If DSECT=YES, the LEVEL= and ID= parameters are ignored. Also,
   if DSECT=YES, BRANCH= must be NO.

Note: The IATYMOD macro expands inline. If you specify BRANCH=YES, the code
   expansion generates a branch around the identification statements. If you
   specify BRANCH=NO, no branch is generated. This could potentially cause
   an error if an attempt is made to execute the identification statements.
   Therefore, if you use BRANCH=NO, you should include a branch around the
   entire IATYMOD expansion before IATYMOD is issued.

If you omit all of the parameters on IATYMOD, the CSECT name is used for the
   ID= value, and the JES3 FMID is used for the LEVEL= value.

IATYRSC (JES3 Resource Table)

The IATYRSC macro specifies the names of critical JES3 resources for use with the
   ADEQ, AENQ, and ATEST macros.

Syntax

   /SM590000/SM590000
   IATYRSC
   label
   NAMES=(name),name
   LIST
   /SM590000

Parameters

NAMES=
   Generates a series of equate statements that equate each specified resource
   name with a unique decimal number. This form must be preceded by a previous
   expansion of the NAMES form. It generates a list of 8-byte constants of the
   previously defined names.

Note: A NAMES form expansion is included in the TVT.

IATYSRF (Map the SRF Header for a Spool Control Block)

The IATYSRF macro maps the header for any single record file control block written
   to the spool.

Syntax

   /SM590000/SM590000
   IATYSRF
   label
   PREFIX=name
   TYPE=value
   START=NO
   /SM590000
Parameters

PREFIX=
   Specifies the 3 or 4-character identification field of the single record file (SRF). These characters are appended to the start of each label in the SRF header. The module creating the spool control block must move the actual characters into the identification field in the header before writing the record to the spool (using the AWRITE macro).

TYPE=
   Specifies the type of data area to be generated. Valid types (as shown by value) are DSECT, CSECT, X, H, F, D. If a type specification is omitted, DSECT is assumed.

START=
   YES
      Specifies that the label generated on the TYPE= statement (for example, DSECT, DS 0H.) will end in the characters “STRT” or “START”.
   NO
      Specifies that the label generated at the start of the header consists only of the characters specified on the PREFIX= parameter.

PREV=
   YES
      Generates a file description block (FDB) in the SRF header to be used as a backward file pointer.
   NO
      Does not generate an FDB for the previous record in the file.

To use the IATYSRF macro, you should code your spool control block as follows (this is a sample only):

<table>
<thead>
<tr>
<th>Sample</th>
</tr>
</thead>
<tbody>
<tr>
<td>MACRO</td>
</tr>
<tr>
<td>IATYXYZ (macro or control block name)</td>
</tr>
<tr>
<td>*** prolog for macro goes here ***</td>
</tr>
<tr>
<td>IATYSRF PREFIX=XYZ</td>
</tr>
<tr>
<td>*** any control block-specific fields go here ***</td>
</tr>
<tr>
<td>MEND</td>
</tr>
</tbody>
</table>

The following two examples show how the macro generates the code.
Example 1

```
IATYSRF  PREFIX=XYZ

YZSTART  DSECT
YZTRK    DC    XL(L'FDBSPADR) '0'
YZCNT    DC    H'0'
YZID     DC    CL4'XYZ'
YZCHN    DC    XL(FDBSRFL) '0'
        DC    F'0'
YZDATA   DS    0F
```

Note that you will have to use an MVC instruction to move the characters 'XYZ' into the XYZID field: for example,

```
MVC  XYZID, = CL4'XYZ'
```

Example 2

```
IATYSRF  PREFIX=XYZA, TYPE=H, PREV=YES, START=NO

YZA     DS    0H
YZATRK  .
        .
YZAPREV DC    XL(FDBSRFL) '0'
        DC    F'0'
YZADATA DS    0F
```

Linkage

This macro does not generate executable code.

Restrictions

None.
Appendix A. Information for JES3 Security Calls

JES3 passes information to the security authorization facility (SAF) by using a RACROUTE request to communicate with the Resource Access Control Facility (RACF). RACF ensures that a user or job has the proper authority to access a JES3-managed resource. JES3 calls installation exit IATUX58 before calling SAF to allow the installation to:

- Respecify the type of security that JES3 requests from RACF.
- Allow RACF to make all the security decisions.
- Perform processing in installation exit IATUX58, call SAF, call installation exit IATUX59 to examine the decision, and then return to JES3.
- Perform processing in installation exit IATUX58, call SAF to perform security checking and then return to JES3.
- Accept or reject the request without requesting the services of RACF to make a decision.

JES3 passes installation exit IATUX58 a parameter list (IATYSSX) which identifies the processing JES3 was performing when the request for security was issued. You should use the call indicator (SSXINDEX in IATYSSX) to determine the type of request JES3 is making to RACF.

JES3 issues 7 types of security calls to SAF. You can determine the type of security call by examining the fifth character of the call indicator. The fourth and fifth characters of the call indicator are called the request indicator. The request indicator indicates the type of security call as one of the following:

### Request Indicator

<table>
<thead>
<tr>
<th>Type of Security Call</th>
<th>Request Indicator</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authority request. JES3 requests RACF to ensure the job or user is authorized to use a resource.</td>
<td>IA</td>
</tr>
<tr>
<td>Extract request. JES3 requests SAF to encrypt a specified password or to extract a password from the user's security token.</td>
<td>IE</td>
</tr>
<tr>
<td>Token build request. JES3 requests SAF to update a security token.</td>
<td>IB</td>
</tr>
<tr>
<td>Token map request. JES3 requests SAF to convert a token to or from either internal or external form.</td>
<td>IM</td>
</tr>
<tr>
<td>Token extract request. JES3 requests SAF to return the token for the calling function.</td>
<td>IX</td>
</tr>
<tr>
<td>Create an ACEE during converter/interpreter processing to authorize the use of a data set specified in a JCLLIB statement.</td>
<td>VF</td>
</tr>
<tr>
<td>Verification request. JES3 requests RACF to verify that a job has the proper authority to run in the system.</td>
<td>IV</td>
</tr>
<tr>
<td>Audit request. JES3 requests SAF to verify the deletion of a job during initialization or as a result of an operator CANCEL command.</td>
<td>AU</td>
</tr>
</tbody>
</table>

Information is passed to installation exit IATUX58 based on the type of request. For each type of request, Table 21 on page 608 identifies the information common to that request that installation exit IATUX58 can examine and modify.

For information specific to a unique IATXSEC invocation, Table 22 on page 610 identifies the fields that installation exit IATUX58 can examine or modify.
<table>
<thead>
<tr>
<th>Request Indicator</th>
<th>Information IATUX58 Can Examine</th>
<th>Information IATUX58 Can Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>IA</td>
<td>Field Name</td>
<td>• SSXATTR</td>
</tr>
<tr>
<td></td>
<td>Description</td>
<td>• SSXENTIT</td>
</tr>
<tr>
<td></td>
<td>SSXINDEX</td>
<td>• SSXLGSTR</td>
</tr>
<tr>
<td></td>
<td>Use it to determine the processing JES3 was performing before issuing the security call.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SSXMODE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Specifies the phase of JES3 processing that the caller of IATXSEC is in.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SSXCLASS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Name of the RACF class for the RACROUTE macro.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SSXATTR</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Security authority needed.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SSXENTIT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Entity name.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SSXLGSTR</td>
<td></td>
</tr>
<tr>
<td></td>
<td>String of characters that will be added to the SMF security record.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SSXMCNTL</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Specifies where messages are to be displayed.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SSXRTOKN</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Resource security token.</td>
<td></td>
</tr>
<tr>
<td>IE</td>
<td>Field Name</td>
<td>• SSXENTIT</td>
</tr>
<tr>
<td></td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SSXINDEX</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Use it to determine the processing JES3 was performing before issuing the security call.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SSXMODE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Specifies the phase of JES3 processing that the caller of IATXSEC is in.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SSXCLASS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Name of the RACF class for the RACROUTE macro.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SSXENTIT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Entity name.</td>
<td></td>
</tr>
<tr>
<td>IB</td>
<td>Field Name</td>
<td>• SSXTOKIN</td>
</tr>
<tr>
<td></td>
<td>Description</td>
<td>• SSXTOKOT</td>
</tr>
<tr>
<td></td>
<td>SSXINDEX</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Use it to determine the processing JES3 was performing before issuing the security call.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SSXMODE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Specifies the phase of JES3 processing that the caller of IATXSEC is in.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SSXTOKIN</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Specifies the input token.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SSXTOKOT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Specifies the output token.</td>
<td></td>
</tr>
</tbody>
</table>
Table 21. Common Information Supplied for Security Calls to SAF (continued)

<table>
<thead>
<tr>
<th>Request Indicator</th>
<th>Information IATUX58 Can Examine</th>
<th>Information IATUX58 Can Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>IM</td>
<td><strong>Field Name</strong></td>
<td>• SSXTOKIN</td>
</tr>
<tr>
<td></td>
<td><strong>Description</strong></td>
<td>• SSXTOKOT</td>
</tr>
<tr>
<td></td>
<td><strong>SSXINDEX</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Use it to determine the processing JES3 was performing before issuing the security call.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>SSXMODE</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Specifies the phase of JES3 processing that the caller of IATXSEC is in.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>SSXTOKIN</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Specifies the input token.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>SSXTOKOT</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Specifies the output token.</td>
<td></td>
</tr>
<tr>
<td>VF</td>
<td><strong>Field Name</strong></td>
<td>• SXXLGSTR</td>
</tr>
<tr>
<td></td>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>SSXINDEX</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Use it to determine the processing JES3 was performing before issuing the security call</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>SSXMODE</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Specifies the phase of JES3 processing that the caller of IATXSEC is in.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>SSXLGSTR</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A string of characters that will be added to the SMF security record.</td>
<td></td>
</tr>
<tr>
<td>IX</td>
<td><strong>Field Name</strong></td>
<td>• SSXTOKOT</td>
</tr>
<tr>
<td></td>
<td><strong>Description</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>SSXINDEX</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Use it to determine the processing JES3 was performing before issuing the security call.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>SSXMODE</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Specifies the phase of JES3 processing that the caller of IATXSEC is in.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>SSXTOKOT</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Specifies the output token.</td>
<td></td>
</tr>
</tbody>
</table>
Table 21. Common Information Supplied for Security Calls to SAF (continued)

<table>
<thead>
<tr>
<th>Request Indicator</th>
<th>Information IATUX58 Can Examine</th>
<th>Information IATUX58 Can Change</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>IV</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Field Name</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>SSXINDEX</td>
<td>Use it to determine the processing JES3 was performing before issuing the security call.</td>
<td></td>
</tr>
<tr>
<td>SSXMODE</td>
<td>Specifies the phase of JES3 processing that the caller of IATXSEC is in.</td>
<td></td>
</tr>
<tr>
<td>SSXLGSTR</td>
<td>A string of characters that will be added to the SMF security record.</td>
<td></td>
</tr>
<tr>
<td>SSXSESSION</td>
<td>Specifies the session type.</td>
<td></td>
</tr>
<tr>
<td>SSXTOKIN</td>
<td>Specifies the input token.</td>
<td></td>
</tr>
<tr>
<td>SSXTOKOT</td>
<td>Specifies the output token.</td>
<td></td>
</tr>
<tr>
<td>AU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Field Name</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>SSXINDEX</td>
<td>Use it to determine the processing JES3 was performing before issuing the security call.</td>
<td></td>
</tr>
<tr>
<td>SSXMODE</td>
<td>Specifies the phase of JES3 processing that the caller of IATXSEC is in.</td>
<td></td>
</tr>
<tr>
<td>SSXLGSTR</td>
<td>String of characters that will be added to the SMF security record.</td>
<td></td>
</tr>
</tbody>
</table>

