

z/OS



# MVS Programming: Workload Management Services



z/OS



# MVS Programming: Workload Management Services

**Note**

Before using this information and the product it supports, be sure to read the general information under Appendix E, "Notices," on page 905.

**Fifteenth Edition, April 2007**

This edition applies to Version 1 Release 8 of z/OS (5694-A01), and to all subsequent releases and modifications until otherwise indicated in new editions.

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

IBM welcomes your comments. A form for readers' comments may be provided at the back of this publication, or you may address your comments to the following address:

IBM Deutschland Entwicklung GmbH  
Department 3248  
Schoenaicher Strasse 220  
D-71032 Boeblingen  
Federal Republic of Germany

FAX (Germany): 07031+16-3456

FAX (Other Countries): (+49)+7031-16-3456

Internet e-mail: [s390id@de.ibm.com](mailto:s390id@de.ibm.com)

World Wide Web: <http://www.ibm.com/servers/eserver/zseries/zos/webqs.html>

If you would like a reply, be sure to include your name, address, telephone number, or FAX number.

Make sure to include the following in your comment or note:

- Title and order number of this book
- Page number or topic related to your comment

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

© **Copyright International Business Machines Corporation 1988, 2007. All rights reserved.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

---

# Contents

<b>Figures</b> . . . . .	xxvii
<b>Tables</b> . . . . .	xxix
<b>About This Document</b> . . . . .	xxxi
Who Should Use This Document . . . . .	xxxi
Where to Find More Information . . . . .	xxxii
The WLM/SRM Web Page . . . . .	xxxii
Using LookAt to look up message explanations . . . . .	xxxii
Using IBM Health Checker for z/OS . . . . .	xxxii
How to Read a Syntax Diagram . . . . .	xxxiii
How to read syntax diagrams. . . . .	xxxiii
<b>Summary of Changes</b> . . . . .	xxxvii

---

## Part 1. Using the Workload Management Services . . . . . 1

<b>Chapter 1. Introducing the Workload Management Services</b> . . . . .	3
Services for Subsystem Work Managers . . . . .	3
Why Use the Work Manager Services . . . . .	3
Why Use the Execution Delay Monitoring Services . . . . .	4
Why Use the Enclave Services . . . . .	6
Why Use the Queueing Manager Services . . . . .	8
Why Use the Routing Manager Services . . . . .	9
Why Use the Scheduling Environment Services . . . . .	10
Why Use the Sysplex Routing Services . . . . .	10
Why Use the Query System Information Service . . . . .	11
Services for Performance Monitors . . . . .	12
Why Use the Workload Reporting Services . . . . .	12
Getting Information from SMF Type 99 . . . . .	13
Services for Application Programs . . . . .	14
WLM Services Supporting 64-bit Address Space . . . . .	14
<b>Chapter 2. Using The Subsystem Work Manager Services</b> . . . . .	17
Considerations Before Using the Services . . . . .	17
Suggested Services for a Single Address Space Transaction Manager . . . . .	18
Using the Execution Delay Monitoring Services . . . . .	20
Suggested Services for a Work Manager Calling a Data Manager . . . . .	22
Services for Multiple Address Space Work Managers . . . . .	23
Execution Delay Monitoring Services for Multiple Address Space Work Managers . . . . .	25
Services for Work Managers that Distribute Work Requests . . . . .	27
Determining the Subsystem Name and Type . . . . .	27
Using IWMWMCON When Distributing Work in a Sysplex . . . . .	27
<b>Chapter 3. Creating and Using Enclaves</b> . . . . .	31
Why Would You Use an Enclave? . . . . .	31
SRBs in Enclaves . . . . .	32
Tasks in Enclaves . . . . .	32
Comparison of Enclaves and Execution Delay Services . . . . .	33
Creating an Enclave . . . . .	33
Independent Versus Dependent Enclaves . . . . .	33
Registering an Enclave . . . . .	34

Multisystem Enclaves . . . . .	35
Scheduling an SRB in an Independent Enclave . . . . .	36
Joining Tasks to an Independent Enclave . . . . .	38
Using Dependent Enclaves . . . . .	40
Using a Multisystem Enclave . . . . .	42
Performance Management of Address Spaces with Enclaves . . . . .	45
Using ENQ/DEQ or Latch Manager Services with Enclaves . . . . .	45
Enclave Resource Accounting . . . . .	46
Managing the Performance of Work in Enclaves . . . . .	48
Using Independent Enclaves . . . . .	48
Using Dependent Enclaves . . . . .	49
Querying an Enclave's Classification Information . . . . .	49
Querying a Dispatchable Unit's Enclave Status . . . . .	49
Deleting an Enclave . . . . .	49
<b>Chapter 4. Participating in Enterprise Workload Management . . . . .</b>	<b>51</b>
Enclave Services and EWLM. . . . .	52
Modelling Your Business Transactions . . . . .	52
Connecting with WLM as an EWLM-Participant . . . . .	53
Disconnecting from WLM . . . . .	54
Creating an Enclave . . . . .	54
Deleting an Enclave . . . . .	54
Classifying Work Requests . . . . .	54
Starting and Stopping Work Requests Explicitly . . . . .	56
Continuing a Work Request at Another Application . . . . .	56
Blocking/Unblocking Work Requests . . . . .	56
Enclave Services and the ARM API . . . . .	56
Instrumenting a C Application for ARM . . . . .	57
Compiling an ARM-instrumented Application . . . . .	57
Binding an ARM-instrumented Application . . . . .	58
Running an ARM-Instrumented Application. . . . .	59
<b>Chapter 5. Using the Queueing Manager Services . . . . .</b>	<b>61</b>
Example of Using the Queueing Manager Services . . . . .	61
Managing the Number of Server Instances per Server Address Space . . . . .	67
Directing Work Requests to a Specific Server Region. . . . .	68
Updating a Service Definition with Application Environment Information . . . . .	69
Using the Queueing Manager Connect Exit . . . . .	69
Exit Routine Environment . . . . .	70
Register Usage. . . . .	70
Restrictions . . . . .	71
<b>Chapter 6. Using the Routing Manager Services . . . . .</b>	<b>73</b>
A Routing Manager Model. . . . .	73
Using the Routing Server Connect Exit . . . . .	77
Exit Routine Environment . . . . .	77
Register Usage. . . . .	78
<b>Chapter 7. Using the Scheduling Environment Services . . . . .</b>	<b>79</b>
Obtaining Scheduling Environment Definitions . . . . .	79
Manipulating Resource State Settings . . . . .	80
A Model Work Flow . . . . .	82
<b>Chapter 8. Using the Sysplex Routing Services . . . . .</b>	<b>85</b>
Why Use the Sysplex Routing Services? . . . . .	85
When to Use the Sysplex Routing Services . . . . .	85

Registering as an Eligible Server . . . . .	86
Determining Where to Route Work . . . . .	86
Deregistering as an Eligible Server . . . . .	87
Example of Using the Sysplex Routing Services. . . . .	87
WLM Sysplex Workload Distribution . . . . .	88
Calculation of server weights. . . . .	89
Example . . . . .	89
<b>Chapter 9. Using the Workload Reporting Services . . . . .</b>	<b>91</b>
When to Use the Workload Reporting Services . . . . .	91
Using ENF Signals to Guide Data Collection . . . . .	91
ENF Event Code 41 . . . . .	92
Using the IWMRCOLL Service . . . . .	93
Using the Information in IWMWRCAA . . . . .	93
Using the Subsystem Work Manager Delay State Information. . . . .	94
Using Delay States to Report Subsystem Interactions . . . . .	96
Using the Response Time Information . . . . .	98
Interpreting Report Class Data . . . . .	99
Using the IWMRQRY Service . . . . .	100
<b>Chapter 10. Using the Administrative Application Services . . . . .</b>	<b>103</b>
Installing a Service Definition . . . . .	103
Mapping a Service Definition . . . . .	103
Adding Program-Specific Extensions to a Service Definition . . . . .	103
Checking a Service Definition Using IWMDINST . . . . .	106
Recommended Validity Checking. . . . .	106
Preventing Service Definition Overlays. . . . .	107
Example of Using IWMDINST to Install a Service Definition . . . . .	108
Extracting a Service Definition. . . . .	109
Example of Using IWMDEXTR to Extract a Service Definition . . . . .	109
Activating a Service Policy . . . . .	109
Example of Activating a Policy using IWMPACT . . . . .	110
Querying the Active Classification Rules . . . . .	110
Example of IWMCQRY . . . . .	111
<b>Chapter 11. Using SMF Record Type 99 . . . . .</b>	<b>113</b>
When to Start SMF Record Type 99. . . . .	114
Starting SMF Record Type 99 . . . . .	114
Identifying Work in SMF Type 99 Records . . . . .	114
Identifying Server Service Classes . . . . .	114
Identifying Internal Service Classes . . . . .	115
Interpreting Trace Table Entries . . . . .	115
Policy Adjustment . . . . .	116
Resource Adjustment . . . . .	116
Receivers and Donors. . . . .	116
Performance Index . . . . .	116
Receiver Value . . . . .	117
Net Value . . . . .	117
Small Processor Consumer . . . . .	117
Storage Housekeeping . . . . .	117
Reverse Housekeeping . . . . .	118
Working Set Management . . . . .	118
Interpreting Management Policy Data . . . . .	118
Dispatching Priority . . . . .	118
MPL Targets . . . . .	119
Swap Protect Time . . . . .	119

Expanded Storage Policies . . . . .	119
Storage Targets . . . . .	120
Cap Slices . . . . .	120
I/O Priority . . . . .	120
Number of Server Address Spaces . . . . .	121
Interpreting Plots. . . . .	121
System Paging Delay Plot . . . . .	121
Period MPL Delay Plot . . . . .	122
Period Ready User Average Plot . . . . .	122
Period Swap Delay Plot . . . . .	122
Period Paging Rate Plot . . . . .	123
Period Proportional Aggregate Speed Plot . . . . .	123
I/O Delay Plot . . . . .	123
Queue Delay Plot . . . . .	124
Address Space Paging Plots . . . . .	124
I/O Velocity Plot . . . . .	125
Interpreting Priority Table Data. . . . .	126
Interpreting Lack of Action . . . . .	126
Examples of Interpreting SMF Record Type 99. . . . .	127
Action Trace Example . . . . .	127
MPL Policy Example . . . . .	129

---

**Part 2. Reference: Workload Management Services . . . . . 133**

<b>Chapter 12. IWMCLSFY – Classify Work . . . . .</b>	<b>135</b>
Environment . . . . .	135
Programming Requirements . . . . .	135
Restrictions. . . . .	135
Input Register Information . . . . .	136
Output Register Information. . . . .	136
Performance Implications . . . . .	136
Syntax . . . . .	136
Parameters. . . . .	138
ABEND Codes . . . . .	147
Return and Reason Codes . . . . .	147
Examples . . . . .	149
<b>Chapter 13. IWMCNTN — WLM Contention Notification . . . . .</b>	<b>151</b>
Environment . . . . .	151
Programming Requirements . . . . .	151
Restrictions. . . . .	151
Input Register Information . . . . .	151
Output Register Information. . . . .	152
Performance Implications . . . . .	152
Syntax . . . . .	152
Parameters. . . . .	153
ABEND Codes . . . . .	157
Return and Reason Codes . . . . .	157
Example . . . . .	158
<b>Chapter 14. IWMCPAFN – WLM CPU Affinity Service . . . . .</b>	<b>161</b>
Environment . . . . .	161
Programming Requirements . . . . .	162
Restrictions. . . . .	162
Input Register Information . . . . .	162
Output Register Information. . . . .	162