Table 22. Support Information for IATUX58

<table>
<thead>
<tr>
<th>SSXINDEX</th>
<th>JES3 Calling Module</th>
<th>Security Requested by JES3</th>
<th>Additional Information IATUX58 Can Examine</th>
<th>Additional Information IATUX58 Can Modify</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSXIANOC</td>
<td>IATISNJ</td>
<td>Authorization to create a JCL/SYSIN data set before transmitting it to another node for jobs submitted with // XMIT or /*ROUTE XEQ statement.</td>
<td>• SSXUTOKN – Specifies the user's security token  • SSXNJDEJ – Address of the NJE job header</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• SSXUTOKN</td>
</tr>
<tr>
<td>SSXIANJC</td>
<td>IATISNJ</td>
<td>Create NJE job JESMSGLG data set.</td>
<td>• SSXUTOKN – Specifies the user's security token</td>
<td>• SSXUTOKN</td>
</tr>
<tr>
<td>SSXIANJO</td>
<td>IATISNJ</td>
<td>Open NJE job JESMSGLG data set.</td>
<td>• SSXUTOKN – Specifies the user's security token</td>
<td>• SSXUTOKN</td>
</tr>
<tr>
<td>SSXIANRC</td>
<td>IATNTRS</td>
<td>Authorization to create an NJE SYSSOUT data set to be rerouted to the home node.</td>
<td>• SSXUTOKN – Specifies the user's security token  • SSXNJDEJ – Address of the data set header for the NJE job</td>
<td>• SSXUTOKN</td>
</tr>
<tr>
<td>SSXINDEX</td>
<td>JES3 Calling Module</td>
<td>Security Requested by JES3</td>
<td>Additional Information IATUX58 Can Examine</td>
<td>Additional Information IATUX58 Can Modify</td>
</tr>
<tr>
<td>-----------</td>
<td>---------------------</td>
<td>-----------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------</td>
<td>-------------------------------------------</td>
</tr>
<tr>
<td>SSXIANRR</td>
<td>IATNTRS</td>
<td>Authorization to create an NJE SYSOUT or SYSIN data set to be rerouted to another node.</td>
<td>• SSXUTOKN – Specifies the user’s security token</td>
<td>• SSXUTOKN</td>
</tr>
<tr>
<td>SSXIANOS</td>
<td>IATNTSD</td>
<td>Simulate a writer access SAF call using the JESSPOOL resource class.</td>
<td>• SSXUTOKN – Specifies the user’s security token</td>
<td>• SSXUTOKN</td>
</tr>
<tr>
<td>SSXIANOW</td>
<td>IATNTSD</td>
<td>Authorization for a writer (BSC NJE sender) to print an outbound NJE data set and for the user to access the writer.</td>
<td>• SSXUTOKN – Specifies the user’s security token</td>
<td>• SSXUTOKN</td>
</tr>
<tr>
<td>SSXIANIC</td>
<td>IATNTSF</td>
<td>Authorization to create a data set received for local processing from another node.</td>
<td>• SSXUTOKN – Specifies the user’s security token</td>
<td>• SSXUTOKN</td>
</tr>
<tr>
<td>SSXIANSC</td>
<td>IATNTSF IATOSNT</td>
<td>Create store and forward NJE data set.</td>
<td>• SSXUTOKN – Specifies the user’s security token</td>
<td>• SSXUTOKN</td>
</tr>
<tr>
<td>SSXIAISC</td>
<td>IATISEN</td>
<td>When creating job-related data sets (JESMSGGLG, JESJCLIN, JESYMSG, JESJCL), JES3 checks to see that the user is authorized.</td>
<td>• SSXUTOKN – Specifies the user’s security token</td>
<td>• SSXUTOKN</td>
</tr>
<tr>
<td>SSXIAISO</td>
<td>IATISEN</td>
<td>When opening job-related data sets (JESMSGGLG, JESJCLIN, JESYMSG, JESJCL), JES3 checks to see that the user is authorized.</td>
<td>• SSXUTOKN – Specifies the user’s security token</td>
<td>• SSXUTOKN</td>
</tr>
<tr>
<td>SSXIAISD</td>
<td>IATISEN</td>
<td>If a PROCESS statement is encountered, determine that job has authorization to invoke this DSP.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSXIASIA</td>
<td>IATSIAD</td>
<td>Determine if a user is authorized to create a SYSIN data set.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSXIASOC</td>
<td>IATSIOR</td>
<td>Determine if a user is authorized to open a SYSIN/PSO data set.</td>
<td>• SSXRECVR – Specifies the receiver name (if receive by userid)</td>
<td>• SSXRECVR</td>
</tr>
<tr>
<td>SSXIASIO</td>
<td>IATSIOR</td>
<td>Determine if a user is authorized to open a SYSIN/PSO data set.</td>
<td>• SSXRECVR – Specifies the receiver name (if receive by userid)</td>
<td>• SSXRECVR</td>
</tr>
<tr>
<td>SSXINDEX</td>
<td>JES3 Calling Module</td>
<td>Security Requested by JES3</td>
<td>Additional Information IATUX58 Can Examine</td>
<td>Additional Information IATUX58 Can Modify</td>
</tr>
<tr>
<td>----------</td>
<td>---------------------</td>
<td>-----------------------------</td>
<td>--------------------------------------------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>SSXIASOO</td>
<td>IATSIOR</td>
<td>Determine if a user is authorized to open a SYSOUT data set.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSXIASIR</td>
<td>IATSIOR</td>
<td>Determine if a user is authorized to open the internal reader.</td>
<td>• SSXUTOKN – Specifies the user's security token</td>
<td>• SSXUTOKN</td>
</tr>
<tr>
<td>SSXIADMJ</td>
<td>IATDMJA</td>
<td>Authorization to delete a data set during auto step restart processing.</td>
<td>• SSXUTOKN – Specifies the user's security token</td>
<td>• SSXUTOKN</td>
</tr>
<tr>
<td>SSXIADMA</td>
<td>IATDMJA</td>
<td>Authorization to delete a data set.</td>
<td>• SSXUTOKN – Specifies the user's security token</td>
<td>• SSXUTOKN</td>
</tr>
<tr>
<td>SSXIAMS</td>
<td>IATMSMS</td>
<td>Authorization to delete previous SYSIN and SYSOUT data sets when restarting a job from the beginning.</td>
<td>• SSXUTOKN – Specifies the user's security token</td>
<td>• SSXUTOKN</td>
</tr>
<tr>
<td>SSXIADOS</td>
<td>IATOSDR</td>
<td>Determine if a user is authorized to delete SYSOUT data sets.</td>
<td>• SSXUTOKN – Specifies the user's security token</td>
<td>• SSXUTOKN</td>
</tr>
<tr>
<td>SSXIASP</td>
<td>IATOSFP</td>
<td>Authorization to delete a spin-off data set currently being processed by an FSS-supported printer.</td>
<td>• SSXUTOKN – Specifies the user's security token</td>
<td>• SSXUTOKN</td>
</tr>
<tr>
<td>SSXIAGC</td>
<td>IATOSPC</td>
<td>Authorization to purge a data set because of security authorization denial while trying to retrieve a TSO data set.</td>
<td>• SSXUTOKN – Specifies the user's security token</td>
<td>• SSXUTOKN</td>
</tr>
<tr>
<td>SSXIOSC</td>
<td>IATOSPC</td>
<td>Authorization to delete a spinoff data set during the delete phase of process SYSOUT processing.</td>
<td>• SSXUTOKN – Specifies the user's security token</td>
<td>• SSXUTOKN</td>
</tr>
<tr>
<td>SSXIOSS</td>
<td>IATOSSC</td>
<td>Authorization to purge a spin-off SYSOUT data set.</td>
<td>• SSXUTOKN – Specifies the user's security token</td>
<td>• SSXUTOKN</td>
</tr>
<tr>
<td>SSXIOW</td>
<td>IATOSWP</td>
<td>Authorization to delete a spin-off data set's TAT entry.</td>
<td>• SSXUTOKN – Specifies the user's security token</td>
<td>• SSXUTOKN</td>
</tr>
<tr>
<td>SSXIAPUR</td>
<td>IATPURG</td>
<td>Authorization to delete SYSOUT data sets.</td>
<td>• SSXUTOKN – Specifies the user's security token</td>
<td>• SSXUTOKN</td>
</tr>
<tr>
<td>SSXIAGR</td>
<td>IATGRAN</td>
<td>Checks to see if JESNEWS that was started with a /*PROCESS statement can be updated.</td>
<td>• SSXUTOKN – Specifies the user's security token</td>
<td>• SSXUTOKN</td>
</tr>
<tr>
<td>SSXIAOGR</td>
<td>IATGRAN</td>
<td>Checks to see if JESNEWS that was started by an operator command can be updated.</td>
<td>• SSXUTOKN – Specifies the user's security token</td>
<td>• SSXUTOKN</td>
</tr>
<tr>
<td>SSXINDEX</td>
<td>JES3 Calling Module</td>
<td>Security Requested by JES3</td>
<td>Additional Information IATUX58 Can Examine</td>
<td>Additional Information IATUX58 Can Modify</td>
</tr>
<tr>
<td>----------</td>
<td>---------------------</td>
<td>-----------------------------</td>
<td>---------------------------------------------</td>
<td>------------------------------------------</td>
</tr>
<tr>
<td>SSXIAOSR</td>
<td>IATOSDR</td>
<td>Audit the creation of a data set for JES3 job0 processing.</td>
<td>• SSXUTOKN – Job’s user security token</td>
<td>• SSXUTOKN</td>
</tr>
</tbody>
</table>
Table 22. Support Information for IATUX58 (continued)

<table>
<thead>
<tr>
<th>SSXINDEX</th>
<th>JES3 Calling Module</th>
<th>Security Requested by JES3</th>
<th>Additional Information IATUX58 Can Examine</th>
<th>Additional Information IATUX58 Can Modify</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSXIAWD1</td>
<td>IATOSWD</td>
<td>JESSPOOL call for data set authorization for JES3 writers</td>
<td>• SSXUTOKN – User security token</td>
<td>• SSXUTOKN</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• SSXWPSLC – Specifies the number of selection characteristics that are specified (length of SSXWPSLM)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• SSXWJNAM – Job name</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• SSXWJBID – Job identifier</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• SSXDDSN – Writer DDNAME</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• SSXWPSLM – WSP selection mask</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• SSXOPRTY – Priority of the job’s output</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• SSXODEST – Destination of the job’s output</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• SSXOMID – Specifies the output’s copy mod identifier</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• SSXOSTCK – Specifies whether a stacker is required</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• SSXTYPE – Device type required</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• SSXOFRMS – Type of form required</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• SSXOFLSH – Specifies the flash required</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• SSXOUCS – UCS identifier</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• SSXOCLSS – SYSOUT class</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• SSXOMODE – Process mode</td>
<td></td>
</tr>
<tr>
<td>SSXIAOS3</td>
<td>IATOSBM</td>
<td>Checks JESSPOOL class for SNA/NJE writer.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSXINDEX</td>
<td>JES3 Calling Module</td>
<td>Security Requested by JES3</td>
<td>Additional Information IATUX58 Can Examine</td>
<td>Additional Information IATUX58 Can Modify</td>
</tr>
<tr>
<td>----------</td>
<td>---------------------</td>
<td>-----------------------------</td>
<td>---------------------------------------------</td>
<td>---------------------------------------------</td>
</tr>
</tbody>
</table>
| SSXIAFG1 | IATOSFG             | JESSPOOL call for SYSOUT data set authorization for FSS writers. | • SSXUTOKN – User security token  
• SSXWPSLC – Specifies the number of selection characteristics that are specified (length of SSXWPSLM)  
• SSXWJNAM – Job name  
• SSXWJBID – Job identifier  
• SSXDDSN – Writer DDNAME  
• SSXWPSLM – WSP selection mask  
• SSXOPRTY – Priority of the job’s output  
• SSXODEST – Destination of the job’s output  
• SSXOMDID – Specifies the output’s copy mod identifier  
• SSXOSTCK – Specifies whether a stacker is required  
• SSXOTYPE – Device type required  
• SSXOFRMS – Type of form required  
• SSXOFLSH – Specifies the flash required  
• SSXOUCS – UCS identifier  
• SSXOCLSS – SYSOUT class  
• SSXOMODE – Process mode | • SSXUTOKN |
<table>
<thead>
<tr>
<th>SSXINDEX</th>
<th>JES3 Calling Module</th>
<th>Security Requested by JES3</th>
<th>Additional Information IATUX58 Can Examine</th>
<th>Additional Information IATUX58 Can Modify</th>
</tr>
</thead>
</table>
| SSXIAUG1 | IATOSPC             | Authorization for a PSO GET (by userid). | • SSXUTOKN – Job’s user security token  
• SSXRECVR – Specifies the receiver name  
• SSXWJNAM – Job name  
• SSXWJBID – Job identifier  
• SSXODEST – Destination of the job’s output  
• SSXOTYPE – Device type required  
• SSXOFRMS – Type of form required  
• SSXOCLSS – SYSOUT class  
• SSXOSSO – Address of IEFSSSO  
• SSXOFLAG – Specifies information about the job’s output, destination, and data set  
• SSXOWRTN – External writer name | • SSXRECVR  
• SSXUTOKN |
| SSXIAUG2 | IATOSPC             | Authorization for PSO PUT (by userid) | • SSXUTOKN – Job’s user security token  
• SSXRECVR – Specifies the receiver name  
• SSXWJNAM – Job name  
• SSXWJBID – Job identifier  
• SSXODEST – Destination of the job’s output  
• SSXOTYPE – Device type required  
• SSXOFRMS – Type of form required  
• SSXOCLSS – SYSOUT class  
• SSXOSSO – Address of IEFSSSO  
• SSXOFLAG – Specifies information about the job’s output, destination, and data set  
• SSXOWRTN – External writer name | • SSXRECVR  
• SSXUTOKN |
<table>
<thead>
<tr>
<th>SSXINDEX</th>
<th>JES3 Calling Module</th>
<th>Security Requested by JES3</th>
<th>Additional Information IATUX58 Can Examine</th>
<th>Additional Information IATUX58 Can Modify</th>
</tr>
</thead>
</table>
| SSXIAREA   | IATOSPC             | Authorization for a PSO GET (not by userid). | • SSXUTOKN – Job's user security token  
• SSXRECVR – Specifies the receiver name  
• SSXWJNAM – Job name  
• SSXWJBID – Job identifier  
• SSXODEST – Destination of the job’s output  
• SSXOTYPE – Device type required  
• SSXOFRMS – Type of form required  
• SSXOCLSS – SYSOUT class  
• SSXOSSO – Address of IEFSSSO  
• SSXOFLAG – Specifies information about the job’s output, destination, and data set  
• SSXOWRTN – External writer name | • SSXRECVR  
• SSXUTOKN |
| SSXIAALT   | IATOSPC             | Authorization for a PSO PUT (not by userid). | • SSXUTOKN – Job’s user security token  
• SSXRECVR – Specifies the receiver name  
• SSXWJNAM – Job name  
• SSXWJBID – Job identifier  
• SSXODEST – Destination of the job’s output  
• SSXOTYPE – Device type required  
• SSXOFRMS – Type of form required  
• SSXOCLSS – SYSOUT class  
• SSXOSSO – Address of IEFSSSO  
• SSXOFLAG – Specifies information about the job’s output, destination, and data set  
• SSXOWRTN – External writer name | • SSXRECVR  
• SSXUTOKN |
<table>
<thead>
<tr>
<th>SSXINDEX</th>
<th>JES3 Calling Module</th>
<th>Security Requested by JES3</th>
<th>Additional Information IATUX58 Can Examine</th>
<th>Additional Information IATUX58 Can Modify</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSXIACMD</td>
<td>IATCNI A</td>
<td>Authorization for commands.</td>
<td>• SSXUTOKN – Operator’s security token</td>
<td>• SSXUTOKN</td>
</tr>
<tr>
<td>SSXIADJ1</td>
<td>IATDJIN</td>
<td>Authorization to create a data set for a job being restored by the dump job facility.</td>
<td>• SSXUTOKN – Job’s user security token</td>
<td>• SSXUTOKN</td>
</tr>
<tr>
<td>SSXIACGP</td>
<td>IATGRPR</td>
<td>Determine if a user is authorized to create CBPRNT and DEBUG data set.</td>
<td>• SSXUTOKN – Specifies the user’s security token</td>
<td>• SSXUTOKN</td>
</tr>
<tr>
<td>SSXIAOGP</td>
<td>IATGRPR</td>
<td>Determine if a user is authorized to open CBPRNT and DEBUG data set.</td>
<td>• SSXUTOKN – Specifies the user’s security token</td>
<td>• SSXUTOKN</td>
</tr>
<tr>
<td>SSXIAGRW</td>
<td>IATGRWQ</td>
<td>Authorization for the user to cancel the object specified by ENTITY=.</td>
<td>• SSXUTOKN – User’s security token (passed from IATSICN)</td>
<td>• SSXUTOKN</td>
</tr>
<tr>
<td>SSXIADJ2</td>
<td>IATDJIN</td>
<td>Authorization to purge a multi-record file.</td>
<td>• SSXUTOKN – Specifies the user’s security token</td>
<td>• SSXUTOKN</td>
</tr>
<tr>
<td>SSXIANUM</td>
<td>IATSINU</td>
<td>Notify user node authority.</td>
<td>• SSXUTOKN – Specifies the user’s security token  • SSXLOG – Specifies the types of access attempts to be recorded on the SMF data set</td>
<td>• SSXUTOKN  • SSXLOG</td>
</tr>
<tr>
<td>SSXIASRO</td>
<td>IATSIOR</td>
<td>Internal reader REOPEN.</td>
<td>• SSXRECVR – Specifies the receiver name (if receive by userid)</td>
<td>• SSXRECVR</td>
</tr>
<tr>
<td>SSXIEDJN</td>
<td>IATDJIN</td>
<td>Call to encrypt the current or new password for a job being restored by the dump job facility.</td>
<td>• Register 3 – Password to be encrypted  • SSXMCNTL – Specifies where messages are to be displayed</td>
<td>• SSXMCNTL  • SSXLOG</td>
</tr>
<tr>
<td>SSXIEISJ</td>
<td>IATISJB</td>
<td>Call to encrypt the current or new password on the job card.</td>
<td>• Register 6 – Password to be encrypted</td>
<td></td>
</tr>
<tr>
<td>SSXIEPE</td>
<td>IATISNJ</td>
<td>Encrypt password(s) specified on the JOB statement before placing them in the outbound job header for jobs with // XMIT or //&quot;ROUTE XEQ statements.</td>
<td>• SSXMCNTL – Specifies where messages are to be displayed  • SSXNEJH – Address of the NJE job header  • SSXENCRY – Address of data to be encrypted  • SSXENCME – Specifies the encryption method</td>
<td>• SSXMCNTL  • SSXLOG</td>
</tr>
<tr>
<td>SSXINDEX</td>
<td>JES3 Calling Module</td>
<td>Security Requested by JES3</td>
<td>Additional Information IATUX58 Can Examine</td>
<td>Additional Information IATUX58 Can Modify</td>
</tr>
<tr>
<td>----------</td>
<td>---------------------</td>
<td>---------------------------</td>
<td>---------------------------------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td>SSXIBNIS</td>
<td>IATNTJS</td>
<td>Update token for inbound NJE SYSOUT stream.</td>
<td>• SSXMCNTL – Specifies where messages are to be displayed • SSXEXND – Address of execution node • SSXNJEJH – Address of the NJE job header</td>
<td>• SSXEXND</td>
</tr>
<tr>
<td>SSXIBNRS</td>
<td>IATNTRS</td>
<td>Update token for rerouted NJE SYSOUT stream.</td>
<td>• SSXEXNOD – Specifies the address of the execution node</td>
<td>• SSXEXNOD</td>
</tr>
<tr>
<td>SSXIBISJ</td>
<td>IATISEN</td>
<td>Update point of entry for all demand select jobs except TSO logons; update node.</td>
<td>• SSXPOE – Name of the entry point</td>
<td>• SSXPOE</td>
</tr>
<tr>
<td>SSXIBISI</td>
<td>IATISRI</td>
<td>Updates a reader token with the SESSION/PO of the operator who called the reader.</td>
<td>• SSXPOE – Name of the entry point</td>
<td></td>
</tr>
<tr>
<td>SSXIBSRL</td>
<td>IATISRL</td>
<td>Update reader token with SESSION/POE (operator who started reader).</td>
<td>• SSXPOE – Name of the entry point • SSXSSION – Specifies the session type.</td>
<td>• SSXPOE • SSXSSION</td>
</tr>
<tr>
<td>SSXIBINI</td>
<td>IATINGN</td>
<td>Build a syslow token for JESNEWS</td>
<td>• SSXLGSTR – String of characters that will be added to the SMF security record • SSXSECLB – Security label</td>
<td>• SSXLGSTR • SSXSECLB</td>
</tr>
<tr>
<td>SSXIBING</td>
<td>IATINGN</td>
<td>Updates a JES3 token with EXENODE.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSXIBSAD</td>
<td>IATSIAD</td>
<td>Update APPC transaction data set token with EXENODE.</td>
<td>• SSXEXNOD – Specifies the address of the execution node</td>
<td>• SSXEXNOD</td>
</tr>
<tr>
<td>SSXIMNJJ</td>
<td>IATISNJ</td>
<td>Convert a job’s SSXUTOKN from internal to external format so that it can be placed in the outbound NJE job header for jobs with // XMIT or //ROUTE XEQ statements.</td>
<td>• SSXFRMOT – Specifies the format of the output token • SSXNJEJH – Address of the NJE job header • SSXMCNTL – Specifies where messages are to be displayed</td>
<td></td>
</tr>
<tr>
<td>SSXINDEX</td>
<td>JES3 Calling Module</td>
<td>Security Requested by JES3</td>
<td>Additional Information IATUX58 Can Examine</td>
<td>Additional Information IATUX58 Can Modify</td>
</tr>
<tr>
<td>----------</td>
<td>---------------------</td>
<td>---------------------------</td>
<td>--------------------------------------------</td>
<td>------------------------------------------</td>
</tr>
</tbody>
</table>
| SSXIMNSD | IATNTDH             | Convert a job's SSXUTOKN from internal to external format so that it can be placed in the outbound NJE data set header for SYSOUT being sent to another node. | • SSXFRMOT – Specifies the format of the output token  
• SSXMCNTL – Specifies where messages are to be displayed  
• SSXNJEDH – Address of the data set header | |
| SSXIMNSJ | IATNTHT             | Convert a job's SSXUTOKN from internal to external format so that it can be placed in the outbound NJE job header for SYSOUT being sent to another node. | • SSXFRMOT – Specifies the format of the output token  
• SSXMCNTL – Specifies where messages are to be displayed  
• SSXNJEDH – Address of the data set header | |
| SSXIMNIJ | IATNTJS             | Convert token received from another node from external format to internal format to use as input to SAF | • SSXMCNTL – Specifies where messages are to be displayed  
• SSXFRMOT – Specifies the format of the output token | |
| SSXIMNIS | IATNTJS             | Convert token from internal to external format to access information in the token for SYSOUT streams received from the NJE network. | • SSXFRMOT – Specifies the format of the output token  
• SSXMCNTL – Specifies where messages are to be displayed | |
| SSXIMNST | IATNTJS             | Map an inbound NJE SYSOUT stream token. | • SSXFRMOT – Specifies the format of the output token  
• SSXMCNTL – Specifies where messages are to be displayed | |
| SSXIMNJU | IATNTJS             | Map an NJE job stream UNKNOWN USER token. | • SSXFRMOT – Specifies the format of the output token  
• SSXMCNTL – Specifies where messages are to be displayed | |
<table>
<thead>
<tr>
<th>SSXINDEX</th>
<th>JES3 Calling Module</th>
<th>Security Requested by JES3</th>
<th>Additional Information IATUX58 Can Examine</th>
<th>Additional Information IATUX58 Can Modify</th>
</tr>
</thead>
</table>
| SSXIMNSX | IATNTSF             | Map a token returned by IATUX67. | • SSXFRMOT – Specifies the format of the output token  
• SSXMCNTL – Specifies where messages are to be displayed  
• SSXNJEJH – Address of the data set header for the NJE job | |
| SSXIMNRS | IATNTRS             | Convert a job stream's TOKEN for SYSOUT being rerouted to the home node from external format to internal format to be used as input to SAF. | • SSXFRMOT – Specifies the format of the output token  
• SSXMCNTL – Specifies where messages are to be displayed  
• SSXNJEJH – Address of NJE job header | |
| SSXIMNRJ | IATNTRS             | Convert a rerouted NJE job stream JH token to external format. | • SSXFRMOT – Specifies the format of the output token  
• SSXMCNTL – Specifies where messages are to be displayed  
• SSXNJEJH – Address of the job header for the NJE job | |
| SSXIMNRT | IATNTRS             | Map a rerouted NJE SYSOUT stream token. | • SSXFRMOT – Specifies the format of the output token  
• SSXMCNTL – Specifies where messages are to be displayed  
• SSXNJEJH – Address of the job header for the NJE job | |
| SSXIMNRX | IATNTRS             | Map a token returned by IATUX67. | • SSXFRMOT – Specifies the format of the output token  
• SSXMCNTL – Specifies where messages are to be displayed  
• SSXNJEJH – Address of the NJE job header | |
<p>| SSXIMNRB | IATNTRS             | Map rerouted NJE SYSIN or SYSOUT job-level token to external form. | • SSXNJEJH – Address of the NJE job header | • SSXNJEJH |</p>
<table>
<thead>
<tr>
<th>SSXINDEX</th>
<th>JES3 Calling Module</th>
<th>Security Requested by JES3</th>
<th>Additional Information IATUX58 Can Examine</th>
<th>Additional Information IATUX58 Can Modify</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSXIMNSE</td>
<td>IATOSBP</td>
<td>Convert inbound NJE SYSOUT stream DSH token to internal format so that it can be used to create a new data set header.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSXIMISJ</td>
<td>IATISEN</td>
<td>Map returned token for jobs to obtain node, userid, and security label for inclusion in JES3 control blocks.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSXIMOSD</td>
<td>IATOSDR</td>
<td>Obtain JES3’s token map for spin-off processing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSXIMINI</td>
<td>IATINGN</td>
<td>Translate security token to obtain node and userid used to build the data set name for JESNEWS.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSXIMDM1</td>
<td>IATDMJA</td>
<td>Obtain information from user token for PSO for unallocation, if received by userid.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSXIMOSO</td>
<td>IATOSDO</td>
<td>To place the security label for the token into the MOSE.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSXIMDJ1</td>
<td>IATDJIN</td>
<td>Map the token created for a job restored by the dump job facility from an earlier release of JES3.</td>
<td></td>
<td>SSXMCNTL – Specifies where messages are to be displayed</td>
</tr>
<tr>
<td>SSXIMISE</td>
<td>IATISEN</td>
<td>Map submitter’s token for NJE jobs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSXIMOSN</td>
<td>IATOSNT</td>
<td>Map job token during NJE packaging for a destination</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSXIMOS2</td>
<td>IATOSNT</td>
<td>Map job token during NJE packaging for a destination.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSXIMAD1</td>
<td>IATSIAD</td>
<td>Map token during APPC SYSOUT allocation.</td>
<td>SSXMCNTL – Specifies where messages are to be displayed</td>
<td></td>
</tr>
<tr>
<td>SSXIMCD1</td>
<td>IATGRCD</td>
<td>Obtain information from user token for callable DSP.</td>
<td>SSXMCNTL – Specifies where messages are to be displayed</td>
<td></td>
</tr>
<tr>
<td>SSXIMSTP</td>
<td>IATIIST</td>
<td>Map job token during setup of C/I security environment</td>
<td>SSXMCNTL – Specifies where messages are to be displayed</td>
<td></td>
</tr>
<tr>
<td>SSXIMAD2</td>
<td>IATSIAD</td>
<td>Map job token during STC/TSO allocation</td>
<td>SSXMCNTL – Specifies where messages are to be displayed</td>
<td></td>
</tr>
</tbody>
</table>
Table 22. Support Information for IATUX58 (continued)