Performance Implications . . . . .	162
Syntax . . . . .	162
Parameters . . . . .	163
ABEND Codes . . . . .	165
Return and Reason Codes . . . . .	165
Example . . . . .	167
<b>Chapter 15. IWMCQRY – Query Classification Attributes . . . . .</b>	<b>169</b>
Environment . . . . .	169
Programming Requirements . . . . .	170
Restrictions. . . . .	170
Input Register Information . . . . .	170
Output Register Information. . . . .	170
Performance Implications . . . . .	171
Syntax . . . . .	171
Parameters . . . . .	171
ABEND Codes . . . . .	173
Return and Reason Codes . . . . .	173
<b>Chapter 16. IWMDXTR – Extract Service Definition . . . . .</b>	<b>177</b>
Environment . . . . .	178
Programming Requirements . . . . .	178
Restrictions. . . . .	178
Input Register Information . . . . .	178
Output Register Information. . . . .	179
Performance Implications . . . . .	179
Syntax . . . . .	179
Parameters . . . . .	179
ABEND Codes . . . . .	181
Return and Reason Codes . . . . .	181
<b>Chapter 17. IWMDINST – Install Service Definition . . . . .</b>	<b>185</b>
Environment . . . . .	186
Programming Requirements . . . . .	186
Restrictions. . . . .	186
Input Register Information . . . . .	187
Output Register Information. . . . .	187
Performance Implications . . . . .	187
Syntax . . . . .	187
Parameters . . . . .	188
ABEND Codes . . . . .	191
Return and Reason Codes . . . . .	191
<b>Chapter 18. IWMEBLK – Work Request Blocked . . . . .</b>	<b>195</b>
Environment . . . . .	195
Programming Requirements . . . . .	195
Restrictions. . . . .	196
Input Register Information . . . . .	196
Output Register Information. . . . .	196
Performance Implications . . . . .	196
Syntax . . . . .	196
Parameters . . . . .	197
ABEND Codes . . . . .	199
Return and Reason Codes . . . . .	199
Examples . . . . .	201

<b>Chapter 19. IWMECQRY – Query Enclave Classification Attributes</b>	203
Environment	203
Programming Requirements	203
Restrictions.	203
Input Register Information	204
Output Register Information.	204
Performance Implications	204
Syntax	205
Parameters.	205
ABEND Codes	207
Return and Reason Codes	207
<b>Chapter 20. IWMEDREG — WLM Enclave Deregister Service</b>	209
Environment	209
Programming Requirements	209
Restrictions.	209
Input Register Information	209
Output Register Information.	210
Performance Implications	210
Syntax	210
Parameters.	210
ABEND Codes	212
Return and Reason Codes	212
Example	213
<b>Chapter 21. IWMEGCOR – Retrieving a Correlator</b>	215
Environment	215
Programming Requirements	215
Restrictions.	216
Input Register Information	216
Output Register Information.	216
Performance Implications	216
Syntax	216
Parameters.	217
ABEND Codes	219
Return and Reason Codes	219
Examples	221
<b>Chapter 22. IWMEJOIN – Join an Enclave</b>	223
Environment	223
Programming Requirements	223
Restrictions.	224
Input Register Information	224
Output Register Information.	224
Performance Implications	224
Syntax	224
Parameters.	225
ABEND Codes	226
Return and Reason Codes	226
Example	228
<b>Chapter 23. IWMELEAV – Leave an Enclave</b>	229
Environment	229
Programming Requirements	229
Restrictions.	230
Input Register Information	230

Output Register Information . . . . .	230
Performance Implications . . . . .	230
Syntax . . . . .	230
Parameters . . . . .	230
ABEND Codes . . . . .	232
Return and Reason Codes . . . . .	232
Example . . . . .	234
<b>Chapter 24. IWMEQTME – Enclave CPU Time Query . . . . .</b>	<b>235</b>
Environment . . . . .	235
Programming Requirements . . . . .	235
Restrictions. . . . .	235
Input Register Information . . . . .	235
Output Register Information. . . . .	235
Performance Implications . . . . .	236
Syntax . . . . .	236
Parameters . . . . .	236
ABEND Codes . . . . .	238
Return and Reason Codes . . . . .	238
<b>Chapter 25. IWMEREG — WLM Enclave Register Service . . . . .</b>	<b>241</b>
Environment . . . . .	241
Programming Requirements . . . . .	241
Restrictions. . . . .	241
Input Register Information . . . . .	242
Output Register Information. . . . .	242
Performance Implications . . . . .	242
Syntax . . . . .	242
Parameters . . . . .	243
ABEND Codes . . . . .	245
Return and Reason Codes . . . . .	245
Example . . . . .	246
<b>Chapter 26. IWMERES — Change an Enclave . . . . .</b>	<b>247</b>
Environment . . . . .	247
Programming Requirements . . . . .	247
Restrictions. . . . .	247
Input Register Information . . . . .	248
Output Register Information. . . . .	248
Performance Implications . . . . .	248
Syntax . . . . .	248
Parameters . . . . .	249
ABEND Codes . . . . .	251
Return and Reason Codes . . . . .	251
Example . . . . .	254
<b>Chapter 27. IWMESTOP – Stopping a Work Request . . . . .</b>	<b>255</b>
Environment . . . . .	255
Programming Requirements . . . . .	255
Restrictions. . . . .	256
Input Register Information . . . . .	256
Output Register Information. . . . .	256
Performance Implications . . . . .	256
Syntax . . . . .	256
Parameters . . . . .	257
ABEND Codes . . . . .	259

Return and Reason Codes . . . . .	259
Example . . . . .	261
<b>Chapter 28. IWMESTRT – Starting a Work Request . . . . .</b>	<b>263</b>
Environment . . . . .	263
Programming Requirements . . . . .	263
Restrictions. . . . .	264
Input Register Information . . . . .	264
Output Register Information. . . . .	264
Performance Implications . . . . .	264
Syntax . . . . .	265
Parameters. . . . .	265
ABEND Codes . . . . .	268
Return and Reason Codes . . . . .	268
Example . . . . .	270
<b>Chapter 29. IWMESQRY – Query Enclave State . . . . .</b>	<b>271</b>
Environment . . . . .	271
Programming Requirements . . . . .	271
Restrictions. . . . .	271
Input Register Information . . . . .	271
Output Register Information. . . . .	271
Performance Implications . . . . .	272
Syntax . . . . .	272
Parameters. . . . .	272
ABEND Codes . . . . .	274
Return and Reason Codes . . . . .	274
Example . . . . .	275
<b>Chapter 30. IWMEUBLK – Work Request No Longer Blocked . . . . .</b>	<b>277</b>
Environment . . . . .	277
Programming Requirements . . . . .	277
Restrictions. . . . .	278
Input Register Information . . . . .	278
Output Register Information. . . . .	278
Performance Implications . . . . .	278
Syntax . . . . .	278
Parameters. . . . .	279
ABEND Codes . . . . .	281
Return and Reason Codes . . . . .	281
Examples . . . . .	283
<b>Chapter 31. IWMEEXPT – WLM Export Service . . . . .</b>	<b>285</b>
Environment . . . . .	285
Programming Requirements . . . . .	285
Restrictions. . . . .	285
Input Register Information . . . . .	285
Output Register Information. . . . .	286
Performance Implications . . . . .	286
Syntax . . . . .	286
Parameters. . . . .	286
ABEND Codes . . . . .	288
Return and Reason Codes . . . . .	288
Example . . . . .	291
<b>Chapter 32. IWMIMPT – WLM Import Service . . . . .</b>	<b>293</b>