<table>
<thead>
<tr>
<th>SSXINDEX</th>
<th>JES3 Calling Module</th>
<th>Security Requested by JES3</th>
<th>Additional Information IATUX58 Can Examine</th>
<th>Additional Information IATUX58 Can Modify</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSXIXDMD</td>
<td>IATDMDM</td>
<td>Extract token of a demand select job to pass with JCL in staging area.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSXIXDME</td>
<td>IATDMEB</td>
<td>Extract token of a TSO submit job</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSXIXJNW</td>
<td>IATINGN</td>
<td>Extract JES3’s token.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSXIXSIA</td>
<td>IATSIAD</td>
<td>Obtain UTOKEN for PSO processing during unallocation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSXIXSIC</td>
<td>IATSICC</td>
<td>Extract token of internal reader jobs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSXIXSCN</td>
<td>IATSICN</td>
<td>Request the user’s security token to be passed to the global for the job to be canceled.</td>
<td>• SSXMCNTL – Specifies where messages are to be displayed</td>
<td></td>
</tr>
<tr>
<td>SSXIXSIJ</td>
<td>IATSIJS</td>
<td>Extract token for SYSLOG</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSXIXPAL</td>
<td>IATSIOP</td>
<td>Extract token for PSO request to the global</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSXIXSAD</td>
<td>IATSIAD</td>
<td>Extract data set token.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSXIXSIR</td>
<td>IATSIOR</td>
<td>Extract token of job’s address space for internal reader sysout allocation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSXIXNUM</td>
<td>IATSINU</td>
<td>Notify user token.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| SSXIVNJE | IATCNNJ             | Verify the node for command entry and obtain the user security token (SSXUTOKN) of the node. | • SSXREMOT – Specifies whether the node is remote  
• SSXPASCK – Specifies whether the password should be checked  
• SSXPOE – Name of the entry point  
• SSXSNODE – Origin node name from the job header  
• SSXSTOKN – Operator’s security token  
• SSXUSERI – Userid  
• SSXMCNTL – Specifies where messages are to be displayed | • SSXPASCK  
• SSXPOE  
• SSXSNODE  
• SSXSTOKN  
• SSXUSERI |
<table>
<thead>
<tr>
<th>SSXINDEX</th>
<th>JES3 Calling Module</th>
<th>Security Requested by JES3</th>
<th>Additional Information IATUX58 Can Examine</th>
<th>Additional Information IATUX58 Can Modify</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSXIVNOI</td>
<td>IATISNJ</td>
<td>Create internal reader token for an outbound NJE job stream.</td>
<td>• SSXPOE – Name of the entry point&lt;br&gt;• SSXSTOKN – Operator's security token&lt;br&gt;• SSXPASCK – Specifies whether the password should be checked&lt;br&gt;• SSXMCNTL – Specifies where messages are to be displayed&lt;br&gt;• SSXENCRT – Specifies the encryption method to be used&lt;br&gt;• SSXEXNOD – Specifies the address of the execution node&lt;br&gt;• SSXSNODE – Origin node name from the job header&lt;br&gt;• SSXNJEJH – Address of the NJE job header</td>
<td>• SSXPOE&lt;br&gt;• SSXSTOKN&lt;br&gt;• SSXPASCK&lt;br&gt;• SSXEXNOD&lt;br&gt;• SSXSNODE</td>
</tr>
<tr>
<td>SSXIVNOE</td>
<td>IATISNJ</td>
<td>Create an unknown user token for an outbound NJE job stream.</td>
<td>• SSXPOE – Name of the entry point&lt;br&gt;• SSXPASCK – Specifies whether the password should be checked&lt;br&gt;• SSXMCNTL – Specifies where messages are to be displayed&lt;br&gt;• SSXENCRT – Specifies the encryption method to be used&lt;br&gt;• SSXEXNOD – Specifies the address of the execution node&lt;br&gt;• SSXSNODE – Origin node name from the job header</td>
<td>• SSXPOE&lt;br&gt;• SSXPASCK&lt;br&gt;• SSXEXNOD&lt;br&gt;• SSXSNODE</td>
</tr>
<tr>
<td>SSXINDEX</td>
<td>JES3 Calling Module</td>
<td>Security Requested by JES3</td>
<td>Additional Information IATUX58 Can Examine</td>
<td>Additional Information IATUX58 Can Modify</td>
</tr>
<tr>
<td>-----------</td>
<td>---------------------</td>
<td>--------------------------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------</td>
<td>---------------------------------------------</td>
</tr>
<tr>
<td>SSXIVNJV</td>
<td>IATISNJ</td>
<td>Check that a job is valid for an outbound NJE job stream.</td>
<td>• SSXPOE – Name of the entry point</td>
<td>• SSXPOE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• SSXSTOKN – Operator’s security token</td>
<td>• SSXPASCK</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• SSXPASCK – Specifies whether the password should be checked</td>
<td>• SSXGROUP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• SSXMCNTL – Specifies where messages are to be displayed</td>
<td>• SSXJOBNM</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• SSXENCRT – Specifies the encryption method to be used</td>
<td>• SSXNPASS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• SSXEXNOD – Specifies the address of the execution node</td>
<td>• SSXPASWD</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• SSXSNODE – Origin node name from the job header</td>
<td>• SSXSECLB</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• SSXNJEJH – Address of the NJE job header</td>
<td>• SSXSNODE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• SSXGROUP – Group name</td>
<td>• SSXNODE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• SSXJOBNM – Job name</td>
<td>• SSXSUSRI</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• SSXNPASS – New password</td>
<td>• SSXEXNOD</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• SSXPASWD – Password</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• SSXSECLB – Security label</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• SSXUSERI – Userid</td>
<td></td>
</tr>
<tr>
<td>SSXIVNIS</td>
<td>IATNTJS</td>
<td>Validate a job for a SYSOUT stream received from another node.</td>
<td>• SSXPOE – Path node name for BSC “NJERDR” for SNA</td>
<td>• SSXPOE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• SSXSNODE – Origin node name from the job header</td>
<td>• SSXSNODE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• SSXNJEJH – Address of the NJE job header</td>
<td>• SSXSSTOKN</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• SSXSUSRI – Submittor’s userid</td>
<td>• SSXSUSRI</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• SSXSTOKN – Operator’s security token</td>
<td>• SSXGROUP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• SSXEXNOD – Address of execution node</td>
<td>• SSXJOBNM</td>
</tr>
</tbody>
</table>

Table 22. Support Information for IATUX58 (continued)
<table>
<thead>
<tr>
<th>SSXINDEX</th>
<th>JES3 Calling Module</th>
<th>Security Requested by JES3</th>
<th>Additional Information IATUX58 Can Examine</th>
<th>Additional Information IATUX58 Can Modify</th>
</tr>
</thead>
</table>
| SSXIVNIU | IATNTJS             | Create an unknown user token for an inbound NJE SYSOUT stream. | • SSXPOE – Name of the entry point  
• SSXEXNOD – Specifies the address of the execution node  
• SSXSNODE – Origin node name from the job header  
• SSXNJEJH – Address of the NJE job header | • SSXPOE  
• SSXSNODE  
• SSXEXNOD |
| SSXIVNJR | IATNTJS             | Create an unknown user token for a Store and Forward NJE job stream. | • SSXPOE – Name of the entry point  
• SSXEXNOD – Specifies the address of the execution node  
• SSXSNODE – Origin node name from the job header  
• SSXNJEJH – Address of the NJE job header | • SSXPOE  
• SSXSNODE  
• SSXEXNOD |
| SSXIVNRR | IATNTRS             | Job validation for a job rerouted to a remote node. | • SSXPASCK – Specifies whether the password should be checked  
• SSXMCNTL – Specifies where messages are to be displayed  
• SSXNJEJH – Address of NJE job header | • SSXPASCK |
| SSXIVNRS | IATNTRS             | Job validation for a SYSOUT stream rerouted to the home node where the origin was a remote node. | • SSXPOE – Name of the entry point  
• SSXSNODE – Origin node name from the job header  
• SSXSUSRI – Submitter’s userid  
• SSXSTOKN – Operator’s security token  
• SSXMCNTL – Specifies where messages are to be displayed  
• SSXPASCK – Specifies whether the password should be  
• SSXNJEJH – Address of NJE job header  
• SSXEXNOD – Address of execution node | • SSXPASCK  
• SSXPOE  
• SSXSTOKN  
• SSXSUSRI  
• SSXSNODE  
• SSXEXNOD |
<table>
<thead>
<tr>
<th>SSXINDEX</th>
<th>JES3 Calling Module</th>
<th>Security Requested by JES3</th>
<th>Additional Information IATUX58 Can Examine</th>
<th>Additional Information IATUX58 Can Modify</th>
</tr>
</thead>
</table>
| SSXIVNRU | IATNTRS             | Create an unknown user token to get UNKNOWN userid value for a rerouted NJE SYSOUT stream. | - SSXPOE – Name of the entry point  
- SSXPASCK – Specifies whether the password should be checked  
- SSXMCNTL – Specifies where messages are to be displayed  
- SSXEXNOD – Specifies the address of the execution node  
- SSXSNODE – Origin node name from the job header  
- SSXNJEJH – Address of the NJE job header | - SSXPOE  
- SSXPASCK  
- SSXSNODE  
- SSXEXNOD |
| SSXIVJS | IATISEN             | Validate jobs other than demand select jobs. Update submitter’s node to home node for all jobs except NJE jobs. | - SSXPOE – Name of the entry point  
- SSXPASCK – Specifies whether the password should be checked  
- SSXGROUP – Group name  
- SSXMCNTL – Specifies where messages are to be displayed  
- SSXJOBNM – Job name  
- SSXNPASS – New password  
- SSXPASWD – Password  
- SSXSECLB – Security label  
- SSXSNODE – Origin node name  
- SSXSTOKN – Submitter’s security token  
- SSXUSERI – Userid  
- SSXENCRY – Encrypt address of password  
- SSXREMOT – Specifies whether the job was submitted locally | - SSXPOE  
- SSXPASCK  
- SSXGROUP  
- SSXJOBNM  
- SSXNPASS  
- SSXPASWD  
- SSXSECLB  
- SSXSNODE  
- SSXSTOKN  
- SSXUSERI |
Table 22. Support Information for IATUX58 (continued)

<table>
<thead>
<tr>
<th>SSXINDEX</th>
<th>JES3 Calling Module</th>
<th>Security Requested by JES3</th>
<th>Additional Information IATUX58 Can Examine</th>
<th>Additional Information IATUX58 Can Modify</th>
</tr>
</thead>
</table>
| SSXIVRJM | IATRJM3             | Verify the BSC RJP terminal SIGNON and allow the password to be changed. | • SSXPASCK – Specifies whether the password should be checked  
• SSXPASWD – Existing password  
• SSXNPASS – New password  
• SSXPOE – Name of the entry point  
• SSXSTOKN – Operator’s security token  
• SSXUSERI – Userid  
• SSXMCNTL – Specifies where messages are to be displayed | • SSXPASCK  
• SSXPASWD  
• SSXNPASS  
• SSXPOE  
• SSXSTOKN  
• SSXUSERI  
• SSXIVSNL  
| IATSNLS  |                     | Verifies the SNA RJP terminal LOGON and allows the password to be changed. | • SSXPASCK – Specifies whether the password should be checked  
• SSXMCNTL – Specifies where messages are to be displayed  
• SSXPASWD – Existing password  
• SSXNPASS – New password  
• SSXPOE – Name of the entry point  
• SSXSTOKN – Operator’s security token  
• SSXUSERI – Userid | • SSXPASCK  
• SSXPASWD  
• SSXPOE  
• SSXSTOKN  
• SSXUSERI  
• SSXNPASS |
<table>
<thead>
<tr>
<th>SSXINDEX</th>
<th>JES3 Calling Module</th>
<th>Security Requested by JES3</th>
<th>Additional Information IATUX58 Can Examine</th>
<th>Additional Information IATUX58 Can Modify</th>
</tr>
</thead>
</table>
| SSXIVDJ1 | IATDJIN             | Authorization for a job restored by the dump job facility from an earlier version of JES3 to reenter the system. | • SSXPASWD – Existing password  
• SSXPOE – Name of the entry point  
• SSXSNODE – Origin node name from the job header  
• SSXSGRP – Submittor’s group  
• SSXSUSRI – Submittor’s userid  
• SSXUSERI – Userid  
• SSXGROUP – Group  
• SSXSTOKN – Operator’s security token  
• Register 3 – Password  
• Register 4 – Type of session. The possible values are:  
  – SSXSSBAT – Indicator for a batch session  
  – SSXSSSTR – Indicator for a started task session  
  – SSXSSTSO – Indicator for a TSO session | • SSXPASCK  
• SSXPASWD  
• SSXPOE  
• SSXSTOKN  
• SSXUSERI  
• SSXSUSER  
• SSXJOBNM  
• SSXSNODE  
• SSXGROUP |
| SSXIVDJ2 | IATDJIN             | Verify that the token is available for the specified job. | • SSXMCNTL – Specifies where messages are to be displayed  
• SSXREMOT – Specifies that the job was submitted locally |  |
| SSXIVDJ3 | IATDJIN             | Verify a job with a session type of TKN UNKNOWN. | • SSXREMOT – Specifies that the job was submitted locally |  |
| SSXAUJVL | IATINJR             | Checks if a job was deleted during restart processing. | • SSXLGSTR – String of characters that will be added to the SMF  
• SSXMCNTL – Specifies where messages are to be displayed  
• SSXINDEX – Specifies the type of security call  
• SSXMODE – Specifies the mode of the security caller | • SSXLGSTR |
<table>
<thead>
<tr>
<th>SSXINDEX</th>
<th>JES3 Calling Module</th>
<th>Security Requested by JES3</th>
<th>Additional Information IATUX58 Can Examine</th>
<th>Additional Information IATUX58 Can Modify</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSXAUCAN</td>
<td>IATGRWM</td>
<td>Checks that a job that was in DELETE-ONLY status has been deleted from the JES3 Job Queue in response to an Operator CANCEL command (*F,J=JOBNAME,C).</td>
<td>- SSXLGSTR – String of characters that will be added to the SMF</td>
<td>- SSXINDEX</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- SSXMCNTL – Specifies where messages are to be displayed</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- SSXINDEX – Specifies the type of security call</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- SSXMODE – Specifies the mode of the security caller</td>
<td></td>
</tr>
<tr>
<td>SSXVFCRT</td>
<td>IATIIST</td>
<td>Creates security environment for converter to determine if the user has authorization to access the JCLLIB data set.</td>
<td>- SSXLGSTR – Started procedure name, if the IATXSEC call was made during C/I processing of a started procedure.</td>
<td>- SSXSTPRC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- SSXTOKIN – Specifies the input token</td>
<td>- SSXPASCK</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- SSXPASCK – Specifies whether the password should be checked.</td>
<td>- SSXTOKIN</td>
</tr>
<tr>
<td>SSXVFDEL</td>
<td>IATIIST</td>
<td>Deletes the security environment which was created by SSXVFCRT.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Appendix B. Accessibility

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use software products successfully. The major accessibility features in z/OS enable users to:

- Use assistive technologies such as screen readers and screen magnifier software
- Operate specific or equivalent features using only the keyboard
- Customize display attributes such as color, contrast, and font size

Using assistive technologies

Assistive technology products, such as screen readers, function with the user interfaces found in z/OS. Consult the assistive technology documentation for specific information when using such products to access z/OS interfaces.

Keyboard navigation of the user interface

Users can access z/OS user interfaces using TSO/E or ISPF. Refer to z/OS TSO/E Primer, z/OS TSO/E User's Guide, and z/OS ISPF User's Guide Vol I for information about accessing TSO/E and ISPF interfaces. These guides describe how to use TSO/E and ISPF, including the use of keyboard shortcuts or function keys (PF keys). Each guide includes the default settings for the PF keys and explains how to modify their functions.

z/OS information

z/OS information is accessible using screen readers with the BookServer/Library Server versions of z/OS books in the Internet library at:

http://www.ibm.com/systems/z/os/zos/bkserv/
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
USA

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 to 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 information 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.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

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 Corporation
Mail Station P300
2455 South Road
Poughkeepsie, NY 12601-5400
USA

Such information may be available, subject to appropriate terms and condition
including in some cases, payment of a fee.