Environment . . . . .	293
Programming Requirements . . . . .	293
Restrictions. . . . .	293
Input Register Information . . . . .	293
Output Register Information. . . . .	293
Performance Implications . . . . .	294
Syntax . . . . .	294
Parameters. . . . .	294
ABEND Codes . . . . .	296
Return and Reason Codes . . . . .	297
Example . . . . .	299
<b>Chapter 33. IWMMABNL – Record Abnormal Event . . . . .</b>	<b>301</b>
Environment . . . . .	301
Programming Requirements . . . . .	301
Restrictions. . . . .	301
Input Register Information . . . . .	302
Output Register Information. . . . .	302
Performance Implications . . . . .	302
Syntax . . . . .	302
Parameters. . . . .	302
ABEND Codes . . . . .	304
Return and Reason Codes . . . . .	304
Example . . . . .	304
<b>Chapter 34. IWMMDELE – Delete Monitoring Environment . . . . .</b>	<b>305</b>
Environment . . . . .	305
Programming Requirements . . . . .	305
Restrictions. . . . .	305
Input Register Information . . . . .	305
Output Register Information. . . . .	305
Performance Implications . . . . .	306
Syntax . . . . .	306
Parameters. . . . .	306
ABEND Codes . . . . .	307
Return and Reason Codes . . . . .	307
Example . . . . .	308
<b>Chapter 35. IWMMEXTR – Monitor Environment Extract Service . . . . .</b>	<b>309</b>
Environment . . . . .	309
Programming Requirements . . . . .	309
Restrictions. . . . .	310
Input Register Information . . . . .	310
Output Register Information. . . . .	310
Performance Implications . . . . .	310
Syntax . . . . .	310
Parameters. . . . .	311
ABEND Codes . . . . .	314
Return and Reason Codes . . . . .	314
<b>Chapter 36. IWMMNTFY – Notify of Work Execution Completion . . . . .</b>	<b>317</b>
Environment . . . . .	317
Programming Requirements . . . . .	317
Restrictions. . . . .	317
Input Register Information . . . . .	318
Output Register Information. . . . .	318

Performance Implications . . . . .	318
Syntax . . . . .	319
Parameters . . . . .	319
ABEND Codes . . . . .	322
Return and Reason Codes . . . . .	323
<b>Chapter 37. IWMMRELA – Relate Monitoring Environment Service . . . . .</b>	<b>327</b>
Environment . . . . .	327
Programming Requirements . . . . .	327
Restrictions. . . . .	328
Input Register Information . . . . .	328
Output Register Information. . . . .	328
Performance Implications . . . . .	328
Syntax . . . . .	329
Parameters . . . . .	329
ABEND Codes . . . . .	333
Return and Reason Codes . . . . .	333
Example . . . . .	335
<b>Chapter 38. IWMMSWCH – Switch Monitoring Environment . . . . .</b>	<b>337</b>
Environment . . . . .	337
Programming Requirements . . . . .	337
Restrictions. . . . .	337
Input Register Information . . . . .	337
Output Register Information. . . . .	338
Performance Implications . . . . .	338
Syntax . . . . .	338
Parameters . . . . .	339
ABEND Codes . . . . .	340
Return and Reason Codes . . . . .	341
Example . . . . .	341
<b>Chapter 39. IWMMXDC – Exit For Collecting Resource Data . . . . .</b>	<b>343</b>
Environment . . . . .	343
Programming Requirements . . . . .	343
Restrictions. . . . .	344
Input Register Information . . . . .	344
Output Register Information. . . . .	344
Performance Implications . . . . .	344
Syntax . . . . .	344
Parameters . . . . .	345
ABEND Codes . . . . .	347
Return and Reason Codes . . . . .	348
Example . . . . .	348
<b>Chapter 40. IWMMXFER – Transfer Monitoring Environment . . . . .</b>	<b>349</b>
Environment . . . . .	349
Programming Requirements . . . . .	349
Restrictions. . . . .	350
Input Register Information . . . . .	350
Output Register Information. . . . .	350
Performance Implications . . . . .	351
Syntax . . . . .	351
Parameters . . . . .	351
ABEND Codes . . . . .	355
Return and Reason Codes . . . . .	355

|  
|  
|  
|  
|  
|  
|  
|  
|  
|  
|  
|  
|

	<b>Chapter 41. IWMMXRA – Exit For Adjusting Resources</b>	357
	Environment	357
	Programming Requirements	357
	Restrictions.	358
	Input Register Information	358
	Output Register Information.	358
	Performance Implications	358
	Syntax	358
	Parameters.	359
	ABEND Codes	361
	Return and Reason Codes	361
	Example	361
	<b>Chapter 42. IWMPACT – Activate Service Policy</b>	363
	Environment	363
	Programming Requirements	363
	Restrictions.	364
	Input Register Information	364
	Output Register Information.	364
	Performance Implications	364
	Syntax	364
	Parameters.	365
	ABEND Codes	366
	Return and Reason Codes	367
	<b>Chapter 43. IWMPQRY – Query Active Service Policy</b>	371
	Environment	371
	Programming Requirements	371
	Restrictions.	372
	Input Register Information	372
	Output Register Information.	372
	Performance Implications	372
	Syntax	372
	Parameters.	373
	ABEND Codes	374
	Return and Reason Codes	374
	<b>Chapter 44. IWMRCOLL – Collecting Workload Activity Data</b>	377
	Environment	377
	Programming Requirements	378
	Restrictions.	378
	Input Register Information	378
	Output Register Information.	378
	Performance Implications	378
	Syntax	379
	Parameters.	379
	ABEND Codes	382
	Return and Reason Codes	382
	Example	384
	<b>Chapter 45. IWMRESET – Change a Job</b>	385
	Environment	385
	Programming Requirements	385
	Restriction	385
	Input Register Information	385
	Output Register Information.	386