The licensed program described in this information and all licensed material
available for it are provided by IBM under terms of the IBM Customer Agreement,
IBM International Program License Agreement, or any equivalent agreement
between us.

This information contains examples of data and reports used in daily business
operations. To illustrate them as completely as possible, the examples include the
names of individuals, companies, brands, and products. All of these names are
fictitious and any similarity to the names and addresses used by an actual business
enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which
illustrates programming techniques on various operating platforms. You may copy,
modify, and distribute these sample programs in any form without payment to IBM,
for the purposes of developing, using, marketing or distributing application programs
conforming to the application programming interface for the operating platform for
which the sample programs are written. These examples have not been thoroughly
tested under all conditions. IBM, therefore, cannot guarantee or imply reliability,
serviceability, or function of these programs. You may copy, modify, and distribute
these sample programs in any form without payment to IBM for the purposes of
developing, using, marketing, or distributing application programs conforming to
IBM's application programming interfaces.

If you are viewing this information softcopy, the photographs and color illustrations
may not appear.

Programming Interface Information

This publication documents intended Programming Interfaces that allow the
customer to write programs to obtain the services of JES3.

Policy for unsupported hardware

Various z/OS elements, such as DFSMS, HCD, JES2, JES3, and MVS, contain
code that supports specific hardware servers or devices. In some cases, this
device-related element support remains in the product even after the hardware
devices pass their announced End of Service date. z/OS may continue to service
element code; however, it will not provide service related to unsupported hardware
deVICES. Software problems related to these devices will not be accepted for
service, and current service activity will cease if a problem is determined to be
associated with out-of-support devices. In such cases, fixes will not be issued.
Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Other company, product, or service names may be trademarks or service marks of others.
Index

Numerics
16Mb (16-megabyte) address 2
JES3 code 57
referring to data below 16 megabytes 2, 19
SWA above 16 megabytes 93
31-bit addressing mode 19
code considerations 19

A
ABACKR macro
description 177
restrictions 178
abend 352
format 352
ABLOCK macro
description 178
restrictions 178
ABNCODE macro
description 180
restrictions 180
ABNCVDEC macro
description 181
restrictions 181
ABNCVHEX macro
description 182
restrictions 182
ABNGET macro
description 182
restrictions 182
ABNPUT macro
description 184
restrictions 184
ABNVRFY macro
description 184
restrictions 184
ACALL macro
description 185
restrictions 185
ACB (access method control block) 315
accessibility 631
accounting information 107
collecting 107
ACLOSE macro
description 187
restrictions 187
ACS (automatic class selection) 63
exit in SMS 63
ACVX macro
description 189
restrictions 189
adding a DEVICE statement
IATDEVA macro 49
adding to the DSP dictionary 4
address space
global 58
JES3 57
address space (continued)
local 58
addressing mode 1
16-megabyte address 21
AMODE statement 20
residency mode 58
RMODE statement 20
rules 174
virtual storage 1
ADEBLOCK macro
description 189
restrictions 189
ADEDELETE macro
description 190
restrictions 190
ADEQ macro
description 192
restrictions 192
adjacent node 54
AENQ macro
description 193
restrictions 193
AFP 84
AFP printer 90
AGETBUFF macro
description 194
restrictions 194
AGETMAIN macro
description 195
restrictions 195
AJES3MAC data set 3
AJES3SRC data set 3
AJOBNUM macro
description 196
restrictions 196
algorithm for converting routing codes 129
installation exit 57 129
algorithm for setup 69
job setup 69
allocation 540
failure 540
ALOAD macro
description 199
restrictions 199
ALLOCATE macro
description 201
restrictions 201
ANOTE macro
description 202
restrictions 202
AOPEN macro
description 203
restrictions 203
AOPEND macro
description 204
restrictions 204
APAR 21
APAR number 21

© Copyright IBM Corp. 1988, 2009 637
APAR (continued)
   APARNUM field 602
APPOINT macro
   description 206
   restrictions 206
APPC (advanced program-to-program communication) 43, 102
APURGE macro
   description 207
   restrictions 207
APUTBUF macro
   description 208
   restrictions 208
APUTMAIN macro
   description 209
   restrictions 209
ARELEASE macro
   description 210
   restrictions 210
ARETURN macro
   description 211
   restrictions 211
ARL (allocation requirements list) 234
   IATXARL macro 234
ASAVE routine
   save area 26
ATEST macro
   description 211
   restrictions 211
ATIME macro
   description 213
   restrictions 213
ATRACK macro
   description 217
   restrictions 217
authority checking for TSO/E commands 99
   authority level 79
   defining for commands 79
auxiliary task 45
   JES3 services available 45
AUXTASK mode 469, 475
AWA control block 102
AWAIT macro
   description 218
   nonserialized resources 29
   restrictions 218
AWRITE macro
   description 220
   restrictions 220

B

banner page
   separator page 82
BDT (bulk data transfer) 61
   BDT language processor (IATXLP macro) 346
   BSID data area 128
   installation exits involved 61
   lost transaction 367
   shuttle staging area 128
   transmit data set header 112
   BFE (buffer entry) 351
   binary job numbers 336
   block letter pattern
      in job header 85
      block spooler 242
   allocating 425
   invoking 242
   broadcast messages 550

C

C/I (converter/interpreter) 2
   C/I driver module 249
   C/I FSS address space 63
   C/I processing 62
   CI DSP data area 66
   converter/interpreter (C/I) 2
   FSS address space 249
   postscan phase 68, 70
   postscan process 67
   calling
      JES3 services 21
   cancel job 146
   carriage control data 376
      CC parameter 376
   cell pool 247
      allocate 300
      create 244
      primary cell pool extent 247
   chained SRF 32
   check parameters
      using IATZPCHK macro 497
   CID (communication identifier) 293
   coding macro instructions 169
   coding rule
      macro 173
   color console 36
   command 5
      *FAan 534
      *MODIFY 45, 469, 601
      CALL 5, 601
      CANCEL 5, 416
      LOGOFF 416
      MODIFY 5, 106
      scanning operands 423
      START 5
      syntax diagrams 169
   common area
      JES3 57
   common quick cell services 245
   communication
      operator communication 22
   CONCNVRT macro
      description 222
      restrictions 222
   CONREVRT macro
      description 223
      restrictions 223
   console appendage 34
      communicate to operator 34
   LOGIN macro 35
console destination 222
  class name 223
  convert to message 222
console service 224
  console input buffer 261
  console profile mask 224
console support 36
  color consoles 36
  define console messages 41
  intensify a message 36
  multiple (MCS) 36
  reserve a cell 247
CONSPROF macro
  description 224
  restrictions 224
control block 68
  identifier 68
  JCT (job control table) 94
  JDAB (job description accounting block) 94
  JMR (job management record) 94
CSECT name 61
  IATUX03 61
  IATUX04 63
  IATUX05 64
  IATUX06 65
  IATUX07 67
  IATUX08 68
  IATUX09 70
  IATUX10 71
  IATUX11 73
  IATUX14 74
  IATUX15 76
  IATUX17 78
  IATUX18 79
  IATUX19 81
  IATUX20 82
  IATUX21 85
  IATUX22 87
  IATUX23 88
  IATUX24 90
  IATUX25 91
  IATUX26 93
  IATUX27 94
  IATUX28 95
  IATUX29 96
  IATUX30 99
  IATUX31 102
  IATUX33 103
  IATUX34 104
  IATUX35 106
  IATUX36 107
  IATUX37 109
  IATUX38 110
  IATUX39 112
  IATUX40 114
  IATUX41 115
  IATUX42 116
  IATUX43 118
  IATUX44 120
  IATUX45 121
  IATUX46 123

CSECT name (continued)
  IATUX48 125
  IATUX49 126
  IATUX50 128
  IATUX57 129
  IATUX58 131
  IATUX59 138
  IATUX60 145
  IATUX61 146
  IATUX62 148
  IATUX63 150
  IATUX66 152
  IATUX67 154
  IATUX68 155
  IATUX69 156
  IATUX70 159

D
  data 362
    moving, IATXMVXM macro 362
    moving, MVCL instruction 561
  data set 3
    data set security 125
    modification 125
    overriding the MODIFY command 125
    SYS1.AJES3MAC data set 3
  data set header 109
    modification 109, 112
    networking 109
  data transmission
    receive a file 116
  DD statement 104
    command validity check 106
    JCL (job control language) 104
  defined
    scheduler elements 78
  DEQMSG macro
    description 224
    restrictions 224
  destination class 41
    mapping to a routing code 554
    select a single routing code 41, 129
  destination queue entry 227
    mark active 227
  device 231
    fence 91
    unavailable JES3-managed device 231
  device entry table 487
    IATYDVE 487
  DEVSCAN macro
    description 225
    restrictions 225
  diagnosis
    tracing register contents 173
    disability 631
  DJC checkpoint record 566
  DJST (dynamic job summary table) 276
    build step entry in slot 304
    dequeue a slot 276
    locating a slot 345
DLOCOFF macro
   description 226
   restrictions 226

DLOCON macro
   description 227
   restrictions 227

DM code 201
   D23 548
   DM080 201
   DM081 201
   DM083 201
   DM102 312
   DM132 548

DRT (device requirements table) 4, 24
DSP (dynamic support program) 1
   adding to the DSP dictionary 4
   assemble 3
   CONSERV DSP and MESSAGE macro 548
   dictionary 4
   DSP driver module name 601
   dynamic support program 1
   failure 229
   initialization 23
   introduction 2
   link-edit 3
   POSTSCAN 70
   protect 5
   registers on entry 23
   start a DSP 5
   termination 24
DSP dictionary 57
DSQLOC macro
   description 228
   restrictions 228

dummy installation exit 16, 26
dummy SMF 26 record 94
dump codes 177
dumps for JES3 20, 230
   APAR number 21
   module name in dump output 20
   PTF number 21
DYNALDSN initialization statement 102

E
ECF (event completion flag) 28, 284
   delete ECF mask 284
ELB (ECF list control block) 283
   enqueuing
      use 31
   error 28
   recoverable 28
ERROR parameter
   general description 177
EXEC statement
   JCL (job control language) 103
execution mode 469
   AUXTASK 469, 475
   NUCLTASK 469, 475
extended area for JES3
   JES3 57

F
FAILDSP macro
   description 229
   restrictions 229
FCB (forms control buffer) 350
   forms control buffer (FCB) 350
FCT (function control table) 5
   function control table (FCT) 5
FCTAGET field 196
FCTFLAG1 field 196
FDB (file description block) 29
   for the JDAB 29
   writing a JSAM buffer 32
fields 94
   JDABFLG2 field 94
fixing pages
   page-fix 391
flags 94
   FCTCFLG 189
   FCTSPFLG 189
   FDBMULT 222
   FDBONSP 222
   JDABNSMF 94
forms alignment 87
FSA (functional subsystem application) 53
FSI (functional subsystem interface) 22
   FSS (functional subsystem) 22
FSS (functional subsystem) 22
   address space 47
   AFP printer 87
   FSI (functional subsystem interface) 22
   set up an FSS environment 295
   writer-get-data-set (GETDS) 121
function control table 5
   address of FCT 23
   changing priority 5
   define the FCT 22
   FCT chain 5
   generate permanent FCT 598
   obtaining 305
functional subsystem
   FSS (functional subsystem) 22

G
generalized subtask facility 21
generate a message
   message 71
generate an AWAIT reason code 584
GETUNIT LIST 24
GETUNIT macro
   description 230
   restrictions 230
   global
   address space 58
   global address space
   JES3 69
H
HWS (high-watermark setup)  69
  installation exit 8  69
  types  69