Performance Implications . . . . .	386
Syntax . . . . .	386
Parameters . . . . .	387
ABEND Codes . . . . .	389
Return and Reason Codes . . . . .	389
Example . . . . .	392
<b>Chapter 46. IWMRPT – Report on Work Request Completion . . . . .</b>	<b>393</b>
Environment . . . . .	393
Programming Requirements . . . . .	393
Restrictions. . . . .	393
Input Register Information . . . . .	394
Output Register Information. . . . .	394
Performance Implications . . . . .	395
Syntax . . . . .	395
Parameters . . . . .	395
ABEND Codes . . . . .	401
Return and Reason Codes . . . . .	402
<b>Chapter 47. IWMRQRY – Collecting Address Space Delay Information . . . . .</b>	<b>405</b>
Environment . . . . .	405
Programming Requirements . . . . .	405
Restrictions. . . . .	406
Input Register Information . . . . .	406
Output Register Information. . . . .	406
Performance Implications . . . . .	406
Syntax . . . . .	407
Parameters . . . . .	407
ABEND Codes . . . . .	410
Return and Reason Codes . . . . .	410
<b>Chapter 48. IWMSEDES – Scheduling Environments Determine Execution Service . . . . .</b>	<b>413</b>
Environment . . . . .	413
Programming Requirements . . . . .	413
Restrictions. . . . .	413
Input Register Information . . . . .	414
Output Register Information. . . . .	414
Performance Implications . . . . .	414
Syntax . . . . .	414
Parameters . . . . .	414
ABEND Codes . . . . .	416
Return and Reason Codes . . . . .	416
Example . . . . .	418
<b>Chapter 49. IWMSEQRY – Scheduling Environments Query Service . . . . .</b>	<b>419</b>
Environment . . . . .	419
Programming Requirements . . . . .	419
Restriction . . . . .	419
Input Register Information . . . . .	420
Output Register Information. . . . .	420
Performance Implications . . . . .	420
Syntax . . . . .	420
Parameters . . . . .	420
ABEND Codes . . . . .	422
Return and Reason Codes . . . . .	422



Example . . . . .	424
<b>Chapter 50. IWMSESET – Scheduling Environments Set Resource . . . . .</b>	<b>425</b>
Environment . . . . .	425
Programming Requirements . . . . .	425
Restrictions. . . . .	425
Input Register Information . . . . .	425
Output Register Information. . . . .	426
Performance Implications . . . . .	426
Syntax . . . . .	426
Parameters. . . . .	426
ABEND Codes . . . . .	428
Return and Reason Codes . . . . .	428
Example . . . . .	430
<b>Chapter 51. IWMSEVAL – Scheduling Environments Validate Service . . . . .</b>	<b>431</b>
Environment . . . . .	431
Programming Requirements . . . . .	431
Restrictions. . . . .	431
Input Register Information . . . . .	431
Output Register Information. . . . .	432
Performance Implications . . . . .	432
Syntax . . . . .	432
Parameters. . . . .	432
ABEND Codes . . . . .	434
Return and Reason Codes . . . . .	434
Example . . . . .	435
<b>Chapter 52. IWMSINF — WLM Server Manager Inform Service . . . . .</b>	<b>437</b>
Environment . . . . .	437
Programming Requirements . . . . .	437
Restrictions. . . . .	438
Input Register Information . . . . .	438
Output Register Information. . . . .	438
Performance Implications . . . . .	438
Syntax . . . . .	438
Parameters. . . . .	439
ABEND Codes . . . . .	441
Return and Reason Codes . . . . .	442
Example . . . . .	443
<b>Chapter 53. IWMSRDNS – Get Sysplex Routing Location List . . . . .</b>	<b>445</b>
Environment . . . . .	445
Programming Requirements . . . . .	445
Restrictions. . . . .	445
Input Register Information . . . . .	445
Output Register Information. . . . .	445
Performance Implications . . . . .	446
Syntax . . . . .	446
Parameters. . . . .	446
ABEND Codes . . . . .	448
Return and Reason Codes . . . . .	448
Example . . . . .	450
<b>Chapter 54. IWMSRDRS – Deregister a Server for Sysplex Routing . . . . .</b>	<b>451</b>
Environment . . . . .	451

Programming Requirements . . . . .	451
Restrictions. . . . .	451
Output Register Information. . . . .	452
Performance Implications . . . . .	452
Syntax . . . . .	452
Parameters. . . . .	452
ABEND Codes . . . . .	454
Return and Reason Codes . . . . .	454
<b>Chapter 55. IWMSRFSV – Finding a Sysplex Routing Server . . . . .</b>	<b>457</b>
Environment . . . . .	457
Programming Requirements . . . . .	457
Restrictions. . . . .	457
Input Register Information . . . . .	458
Output Register Information. . . . .	458
Performance Implications . . . . .	458
Syntax . . . . .	458
Parameters. . . . .	459
ABEND Codes . . . . .	460
Return and Reason Codes . . . . .	461
Example . . . . .	464
<b>Chapter 56. IWMSRSG – Register a Server for Sysplex Routing . . . . .</b>	<b>465</b>
Environment . . . . .	465
Programming Requirements . . . . .	466
Restrictions. . . . .	466
Input Register Information . . . . .	466
Output Register Information. . . . .	466
Performance Implications . . . . .	467
Syntax . . . . .	467
Parameters. . . . .	467
ABEND Codes . . . . .	470
Return and Reason Codes . . . . .	470
Example . . . . .	471
<b>Chapter 57. IWMSRSRS – Determine Where to Route Work. . . . .</b>	<b>473</b>
Environment . . . . .	474
Programming Requirements . . . . .	474
Restrictions. . . . .	474
Input Register Information . . . . .	475
Output Register Information. . . . .	475
Performance Implications . . . . .	475
Syntax . . . . .	475
Parameters. . . . .	476
ABEND Codes . . . . .	479
Return and Reason Codes . . . . .	479
Example . . . . .	481
<b>Chapter 58. IWMUEXPT – WLM Undo Export Service . . . . .</b>	<b>483</b>
Environment . . . . .	483
Programming Requirements . . . . .	483
Restrictions. . . . .	483
Input Register Information . . . . .	483
Output Register Information. . . . .	483
Performance Implications . . . . .	484
Syntax . . . . .	484

Parameters . . . . .	484
ABEND Codes . . . . .	486
Return and Reason Codes . . . . .	486
Example . . . . .	488
<b>Chapter 59. IWMUIMPT – WLM Undo Import Service . . . . .</b>	<b>489</b>
Environment . . . . .	489
Programming Requirements . . . . .	489
Restrictions. . . . .	489
Input Register Information . . . . .	489
Output Register Information. . . . .	489
Performance Implications . . . . .	490
Syntax . . . . .	490
Parameters. . . . .	490
ABEND Codes . . . . .	492
Return and Reason Codes . . . . .	492
Example . . . . .	494
<b>Chapter 60. IWMWMCON – Modify Connect Information . . . . .</b>	<b>495</b>
Environment . . . . .	495
Programming Requirements . . . . .	495
Restrictions. . . . .	496
Input Register Information . . . . .	496
Output Register Information. . . . .	496
Performance Implications . . . . .	496
Syntax . . . . .	497
Parameters. . . . .	497
ABEND Codes . . . . .	499
Return and Reason Codes . . . . .	499
Examples . . . . .	501
<b>Chapter 61. IWMWQRY – Query Service . . . . .</b>	<b>503</b>
Environment . . . . .	503
Programming Requirements . . . . .	503
Restrictions. . . . .	503
Input Register Information . . . . .	503
Output Register Information. . . . .	504
Performance Implications . . . . .	504
Syntax . . . . .	504
Parameters. . . . .	504
ABEND Codes . . . . .	506
Return and Reason Codes . . . . .	506
Example . . . . .	506
<b>Chapter 62. IWMWQWRK – Query Work Service . . . . .</b>	<b>507</b>
Environment . . . . .	507
Programming Requirements . . . . .	507
Restrictions. . . . .	508
Input Register Information . . . . .	508
Output Register Information. . . . .	508
Performance Implications . . . . .	508
Syntax . . . . .	508
Parameters. . . . .	509
ABEND Codes . . . . .	512
Return and Reason Codes . . . . .	512
Example . . . . .	513

<b>Chapter 63. IWMWSYSQ – Querying System Information.</b>	515
Environment	516
Programming Requirements	516
Restrictions.	516
Input Register Information	517
Output Register Information.	517
Performance Implications	517
Syntax	517
Parameters.	517
ABEND Codes	519
Return and Reason Codes	519
Example	521
<b>Chapter 64. IWM4AEDF — Defining Dynamic Application Environments to Workload Management</b>	523
Environment	523
Programming Requirements	523
Restrictions.	524
Input Register Information	524
Output Register Information.	524
Performance Implications	524
Syntax	524
Parameters.	525
ABEND Codes	528
Return and Reason Codes	528
Example	530
<b>Chapter 65. IWM4CON – Connecting to Workload Management</b>	533
Environment	533
Programming Requirements	534
Restrictions.	534
Input Register Information	534
Output Register Information.	535
Performance Implications	535
Syntax	536
Parameters.	537
ABEND Codes	545
Return and Reason Codes	545
Examples	551
<b>Chapter 66. IWM4DIS – Disconnecting from Workload Management</b>	553
Environment	553
Programming Requirements	554
Restrictions.	554
Input Register Information	554
Output Register Information.	554
Performance Implications	555
Syntax	555
Parameters.	555
ABEND Codes	557
Return and Reason Codes	557
<b>Chapter 67. IWM4ECRE – Create an Enclave</b>	561
Environment	562
Programming Requirements	562
Restrictions.	562

Input Register Information . . . . .	563
Output Register Information . . . . .	563
Performance Implications . . . . .	563
Syntax . . . . .	564
Parameters . . . . .	564
ABEND Codes . . . . .	568
Return and Reason Codes . . . . .	568
<b>Chapter 68. IWM4EDEL – Delete an Enclave . . . . .</b>	<b>573</b>
Environment . . . . .	573
Programming Requirements . . . . .	573
Restrictions. . . . .	574
Input Register Information . . . . .	574
Output Register Information. . . . .	574
Performance Implications . . . . .	574
Syntax . . . . .	574
Parameters . . . . .	575
ABEND Codes . . . . .	578
Return and Reason Codes . . . . .	578
Example . . . . .	579
<b>Chapter 69. IWM4HLTH – Set Server Health Indicator . . . . .</b>	<b>581</b>
Environment . . . . .	581
Programming Requirements . . . . .	581
Restrictions. . . . .	582
Input Register Information . . . . .	582
Output Register Information. . . . .	582
Performance Implications . . . . .	582
Syntax . . . . .	582
Parameters . . . . .	583
ABEND Codes . . . . .	584
Return and Reason Codes . . . . .	584
Example . . . . .	586
<b>Chapter 70. IWM4MCHS – Monitoring Change State Of Work Unit . . . . .</b>	<b>587</b>
Environment . . . . .	587
Programming Requirements . . . . .	587
Restrictions. . . . .	587
Input Register Information . . . . .	588
Output Register Information. . . . .	588
Performance Implications . . . . .	588
Syntax . . . . .	589
Parameters . . . . .	589
ABEND Codes . . . . .	592
Return and Reason Codes . . . . .	593
<b>Chapter 71. IWM4MCRE – Create Delay Monitoring Environment . . . . .</b>	<b>595</b>
Environment . . . . .	595
Programming Requirements . . . . .	595
Restrictions. . . . .	596
Input Register Information . . . . .	596
Output Register Information. . . . .	596
Performance Implications . . . . .	597
Syntax . . . . .	597
Parameters . . . . .	597
ABEND Codes . . . . .	602