I
I/O driver
  for JES3  57
I/O operation  29
  example of macros  29
I/O macros  29
  JESCLOSE  29
  JESEXCP  29
  JESOPEN  29
  local and remote devices  29
IATDEVA macro  49
IATDEVC macro  50
IATJ3ASM in SYS1.SIATSAMP  60
IATSNDC module  490
IATUX69  156
IATUX70  159
IATXADD macro
  description  232
  restrictions  232
IATXAMDV macro
  description  233
  restrictions  233
IATXARL macro
  description  234
  restrictions  234
IATXARQ macro
  description  235
  restrictions  235
IATXASR macro
  description  236
  restrictions  236
IATXATDE macro
  description  237
  restrictions  237
IATXATF macro
  description  238
  restrictions  238
IATXBDSN macro
  description  238
  restrictions  238
IATXBFM macro
  description  239
  restrictions  239
IATXBGM macro
  description  240
  restrictions  240
IATXBIKO macro
  description  242
  restrictions  242
IATXBPPL macro
  description  244
  restrictions  244
IATXBOQN macro
  description  247
  IATXBOQN macro  (continued)
    restrictions  247
IATXCDVE macro
    description  248
    restrictions  248
IATXCIO macro
    description  249
    restrictions  249
IATXCKPT macro
    description  251
    restrictions  251
IATXCNDB macro
    description  254
    restrictions  254
IATXCNNS macro
    description  261
    restrictions  261
IATXCNT macro
    description  263
    restrictions  263
IATXCPYF macro  264
  copy one multi-record file to another  264
    description  264
IATXCRPL macro
    description  266
    restrictions  266
IATXCSF macro
    description  267
    restrictions  267
IATXCUE macro
    description  270
    restrictions  270
IATXCVB macro
    description  272
    restrictions  272
IATXCVD macro
    description  273
IATXDATX macro
    description  275
IATXDDEL macro
    description  276
    restrictions  276
IATXDEQ macro
    description  277
    restrictions  277
IATXDEV macro
    description  277
    restrictions  277
IATXDPL macro
    description  279
    restrictions  279
IATXDST macro
    description  280
    restrictions  280
IATXDYH macro
    description  281
    restrictions  281
IATXDYT macro
    description  282
    restrictions  282
<table>
<thead>
<tr>
<th>Macro</th>
<th>Description</th>
<th>Restrictions</th>
</tr>
</thead>
<tbody>
<tr>
<td>IATXELA</td>
<td>283</td>
<td>283</td>
</tr>
<tr>
<td>IATXELD</td>
<td>284</td>
<td>284</td>
</tr>
<tr>
<td>IATXELS</td>
<td>285</td>
<td>285</td>
</tr>
<tr>
<td>IATXENQ</td>
<td>287</td>
<td>287</td>
</tr>
<tr>
<td>IATXENT</td>
<td>288</td>
<td>288</td>
</tr>
<tr>
<td>IATXERCK</td>
<td>289</td>
<td>289</td>
</tr>
<tr>
<td>IATXERCV</td>
<td>290</td>
<td>290</td>
</tr>
<tr>
<td>IATXFDB</td>
<td>291</td>
<td>291</td>
</tr>
<tr>
<td>IATXFIFO</td>
<td>292</td>
<td>292</td>
</tr>
<tr>
<td>IATXFLCB</td>
<td>293</td>
<td>293</td>
</tr>
<tr>
<td>IATXFRQ</td>
<td>294</td>
<td>294</td>
</tr>
<tr>
<td>IATXFSS</td>
<td>295</td>
<td>295</td>
</tr>
<tr>
<td>IATXFSV</td>
<td>299</td>
<td>299</td>
</tr>
<tr>
<td>IATXFWSB</td>
<td>299</td>
<td>299</td>
</tr>
<tr>
<td>IATXGCL</td>
<td>300</td>
<td>300</td>
</tr>
<tr>
<td>IATXGENF</td>
<td>302</td>
<td>302</td>
</tr>
<tr>
<td>IATXGET</td>
<td>304</td>
<td>304</td>
</tr>
<tr>
<td>IATXGFC</td>
<td>305</td>
<td>305</td>
</tr>
<tr>
<td>IATXGF</td>
<td>307</td>
<td>307</td>
</tr>
<tr>
<td>IATXGFM</td>
<td>309</td>
<td>309</td>
</tr>
<tr>
<td>IATXGFM</td>
<td>345</td>
<td>345</td>
</tr>
<tr>
<td>IATXGOSE</td>
<td>310</td>
<td>310</td>
</tr>
<tr>
<td>IATXGRQ</td>
<td>311</td>
<td>311</td>
</tr>
<tr>
<td>IATXGSV</td>
<td>313</td>
<td>313</td>
</tr>
<tr>
<td>IATXHOME</td>
<td>314</td>
<td>314</td>
</tr>
<tr>
<td>IATXICA</td>
<td>315</td>
<td>315</td>
</tr>
<tr>
<td>IATXIMSG</td>
<td>317</td>
<td>317</td>
</tr>
<tr>
<td>IATXIOE</td>
<td>316</td>
<td>316</td>
</tr>
<tr>
<td>IATXIOX</td>
<td>316</td>
<td>316</td>
</tr>
<tr>
<td>IATXISMG</td>
<td>317</td>
<td>317</td>
</tr>
<tr>
<td>IATXIWT</td>
<td>318</td>
<td>318</td>
</tr>
<tr>
<td>IATXJBNO</td>
<td>320</td>
<td>320</td>
</tr>
<tr>
<td>IATXJBTS</td>
<td>320</td>
<td>320</td>
</tr>
<tr>
<td>IATXJCT</td>
<td>323</td>
<td>323</td>
</tr>
<tr>
<td>IATXJDS</td>
<td>326</td>
<td>326</td>
</tr>
<tr>
<td>IATXJLOC</td>
<td>326</td>
<td>326</td>
</tr>
<tr>
<td>IATXJMR</td>
<td>332</td>
<td>332</td>
</tr>
<tr>
<td>IATXJOB</td>
<td>334</td>
<td>334</td>
</tr>
<tr>
<td>IATXJQE</td>
<td>336</td>
<td>336</td>
</tr>
<tr>
<td>IATXJL3</td>
<td>340</td>
<td>340</td>
</tr>
<tr>
<td>IATXLOC</td>
<td>347</td>
<td>347</td>
</tr>
<tr>
<td>IATXLP</td>
<td>345</td>
<td>345</td>
</tr>
<tr>
<td>Writing a message</td>
<td>522</td>
<td>522</td>
</tr>
</tbody>
</table>
IATXLP macro (continued) restrictions 346
IATXLRT macro description 349 restrictions 349
IATXLRT macro description 350 restrictions 350
IATXLRT macro description 350 restrictions 350
IATXMGBD macro description 351 restrictions 351
IATXMLGET macro description 352 restrictions 352
IATXMID macro description 354 restrictions 354
IATXMLLWO macro description 355 restrictions 355
IATXMMNMX macro description 357 restrictions 357
IATXMPC macro description 358 restrictions 358
IATXMVDA macro description 361 restrictions 361
IATXMVXW macro description 362 restrictions 362
IATXNGRS macro description 363 restrictions 363
IATXNTS macro description 364 restrictions 364
IATXOSBM macro description 366 restrictions 366
IATXOSCI macro description 368 restrictions 368
IATXOSCO macro description 369 restrictions 369
IATXOSG macro description 370 restrictions 370
IATXOSOI macro description 372 restrictions 372
IATXOSOO macro description 373 restrictions 373
IATXOSP macro description 374 restrictions 374
IATXOSP macro description 377
IATXRCL macro (continued)  
restrictions 408

IATXRCVL macro  
description 409  
restrictions 409

IATXRDCCH macro  
description 410  
read a chained single-record file into storage 410

IATXRELC macro  
description 412  
restrictions 412

IATXRMEP macro  
description 413  
restrictions 413

IATXRST macro  
description 414  
restrictions 414

IATXRUGT macro  
description 415  
restrictions 415

IATXRUPT macro  
description 416  
restrictions 416

IATXSAS macro  
description 417  
restrictions 417

IATXSCH macro  
description 419  
restrictions 419

IATXSCHN1 macro  
description 421  
restrictions 421

IATXSCHN2 macro  
description 423  
restrictions 423

IATXSDM macro 31  
description 424  
restrictions 424

IATXSEC macro  
description 429  
restrictions 429

IATXSIO macro  
description 439  
restrictions 439

IATXSMF macro  
description 440  
restrictions 440

IATXSMGR macro  
description 441  
restrictions 441

IATXSNFG macro  
description 442  
restrictions 442

IATXSNFM macro  
description 443  
restrictions 443

IATXSNFS macro  
description 445  
restrictions 445

IATXSNGM macro  
description 448

IATXSNGM macro (continued)  
restrictions 448

IATXSNLK macro  
description 450  
restrictions 450

IATXSNMF macro  
description 454  
restrictions 454

IATXSNST macro  
description 455  
restrictions 455

IATXSNTR macro  
description 458  
restrictions 458

IATXSPCK macro  
description 459  
restrictions 459

IATXSPR macro  
description 460  
restrictions 460

IATXSQE macro  
description 461  
restrictions 461

IATXSRS macro  
description 463  
restrictions 463

IATXSTAT macro  
description 463  
restrictions 463

IATXSTMD macro  
description 469  
restrictions 469

IATXSTTA macro  
description 471  
restrictions 471

IATXSYSU macro 472  
access SYSUNIT tables 472  
description 472

IATXTEST macro  
description 474  
restrictions 474

IATXTOD macro  
description 475  
restrictions 475

IATXTRC macro  
description 476  
restrictions 476

IATXTREG macro  
description 478  
restrictions 478

IATXTRMT macro  
description 478  
restrictions 478

IATXUBAL macro  
description 479  
restrictions 479

IATXVAL macro  
description 480  
restrictions 480

IATXVMSG macro  
description 484
IATXVMSG macro (continued)  
restrictions 484
IATXVSRE macro  
description 485  
restrictions 485
IATXVSRV macro  
description 485  
restrictions 485
IATXWLST macro  
description 487  
restrictions 487
IATXWOPN macro  
description 487  
restrictions 487
IATXWRE macro  
description 489  
restrictions 489
IATXWSCL macro  
description 490  
restrictions 490
IATYASM macro  
generate control blocks 583
IATYAWDR macro  
generate control blocks 584
IATYAWR macro  
generate control blocks 585
IATYCNS macro  
generate control blocks 587
IATYDAT data area 370  
DATCC field 370
IATYDJF macro  
generate control blocks 588
IATYDJR macro  
generate control blocks 593
IATYDSD macro  
generate control blocks 595
IATYFCD macro  
generate control blocks 598
IATYFCT macro 22
IATYMOD macro  
generate control blocks 602
IATYRSC macro  
generate control blocks 603
IATYSRF macro 31  
generate control blocks 603
IATYSVT data area 281
IATYTVT macro 22
IATZCALL macro  
description 491  
restrictions 491
IATZHEX macro  
description 492  
restrictions 492
IATZLOAD macro  
description 493  
restrictions 493
IATZMEXC macro  
description 494  
restrictions 494
IATZMNOT macro (continued)  
restrictions 495
IATZMOVE macro  
description 496  
restrictions 496
IATZPARM macro  
description 497  
restrictions 497
IATZPCHK macro 497  
description 497  
parameter checker 497  
verify parameters 497
IATZTYPE macro  
description 502  
restrictions 502
ICARDRD macro  
description 502  
restrictions 502
ICONVBIN macro  
description 503  
restrictions 503
ICONVHEX macro  
description 503  
restrictions 503
IEFUSI exit 195, 448
IJS (intermediate job summary table) 67
IKJEFF53 TSO/E exit 99
INISH parameter  
on IATYFCD macro 600  
initialization 61  
initialization stream check 77  
installation exits involved 61, 74  
PROCESS 10  
scan initialization statement 76
INITMWLE macro  
description 504  
restrictions 504
INPOINT field 507
INQUIRY command 106
installation exit main mask 124
installation exit routine 16  
description 58  
IATUX10 320  
IATUX31 41  
IATUX57 41  
IATUX69 41  
IATUX70 41  
introduction 16  
listed by JES3 function 60  
protect a resource 5  
test a DSP 45  
test installation exit 45
installation exits for modification  
data set header 55  
incoming data 55  
job header 55  
network data 55
installation of JES3 exits 60
installing JES3  
maintenance implications 1
INTERCOM macro
   description 504
   restrictions 504
   internal text 61
IPCS program 354
IPURGE macro
   description 506
   restrictions 506
ISCAN1 macro
   description 507
   restrictions 507
ISCAN2 macro
   description 508
   restrictions 508
ISORT macro
   description 508
   restrictions 508
ITREAD macro
   description 509
   restrictions 509
ITWRITE macro
   description 510
   restrictions 510
IWASPOUT macro
   description 511
   restrictions 511