|  
|  
|  
|  
|  
|  
|  
|  
|  
|  
|  
|

	Return and Reason Codes . . . . .	602
	<b>Chapter 72. IWM4MDRG – Deregister a Resource From Monitoring</b> . . . . .	605
	Environment . . . . .	605
	Programming Requirements . . . . .	605
	Restrictions. . . . .	605
	Input Register Information . . . . .	606
	Output Register Information. . . . .	606
	Performance Implications . . . . .	606
	Syntax . . . . .	606
	Parameters. . . . .	606
	ABEND Codes . . . . .	608
	Return and Reason Codes . . . . .	608
	Example . . . . .	610
	<b>Chapter 73. IWM4MINI – Initialize Monitoring Environment</b> . . . . .	611
	Environment . . . . .	611
	Programming Requirements . . . . .	612
	Restrictions. . . . .	612
	Input Register Information . . . . .	612
	Output Register Information. . . . .	612
	Performance Implications . . . . .	613
	Syntax . . . . .	613
	Parameters. . . . .	614
	ABEND Codes . . . . .	623
	Return and Reason Codes . . . . .	623
	Example . . . . .	625
	<b>Chapter 74. IWM4MREG –Register a Resource For Monitoring</b> . . . . .	627
	Environment . . . . .	627
	Programming Requirements . . . . .	627
	Restrictions. . . . .	628
	Input Register Information . . . . .	628
	Output Register Information. . . . .	628
	Performance Implications . . . . .	628
	Syntax . . . . .	628
	Parameters. . . . .	629
	ABEND Codes . . . . .	631
	Return and Reason Codes . . . . .	632
	Example . . . . .	633
	<b>Chapter 75. IWM4QDE – Deleting a Request from the Queue for An Execution Address Space</b> . . . . .	635
	Environment . . . . .	635
	Programming Requirements . . . . .	635
	Restrictions. . . . .	635
	Input Register Information . . . . .	635
	Output Register Information. . . . .	636
	Performance Implications . . . . .	636
	Syntax . . . . .	636
	Parameters. . . . .	636
	ABEND Codes . . . . .	638
	Return and Reason Codes . . . . .	638
	Example . . . . .	640

<b>Chapter 76. IWM4QIN – Inserting a Request Onto the Queue for An Execution Address space</b>	641
Environment	641
Programming Requirements	641
Restrictions.	642
Input Register Information	642
Output Register Information.	642
Performance Implications	642
Syntax	642
Parameters.	643
ABEND Codes	646
Return and Reason Codes	647
Example	651
<b>Chapter 77. IWM4SLI – Application Environment Limit Service</b>	653
Environment	653
Programming Requirements	653
Restrictions.	654
Input Register Information	654
Output Register Information.	654
Performance Implications	654
Syntax	654
Parameters.	655
ABEND Codes	657
Return and Reason Codes	657
Example	659
<b>Chapter 78. IWM4SRSC – Obtain Server-Specific Routing Information</b>	661
Environment	662
Programming Requirements	662
Restrictions.	662
Input Register Information	662
Output Register Information.	662
Performance Implications	663
Syntax	663
Parameters.	663
ABEND Codes	665
Return and Reason Codes	665
Example	668
<b>Chapter 79. IWM4SSL – Selecting a Request from a Caller’s Work Manager Queue</b>	669
Environment	669
Programming Requirements	669
Restrictions.	669
Input Register Information	670
Output Register Information.	670
Performance Implications	670
Syntax	670
Parameters.	671
ABEND Codes	673
Return and Reason Codes	673
Example	675
<b>Chapter 80. IWM4SSM – WLM Server Select Secondary Service</b>	677
Environment	677

Programming Requirements . . . . .	678
Restrictions. . . . .	678
Input Register Information . . . . .	678
Output Register Information. . . . .	678
Performance Implications . . . . .	678
Syntax . . . . .	678
Parameters. . . . .	679
ABEND Codes . . . . .	680
Return and Reason Codes . . . . .	680
Example . . . . .	682
<b>Chapter 81. IWM4STBG – Beginning a Request from a Caller’s Work Manager Queue . . . . .</b>	<b>683</b>
Environment . . . . .	683
Programming Requirements . . . . .	683
Restrictions. . . . .	684
Input Register Information . . . . .	684
Output Register Information. . . . .	684
Performance Implications . . . . .	684
Syntax . . . . .	685
Parameters. . . . .	685
ABEND Codes . . . . .	687
Return and Reason Codes . . . . .	687
Example . . . . .	689
<b>Chapter 82. IWM4STEN – End a Request from a Caller’s Work Manager Queue . . . . .</b>	<b>691</b>
Environment . . . . .	691
Programming Requirements . . . . .	691
Restrictions. . . . .	691
Input Register Information . . . . .	691
Output Register Information. . . . .	692
Performance Implications . . . . .	692
Syntax . . . . .	692
Parameters. . . . .	692
ABEND Codes . . . . .	694
Return and Reason Codes . . . . .	694
Example . . . . .	696
<b>Chapter 83. IWM4TAF — WLM Temporal Affinity Service . . . . .</b>	<b>697</b>
Environment . . . . .	697
Programming Requirements . . . . .	697
Restrictions. . . . .	697
Input Register Information . . . . .	698
Output Register Information. . . . .	698
Performance Implications . . . . .	698
Syntax . . . . .	698
Parameters. . . . .	698
ABEND Codes . . . . .	700
Return and Reason Codes . . . . .	700
Example . . . . .	703