J
JCL (job control language) 61
   add statement parameters 120
   comment statement 103
   DD statement 104
   error messages 522
   EXEC statement 64, 103
   internal text 61
   JCL error messages 317
   JOB data to SMF record 96
   JOB statement 63, 96
   modification 120
   override JCL limit 115
   region parameter 448
   REGION parameter for JES3 195
   JCT (job control table) 9, 96
     JCT data space 327
     validation during a restart 9
   JCT access 327
     accessing the JCT 327
   JDS (job data set block) 512
     add a new entry 512
   JDS (job data set) 110
   JDSADD macro
     description 512
     restrictions 512
   JDSGET macro
     description 513
     restrictions 513
   JDSHOLD macro
     description 515
     restrictions 515
   JDSPPUT macro
     description 516
     restrictions 516
   JDSPUT macro
     description 517
     restrictions 517
   JDSREL macro
     description 518
     restrictions 518
   JES2 statement 103
   JES3 1
     common area 57
     dynamic support program 1
     execute a user-written program 2
     extended area 57
     introduction 2
   JES3 AWAIT reason code 585
   JES3 communication 61
     installation exits involved 61
   JES3 complex management 61
     JES3 functions (and exits) involved 61
     abnormal termination and recovery 61
     console services 61
     dump job 61
     general routines 61
     inquiry/modify 61
     utilities 61
   JES3 control statements 103
     //**DATASET 103
     //**ENDDATASET 103
   JES3 function 60
     mnemonic names 60
   JES3 functions 169
     coding macro instructions 169
     dump codes 177
     executable macro instructions 174
     generate control blocks 583, 584, 585, 587, 588,
       593, 595, 598, 602, 603
   JES3 initialization statements 34
     CIPARM 34, 96
     CONSOLE 79
     DEVICE 83, 85, 89, 596, 600
     DYNALDSN 102
     DYNALLOC 77
     error during JES3 initialization 511
     isolate field 507
     isolate subfield 508
     MAINPROC 42, 124
     MAXJOBST parameter on STANDARDS 115
     MSGROUTE 129
     OPTIONS 45
     read intermediate text 509
     SETPARAM 545
     STANDARDS 115
     use of ICARDRD macro 502
     use of ICONVBIN macro 503
     use of ICONVHEX macro 503
     use of INITMWLE macro 504
     use of INTERCOM macro 504
     write initialization messages 511
<table>
<thead>
<tr>
<th>JES3 macro functions</th>
<th>JES3 macro functions (continued)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABACKR 177</td>
<td>IATXCNDB 254</td>
</tr>
<tr>
<td>ABLOCK 178</td>
<td>IATXCNS 261</td>
</tr>
<tr>
<td>ABNCODE 180</td>
<td>IATXCNT 263</td>
</tr>
<tr>
<td>ABNCVDEC 181</td>
<td>IATXCRPL 266</td>
</tr>
<tr>
<td>ABNCVHEX 182</td>
<td>IATXCSF 267</td>
</tr>
<tr>
<td>ABNGET 182</td>
<td>IATXCUE 270</td>
</tr>
<tr>
<td>ABINPUT 184</td>
<td>IATXCVB 272</td>
</tr>
<tr>
<td>ABNVRFY 184</td>
<td>IATXCVD 273</td>
</tr>
<tr>
<td>ACALL 185</td>
<td>IATXDATX 275</td>
</tr>
<tr>
<td>ACLOSE 187</td>
<td>IATXDEI 276</td>
</tr>
<tr>
<td>ACVX 189</td>
<td>IATXDEQ 277</td>
</tr>
<tr>
<td>ADEBLOCK 189</td>
<td>IATXDEV 277</td>
</tr>
<tr>
<td>ADELETE 190</td>
<td>IATXDPL 279</td>
</tr>
<tr>
<td>ADEQ 192</td>
<td>IATXDIST 280</td>
</tr>
<tr>
<td>AENOQ 193</td>
<td>IATXDYH 281</td>
</tr>
<tr>
<td>AGETBUFF 194</td>
<td>IATXDT 282</td>
</tr>
<tr>
<td>AGETMAIN 195</td>
<td>IATXELA 283</td>
</tr>
<tr>
<td>AJOBNUM 196</td>
<td>IATXELD 284</td>
</tr>
<tr>
<td>ALOAD 199</td>
<td>IATXELS 285</td>
</tr>
<tr>
<td>ALOCATE 201</td>
<td>IATXENQ 287</td>
</tr>
<tr>
<td>ANOTE 202</td>
<td>IATXENT 288</td>
</tr>
<tr>
<td>AOPEN 203</td>
<td>IATXERCK 289</td>
</tr>
<tr>
<td>AOPEND 204</td>
<td>IATXERCVR 290</td>
</tr>
<tr>
<td>APOINT 206</td>
<td>IATXFIFO 291</td>
</tr>
<tr>
<td>AFURGE 207</td>
<td>IATXFLCB 293</td>
</tr>
<tr>
<td>APUTBUF 208</td>
<td>IATXFRO 294</td>
</tr>
<tr>
<td>APUTMAIN 209</td>
<td>IATXFSS 295</td>
</tr>
<tr>
<td>ARELEASE 210</td>
<td>IATXFVS 299</td>
</tr>
<tr>
<td>ARETURN 211</td>
<td>IATXFWSB 299</td>
</tr>
<tr>
<td>ATEST 211</td>
<td>IATXGCL 300</td>
</tr>
<tr>
<td>ATIME 213</td>
<td>IATXGENF 302</td>
</tr>
<tr>
<td>ATRACK 217</td>
<td>IATXGET 304</td>
</tr>
<tr>
<td>AWAIT 218</td>
<td>IATXGFC 305</td>
</tr>
<tr>
<td>AWRITE 220</td>
<td>IATXGFM 307</td>
</tr>
<tr>
<td>CONCNVRT 222</td>
<td>IATXGFM (list form) 309</td>
</tr>
<tr>
<td>CONREVRT 223</td>
<td>IATXGOSE 310</td>
</tr>
<tr>
<td>CONSPROF 224</td>
<td>IATXGRO 311</td>
</tr>
<tr>
<td>DEQMSG 224</td>
<td>IATXGSV 313</td>
</tr>
<tr>
<td>DEVSCAN 225</td>
<td>IATXHOME 314</td>
</tr>
<tr>
<td>DLOCOFF 226</td>
<td>IATXICA 315</td>
</tr>
<tr>
<td>DLOCON 227</td>
<td>IATXIOE 316</td>
</tr>
<tr>
<td>DSQLOC 228</td>
<td>IATXIOX 316</td>
</tr>
<tr>
<td>FAILDSP 229</td>
<td>IATXISMG 317</td>
</tr>
<tr>
<td>GETUNIT 230</td>
<td>IATXIWT 318</td>
</tr>
<tr>
<td>IATXADD 232</td>
<td>IATXJBNO 320</td>
</tr>
<tr>
<td>IATXAMDV 233</td>
<td>IATXJBTS 323</td>
</tr>
<tr>
<td>IATXRAL 234</td>
<td>IATXJCT 326</td>
</tr>
<tr>
<td>IATXARQ 235</td>
<td>IATXJCT macro 327</td>
</tr>
<tr>
<td>IATXASR 236</td>
<td>IATXJDS 328</td>
</tr>
<tr>
<td>IATXATDE 237</td>
<td>IATXJLOK 332</td>
</tr>
<tr>
<td>IATXATF 238</td>
<td>IATXJMR 334</td>
</tr>
<tr>
<td>IATXBDSN 238</td>
<td>IATXJOB 336</td>
</tr>
<tr>
<td>IATXBFM 239</td>
<td>IATXJQE 340</td>
</tr>
<tr>
<td>IATXBGM 240</td>
<td>IATXLOC 345</td>
</tr>
<tr>
<td>IATXBKIO 242</td>
<td>IATXLP 346</td>
</tr>
<tr>
<td>IATXBPL 244</td>
<td>IATXLPJ3 347</td>
</tr>
<tr>
<td>IATXBQN 247</td>
<td>IATXLRGT 349</td>
</tr>
<tr>
<td>IATXCDVE 248</td>
<td>IATXLRPT 350</td>
</tr>
<tr>
<td>IATXCIO 249</td>
<td>IATXMBE 351</td>
</tr>
<tr>
<td>IATXCKPT 251</td>
<td>IATXMGET 352</td>
</tr>
<tr>
<td>JES3 macro functions (continued)</td>
<td>JES3 macro functions (continued)</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td>IATXMID 354</td>
<td>IATXSPR 460</td>
</tr>
<tr>
<td>IATXMLWO 355</td>
<td>IATXSQE 461</td>
</tr>
<tr>
<td>IATXMMNX 357</td>
<td>IATXSRS 463</td>
</tr>
<tr>
<td>IATXMPC 358</td>
<td>IATXSTAT 463</td>
</tr>
<tr>
<td>IATXMDVA 361</td>
<td>IATXSTMD 469</td>
</tr>
<tr>
<td>IATXMXMM 362</td>
<td>IATXSTTA 471</td>
</tr>
<tr>
<td>IATXNTS 364</td>
<td>IATXTOD 475</td>
</tr>
<tr>
<td>IATXOSBM 366</td>
<td>IATXTRE 476</td>
</tr>
<tr>
<td>IATXOSCI 368</td>
<td>IATXTRG 478</td>
</tr>
<tr>
<td>IATXOSCO 369</td>
<td>IATXTRMT 478</td>
</tr>
<tr>
<td>IATXOSG 370</td>
<td>IATXUBAL 479</td>
</tr>
<tr>
<td>IATXOSIO 372</td>
<td>IATXVAL 480</td>
</tr>
<tr>
<td>IATXOSOO 373</td>
<td>IATXVFDB 481</td>
</tr>
<tr>
<td>IATXOSP 374</td>
<td>IATXVMSG 484</td>
</tr>
<tr>
<td>IATXOSPC 377</td>
<td>IATXVSR 485</td>
</tr>
<tr>
<td>IATXOSPM 378</td>
<td>IAXWLS 487</td>
</tr>
<tr>
<td>IATXOSSC 382</td>
<td>IATXWOPN 487</td>
</tr>
<tr>
<td>IATXOSUP 383</td>
<td>IATXWSR 489</td>
</tr>
<tr>
<td>IATXOSWS 385</td>
<td>IATXWSCL 490</td>
</tr>
<tr>
<td>IATXPDO 387</td>
<td>IAYASM 583</td>
</tr>
<tr>
<td>IATXPGXM 391</td>
<td>IATXAWD 584</td>
</tr>
<tr>
<td>IATXPHEX 392</td>
<td>IATXAWR 585</td>
</tr>
<tr>
<td>IATXPJ3 393</td>
<td>IATYCNS 587</td>
</tr>
<tr>
<td>IATXPOSE 394</td>
<td>IATYDJF 588</td>
</tr>
<tr>
<td>IATXPOST 395</td>
<td>IATYDJR 593</td>
</tr>
<tr>
<td>IATXPRRES 396</td>
<td>IATYDSD 595</td>
</tr>
<tr>
<td>IATXPRMD 397</td>
<td>IATYFCD 598</td>
</tr>
<tr>
<td>IATXPRT 400</td>
<td>IATYMOD 602</td>
</tr>
<tr>
<td>IATXPSC 402</td>
<td>IATYRSC 603</td>
</tr>
<tr>
<td>IATXPTCH 403</td>
<td>IATYSRF 603</td>
</tr>
<tr>
<td>IATXPTCK 404</td>
<td>IATZCALL 491</td>
</tr>
<tr>
<td>IATXRABC 405</td>
<td>IATZHEX 492</td>
</tr>
<tr>
<td>IATXRABD 406</td>
<td>IATZLOAD 493</td>
</tr>
<tr>
<td>IATXRABP 407</td>
<td>IATZMEX 494</td>
</tr>
<tr>
<td>IATXRCL 408</td>
<td>IATZMNOR 495</td>
</tr>
<tr>
<td>IATXRCLVL 409</td>
<td>IATZMOVE 496</td>
</tr>
<tr>
<td>IATXRELC 412</td>
<td>IATZPARM 497</td>
</tr>
<tr>
<td>IATXRMP 413</td>
<td>IATZTYPE 502</td>
</tr>
<tr>
<td>IATXRST 414</td>
<td>ICARDRD 502</td>
</tr>
<tr>
<td>IATXRUGT 415</td>
<td>ICONVBIN 503</td>
</tr>
<tr>
<td>IATXRUPT 416</td>
<td>ICONVHEX 503</td>
</tr>
<tr>
<td>IATXSAS 417</td>
<td>INITMWL 504</td>
</tr>
<tr>
<td>IATXSCH 419</td>
<td>INTERCOM 504</td>
</tr>
<tr>
<td>IATXSCN 421</td>
<td>IPURGE 506</td>
</tr>
<tr>
<td>IATXSCN2 423</td>
<td>ISCAN1 507</td>
</tr>
<tr>
<td>IATXSEC 429</td>
<td>ISCAN2 508</td>
</tr>
<tr>
<td>IATXSIO 439</td>
<td>ISORT 508</td>
</tr>
<tr>
<td>IATXSMF 440</td>
<td>ITREAD 509</td>
</tr>
<tr>
<td>IATXSMGR 441</td>
<td>ITWRITE 510</td>
</tr>
<tr>
<td>IATXSNFG 442</td>
<td>IWASPOUT 511</td>
</tr>
<tr>
<td>IATXSNFM 443</td>
<td>JDSADD 512</td>
</tr>
<tr>
<td>IATXSNFS 445</td>
<td>JDSGET 513</td>
</tr>
<tr>
<td>IATXSNGM 448</td>
<td>JDSHOLD 515</td>
</tr>
<tr>
<td>IATXSNLK 450</td>
<td>JDSPOINT 516</td>
</tr>
<tr>
<td>IATXSNM 454</td>
<td>JDSPUT 517</td>
</tr>
<tr>
<td>IATXSNS 455</td>
<td>JDSREL 518</td>
</tr>
<tr>
<td>IATXSNT 458</td>
<td>JESCKPNT 520</td>
</tr>
<tr>
<td>IATXSNTR 459</td>
<td>JESCCLOSE 520</td>
</tr>
</tbody>
</table>
JES3 macro functions

JESEXCP 521
JESMSG 522
JESOPEN 524
JESREAD 525
JESTAE 527
JNADD 528
JNCBHLD 529
JNCBREL 529
JNDEL 530
JNGET 530
JSERV 531
LOGIN 533
LOGOUT 534
MDSALLOC 535
MDSDSN 537
MDSERQ 538
MDSID 539
MDSJERR 540
MDSJGET 541
MDSJST 542
MDSMSG 544
MDSVGET 546
MDSVML 547
MESSAGE 556, 557
MESSAGE (standard form) 548
MNTRDEF 558
MOVEDATA 561
MTBL 562
NCADD 566
NCKDEL 567
NCNTAADD 563
NCNTAFND 563
NCNTAGET 564
NCNTAPUT 565
NCNTAREL 566
PURCHAIN 567
PURUNIT 569
RQTADD 569
RQTADEL 567
RQTAGEN 571
RQTAPUT 572
SPINOFF 573
SSISERV 574
SUPFLAG 578
VIOLATE 579
WRITEOSE 579
WRTCCHAIN 581
ZEROCODE 583

JES3 networking 54

JES3 purge processing 94

JES3 services

calling 21

JES3 trace table 30

trace table 30

JES3OUT data set 43, 78

JESCKPNT macro
description 520
restrictions 520

JESCLOSE macro
description 520
JOB statement
  job control language (JCL)  63
job status 70
  JST (job summary table)  70
  JVT (job volume table) 70, 546
job step 543
  step positioning 543
job-related JES3 function 61
  job input and management 61
  job output and termination 61
  job resource management 61
  job scheduling management 61
JQE (job queue element)  340
  obtain 340
JSAM (JES3 spool access method) 32
  allocate spool space 407
JSAM buffer 32
JSAM buffers 57
JSERV macro
  description 531
  restrictions 531
JSS (job segment scheduler)
  examine final job status 70
  job segment scheduler 23
  load a CSECT 26
  return code 25
  return from DSP 25
JST (job summary table)
  access a JST 541
  job status 70
JVT (job volume table)
  job status 70

K
  keyboard 631

L
  LCB (logical unit control block) 293
    IATXFLCB macro (find LCB) 293
    IATXNGRS macro (negative response) 364
 linkage routine (ASAVE) 26
  save area 26
LOC= parameter 21, 30, 195
  AGETMAIN macro 195, 448
local
  address space 58
locate
  inhibit printing 73
  JESMSG data set 73
  request 73
  response message 73
lock 31
  system lock 31
logical output printer 112
LOGIN macro 22, 23, 35
  description 533
  restrictions 533
LOGOUT macro
  description 534
LOGOUT macro (continued)
  restrictions 534
  lost transaction in BDT 367

M
  macro 3
    ADEQ 31
    AENOQ 31
    ATEST 31
    AWRITE 32
    CONCNVRT 550
    DEQMSG 35, 551
    IATDEVA 49
    IATDEVSC 49, 50
    IATXAMDV 231
    IATXBFM 30
    IATXBGM 30
    IATXCNV 24
    IATXCSF 27
    IATXISMG 317
    IATXIWT 64, 71
    IATXJOB 196
    IATXPRMD 397
    IATXSDM 31
    IATXTRC 30
    IATYASV 20
    IATYDSD 10
    IATYFCT 22
    IATYTVT 22
    INTERCOM 79
    JDSGET 24
    JESCLOSE 29
    JESEXCP 29
    JESOPEN 29
    JESREAD 32
    library 3
    LOGIN 35
    LOGOUT 35
    PURCHASE 32
    SWAREQ 34
    syntax conventions 169
    SYS1.MACLIB macro library 3
    WRTCHAIN 31, 32
    WTO 41
mainframe
  education xv
map data areas 24
IATYASV 583
IATYAWDR 584
IATYAWR 585
IATYCNV 587
IATYDFJ 588
IATYDJR 593
IATYDSD 24, 318, 595
IATYDSQ 229
IATYDST 227
IATYDUM 229
IATYDVE 487
IATYELB 283
IATYFCD 598
map data areas (continued)
IATYFSCB 477
IATYICT 116
IATYIDD 64
IATYIIW 93
IATYLRS 67, 68
IATYMDS 537, 539, 544
IATYMOD 602
IATYMPC 532
IATYNBF 106
IATYNJH 118
IATYNJY 364
IATYOSB 367
IATYRSC 31, 603
IATYRSQ 570
IATYSOR 509
IATYSPP 460
IATYSRD 455
IATYSRF 603
IATYSRL 121, 509
IATYSRS 147
IATYSUP 578
IATYTSWK 100
IATYVSR 94
IATYWSP 386
IEFCNPRM 62
MCS (multiple console support) 36
descriptor code 36
routing code 552
ROUT= parameter on MESSAGE macro 552
MDS (main device scheduler) 61
allocate a resource 535
create MDS trace entry 539
error queue 146
installation exit 61 146
installation exit 62 148
installation exit 63 150
installation exits involved 61
MDS error queue 538
MDSERRQ macro 538
operator intervention 146
output messages for MDS 544
validity of tape mount 148
MDSALLOC macro
description 535
restrictions 535
MDSDSN macro
description 537
restrictions 537
MDSERRQ macro
description 538
restrictions 538
MDSID macro
description 539
restrictions 539
MDSJERR macro
description 540
restrictions 540
MDSJGET macro
description 541
MDSJGET macro (continued)
restrictions 541
MDSJST macro
description 542
restrictions 542
MDSMSG macro
description 544
restrictions 544
MDSVGET macro
description 546
restrictions 546
MDSVLMMacro
description 547
restrictions 547
message 16
ACTION parameter 36
algorithm for routing codes 41
broadcast messages 550
choose the appropriate macro 42
dequeue messages 548
DEQMSG.macroue messages 548
MESSAGEmacroue messages 548
message 16
DEST parameter 36
destination code 36
generate 71
IAT3012 77
IAT3020 16, 26, 58
IAT3102 16, 26, 58
IAT3673 247
IAT4401 73
IAT4402 73
IAT4405 73
IAT5310 150
IAT6308 201
IAT7130 35
IAT9154 117
IATXMID macro 354
MESSAGEmacro 36
multiline messages 43
operator message 36
routing message 36, 41
simulate operator message 504
time stamp a message 522
to operator 72, 129
write messages during initialization 511
write to JESMSG data set 65
MESSAGEmacro
description 556, 557
restrictions 556, 557
MESSAGE macro (standard form)
description 548
restrictions 548
message processing 37
installation exits 37
MVS facilities 37
routing code 37
WTO/WTOR 39
mixed-format field 336
convert a job number 336
MNTRDEF macro
description 558
MNTRDEF macro (continued)
restrictions 558
MODIFY command 106
modifying a DEVICE statement 50
IATDEV macro 49
module identification 20
module loading 21
dynamic subprogram loading 25
IATYMOD macro 21
organize a DSP 25
use of ADELETE 25
use of ALOAD 25
module name 26
IATAUX 45
IATCNIA 79
IATCNNJ 106
IATCNWO 548
IATGRCD 94
IATGRJR 26, 70
IATGRJX 344
IATGRQC 245
IATGRSV 26
IATICM 63, 64, 65
IATIXX 61
IATIIDR 70
IATIMIS 71
IATIPN 69
IATIPO 67, 73
IATIIPR 63, 64, 65
IATIIST 115
IATINGS 76
IATINRN 76
IATISDT 103
IATISEN 96
IATISJB 78
IATISLG 103, 104
IATISNJ 114
IATNHT 114
IATNTJS 107, 155
IATNTSF 110, 116
IATODNJ 113
IATOSBM 367
IATOSDO 81
IATOSFG 121
IATOSNJ 112
IATOSP 82
IATOSPR 82
IATOSSN 82
IATPURG 94
IATSICA 282
IATSWO 129
IATSNDC 490
IATUTIS 77
IAYIDD 116
MOVEDATA macro
description 561
restrictions 561
MPF (message processing facility) 36
console messages in color 36, 41
MTBL macro
description 562
node table 364
NORMAL parameter
  general description 177
nucleus
  FSS (IATNUCF) 5
  JES3 (IATNUC) 5
NUCTASK mode 469, 475

O
  obtain
    JOE (job queue element) 340
operator command 22
  command 22
  MODIFY 125
operator message 36
operators 34
  console appendage 34
  console service function 34
  sending messages 34
OSB (output service BDT control block) 367
OSE (output scheduling element) 113
  data set section 113
output 82
  data set 112
  data set deletion 125
  data set header 85
  job header for output 82
  output for a purged job 125
  SYSOUT class 85
output service 81
  close output device 369
IATXOSWS macro 385
  modify an OSE 81
open an output device 373
OSE scheduling matrix 311
restart chain 390
  trailer record 369
writer requests 385
output writer FSS 53
  data set header 121
  installation exit 45 121
  installation exits 53
  obtain a data set 121
  output writer FSA 53

P
  page-fix 391
  IATXPGX macro 391
    in auxiliary address space 391
  parameter field 172
  parameter variables 171
  patch area
    generate a patch area 403
performance
  degradation 254
  measuring 281, 282
  MVS WAIT macro 27
  reduce I/O for the JDS 518

performance (continued)
  system degradation 27
prescan phase of C/I 63
presentation services function 363
print data 351
  carriage control character 351
private area 49, 461
  C/I FSS address space 49
  move data (IATXMOV macro) 362
  SQE (storage management queue) 461
PROCESS job control statement ("/PROCESS) 600
  process mode 397
  define process mode table 397
  processor selection 123
  override the processor selection 126
  select a processor for C/I 123
programming convention 19
  installation 19
  write a DSP 19
protection
  DSP 5
  installation exit 5
  protection key
    VIOLATE macro 579
PTF number 21
PURCHAIN macro
  description 567
  restrictions 567
purge processing 94
  avoid dummy SMF 26 records 94
PURUNIT macro
  description 569
  restrictions 569

Q
  QISAM (queued indexed sequential access method) 27
  QSAM (queued sequential access method) 27

R
  RACF (resource access control facility) 131
    JES3 request 607
    perform security 131
    security decision 154
    security decisions 607
recovery 527
  JESTAE macro 527
  setting up a recovery environment 527
reentrant linkage 57
  IATXCU macro 57
  IATYUXL macro 57
  installation exit 270
REGION parameter 195
  amount available on JCL 195
register 22
  conventions 173
  restrictions 173
  saving 173
  usage 22
register (continued)
usage conventions 23
registry status 366
remote processing 11
   installation exits involved 61
rerouting job 54
   send jobs to other nodes 54
restart job 74, 146
return code 25
   JSS 25
RJP (remote job processing) 79
RMODE (residency mode) 20
   RMODE statement 20
routine 11
   installation exit 16
   programming considerations 12
   restart portion 12
   validation portion 11
routing code 129
   algorithm for selecting one 41
   convert to destination class 129
   installation exit 57, 129
   MCS (multiple console support), routing codes 39
routing message 36
RPL (request parameter list)
   reset a VTAM RPL 266
RQTADD macro
   description 569
   restrictions 569
RQTADEL macro
   description 570
   restrictions 570
RQTAGEN macro
   description 571
   restrictions 571
RQTAPUT macro
   description 572
   restrictions 572
rules 173
   address mode 173
   register usage 173

SAF (security authorization facility) 131
   communicate 607
   controlled use of operator commands 138
   controlling network data 138
   controlling network jobs 138
IATGRSC 131
IATXSEC 429
   information for JES3 security calls 607
JES3 131, 138
   RACROUTE request 607
   restricted access SYSIN/SYSOUT data sets 138
save area 26
   example of ACALL and ARETURN 26
   IATGRSV module 26
   linkage routine 26
   return save area to pool 299
save area (continued)
   save a register 173
   user save area (IIWUSAVE) 93
   scan initialization statement 76
scheduler
   control block 93
   converter/interpreter service 93
   scheduler element 10
      validate 10
security 5
   installation DSPs 5
   installation exits 5
security calls
   types of 6
SEL (service entrance list) 282
separator page 82
SEND macro mapping options 351
   cards 85, 88
   data set header label 85
   data set header page 85
   installation exit 45, 121
   job header page 82
   job trailer information 88
   job trailer page 88
   JSPA (job separator page area) 121
setup job 69
   job setup 69
shortcut keys 631
SJF (scheduler JCL facility) 84
SMF (system management facilities) 440
   dummy SMF 26 record 94
   JOB statement information 96
   user field 63, 64
   write an SMF record 440
SMP (system modification program) 3
   example of SMP 4
   use SMP to code an installation exit 3
SMP/E (system modification program extended) 60
SMS (storage management system) 104
   DATACLAS keyword 104
   MGMTCLAS keyword 104
   STORCLAS keyword 104
SNA (systems network architecture) 61
   enqueue block of data 287
   identify SNA RJP entry point 288
   installation exits involved 61
   networking 61
   open a session for SNA RJP 487
   print to a SNA workstation 350
   SNA device entry table 248
   SNARJP CSECT 267
SNA/NJE 152
   job priority 152
   systems network architecture network job entry 152
   sorting table entries 508
SPINOFF macro
   description 573
   restrictions 573
spool data management 61
installation exits involved 61
spool data set 31
access spool under a subtask 31
recovering spool space 506
STT (single track table) allocation 471
write a single record file 220
write an OSE 334, 579
spool-resident control block 31
access a spool control block 31
map SRF header 603
routine reads record 31
serialize 31
shared access 32
write JSAM buffer 31
SRDAMPR work area 455
SRL (service request list) 121
for output writer FSS 121
SSISERV macro
description 574
restrictions 574
SSVT (subsystem vector table) 282
step positioning 543
storage 21
clear storage using ZEROCORE macro 583
four macros manage JES3 30
LOC= parameter 21, 30
manage storage 30
obtain contiguous storage 195
obtain storage in subpool 240
protection key 579
release 209
storage protection violation 579
system mask 579
storage management 30
storage management subsystem 63
ACS exit 63
converting C/I internal text 63
DD statements 66
high watermark setup 69
JST entries 71
storage queue element 461
add or delete entry 461
IATXSE macro 461
storage management queue 461
store-and-forward function 110
networking 110
STT (single track table) 221, 580
subpool 30
subpool 231 30, 240
subpool 241 30, 240
subpool 245 30, 240
subtask 27, 31
access spool under a subtask 31
establish a subtask environment 27
general subtask 267
index 269
JES3 116
specific subtask 267
SUPFLAG macro
description 578
SUPUNITS table 83
check for job header 83
reset flags 578
support a unit 231
SUSPEND parameter
send SYSOUT data 369
SVF sequence 350
SVT (system vector table) 413
SVTRMVT field 413
SWA (scheduler work area) 33
CNPRSWA bit in CNPRPARM 62
IDDSWA field in IATYIDD 63, 64, 66
JCL interpretation 33
manage SWA block 33
move SWA block above 16 megabytes 62, 63, 64, 66, 93
virtual storage 64
write SWA block to spool 33
SWAREQ macro 34
synchronize
JES3 function 28
syntax diagrams
how to read 169
SYSOUT (system output stream) 109
class 109
data set 109
system failure 74
restart a job 74
system lock 25
release 25
system output 109
SYSOUT (system output stream) 109
system performance 27
SYSUNIT table object services
using IATXSYSU macro 472
T
table reference techniques 27
TAT (track allocation table) 217
for spool activity 217
termination
abnormal 21, 30
JESTAE macro 21
time 282
measure elapsed time 282
obtain time of day 476
take a time stamp 282
timer
appendage 213
queue element 213
routine 213
trace table 30
create a trace entry 30, 476
IATXTRC macro 30
standard header 30
transfer vector table 22
default subpool number 196, 444, 449
define the TVT 22
transfer vector table  (continued)
  IATYTVT macro 22
TSO (time sharing option) 99
  security reasons 145
  terminal commands 99
  TSO/E userid flag 100
TSO command 99
  CANCEL 99
  OUTPUT 99
  STATUS 99
TVT (transfer vector table) 22

U
  unit 67
    unit count 69
    unit type 67
USAM (user spool access method) 407
    allocate spool space 407
user save area 62
user-written program 2
    execute 2

V
  validate control block 74
    FRP and OSE 75
    IBM control block 74
    JCT, JDAB and JMR 75
    JDS 75
    JST, JVT and DJST 75
    PDB 75
validation 8
    control block 9
    restart process 8
    restart routine 8
    use a scheduler element 9
    use of some fields 13
    user control block 8
verify parameters
    using IATZPCHK macro 497
VIOLATE macro
    description 579
    restrictions 579
volume 67
    volume serial number 67
    wrong volume serial number 92
volume label 91
VTAM (virtual telecommunications access method) 289, 456, 479
    CHECK macro 289
    CLSDST macro 456, 479

W
  WAIT macro 21
    generalized subtask facility 21
    slow performance 27
    use a JES3 subtask 267
WRITEOSE macro
    description 579
Readers’ Comments — We’d Like to Hear from You

z/OS
JES3 Customization

Publication No. SA22-7542-08

We appreciate your comments about this publication. Please comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. The comments you send should pertain to only the information in this manual or product and the way in which the information is presented.

For technical questions and information about products and prices, please contact your IBM branch office, your IBM business partner, or your 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. IBM or any other organizations will only use the personal information that you supply to contact you about the issues that you state on this form.

Comments:

Thank you for your support.
Submit your comments using one of these channels:
• Send your comments to the address on the reverse side of this form.
• Send your comments via e-mail to: mhvrcfs@us.ibm.com

If you would like a response from IBM, please fill in the following information:

Name
Address

Company or Organization

Phone No. E-mail address
Program Number: 5694-A01

Printed in USA