---

**Part 3. Appendixes . . . . . 705**

**Appendix A. SMF Type 99 Action Codes . . . . . 707**



<b>Appendix B. Application Validation Reason Codes</b> . . . . .	751
<b>Appendix C. C Language Interfaces for Workload Management Services</b>	761
Interfaces for Sysplex Routing Services . . . . .	763
Interface for Querying a Virtual Server. . . . .	763
<b>Appendix D. WLM Services supporting 31-bit address space only</b> . . . . .	765
IWMAEDEF — Defining Dynamic Application Environments to Workload	
Management . . . . .	765
Environment . . . . .	765
Programming Requirements . . . . .	765
Restrictions. . . . .	766
Input Register Information . . . . .	766
Output Register Information. . . . .	766
Performance Implications . . . . .	766
Syntax . . . . .	766
Parameters. . . . .	767
ABEND Codes . . . . .	770
Return and Reason Codes . . . . .	770
Example . . . . .	772
IWMCONN – Connecting to Workload Management. . . . .	773
Environment . . . . .	774
Programming Requirements . . . . .	774
Restrictions. . . . .	774
Input Register Information . . . . .	775
Output Register Information. . . . .	775
Performance Implications . . . . .	775
Syntax . . . . .	776
Parameters. . . . .	777
ABEND Codes . . . . .	785
Return and Reason Codes . . . . .	785
Examples . . . . .	791
IWMDISC – Disconnecting from Workload Management . . . . .	792
Environment . . . . .	792
Programming Requirements . . . . .	793
Restrictions. . . . .	793
Input Register Information . . . . .	793
Output Register Information. . . . .	793
Performance Implications . . . . .	794
Syntax . . . . .	794
Parameters. . . . .	794
ABEND Codes . . . . .	795
Return and Reason Codes . . . . .	796
IWMECREA – Create an Enclave . . . . .	798
Environment . . . . .	799
Programming Requirements . . . . .	799
Restrictions. . . . .	800
Input Register Information . . . . .	800
Output Register Information. . . . .	800
Performance Implications . . . . .	801
Syntax . . . . .	801
Parameters. . . . .	801
ABEND Codes . . . . .	806
Return and Reason Codes . . . . .	806
IWMEDELE – Delete an Enclave. . . . .	809
Environment . . . . .	809

Programming Requirements . . . . .	810
Restrictions. . . . .	810
Input Register Information . . . . .	810
Output Register Information. . . . .	810
Performance Implications. . . . .	811
Syntax . . . . .	811
Parameters . . . . .	811
ABEND Codes . . . . .	814
Return and Reason Codes . . . . .	814
Example . . . . .	816
IWMMCHST – Monitoring Change State Of Work Unit . . . . .	816
Environment . . . . .	816
Programming Requirements . . . . .	816
Restrictions. . . . .	817
Input Register Information . . . . .	817
Output Register Information. . . . .	817
Performance Implications . . . . .	818
Syntax . . . . .	818
Parameters. . . . .	818
ABEND Codes . . . . .	822
Return and Reason Codes . . . . .	822
IWMMCREA – Create Delay Monitoring Environment . . . . .	822
Environment . . . . .	823
Programming Requirements . . . . .	823
Restrictions. . . . .	823
Input Register Information . . . . .	824
Output Register Information. . . . .	824
Performance Implications . . . . .	824
Syntax . . . . .	824
Parameters. . . . .	825
ABEND Codes . . . . .	829
Return and Reason Codes . . . . .	829
IWMMINIT – Initialize Monitoring Environment . . . . .	832
Environment . . . . .	833
Programming Requirements . . . . .	833
Restrictions. . . . .	833
Input Register Information . . . . .	834
Output Register Information. . . . .	834
Performance Implications . . . . .	834
Syntax . . . . .	834
Parameters. . . . .	836
ABEND Codes . . . . .	845
Return and Reason Codes . . . . .	845
Examples . . . . .	846
IWMQDEL – Deleting a Request from the Queue for An Execution Address	
Space . . . . .	846
Environment . . . . .	847
Programming Requirements . . . . .	847
Restrictions. . . . .	847
Input Register Information . . . . .	847
Output Register Information. . . . .	847
Performance Implications . . . . .	848
Syntax . . . . .	848
Parameters. . . . .	848
ABEND Codes . . . . .	850
Return and Reason Codes . . . . .	850

Example . . . . .	852
IWMQINS – Inserting a Request Onto the Queue for An Execution Address	
space . . . . .	852
Environment . . . . .	853
Programming Requirements . . . . .	853
Restrictions. . . . .	853
Input Register Information . . . . .	853
Output Register Information. . . . .	853
Performance Implications . . . . .	854
Syntax . . . . .	854
Parameters. . . . .	855
ABEND Codes . . . . .	858
Return and Reason Codes . . . . .	859
Example . . . . .	863
IWMSLIM – Application Environment Limit Service . . . . .	864
Environment . . . . .	864
Programming Requirements . . . . .	864
Restrictions. . . . .	865
Input Register Information . . . . .	865
Output Register Information. . . . .	865
Performance Implications . . . . .	865
Syntax . . . . .	865
Parameters. . . . .	866
ABEND Codes . . . . .	868
Return and Reason Codes . . . . .	868
Example . . . . .	870
IWMSSSEL – Selecting a Request from a Caller's Work Manager Queue . . . . .	871
Environment . . . . .	871
Programming Requirements . . . . .	871
Restrictions. . . . .	871
Input Register Information . . . . .	872
Output Register Information. . . . .	872
Performance Implications . . . . .	872
Syntax . . . . .	872
Parameters. . . . .	873
ABEND Codes . . . . .	875
Return and Reason Codes . . . . .	875
Example . . . . .	877
IWMSSEM – WLM Server Select Secondary Service . . . . .	877
Environment . . . . .	878
Programming Requirements . . . . .	878
Restrictions. . . . .	879
Input Register Information . . . . .	879
Output Register Information. . . . .	879
Performance Implications . . . . .	879
Syntax . . . . .	879
Parameters. . . . .	879
ABEND Codes . . . . .	881
Return and Reason Codes . . . . .	881
Example . . . . .	884
IWMSTBGN – Beginning a Request from a Caller's Work Manager Queue . . . . .	884
Environment . . . . .	884
Programming Requirements . . . . .	885
Restrictions. . . . .	885
Input Register Information . . . . .	885
Output Register Information. . . . .	885

Performance Implications . . . . .	885
Syntax . . . . .	886
Parameters . . . . .	886
ABEND Codes . . . . .	888
Return and Reason Codes . . . . .	888
Example . . . . .	891
<b>IWMSTEND – End a Request from a Caller’s Work Manager Queue . . . . .</b>	<b>891</b>
Environment . . . . .	891
Programming Requirements . . . . .	892
Restrictions. . . . .	892
Input Register Information . . . . .	892
Output Register Information. . . . .	892
Performance Implications . . . . .	892
Syntax . . . . .	892
Parameters . . . . .	893
ABEND Codes . . . . .	894
Return and Reason Codes . . . . .	894
Example . . . . .	897
<b>IWMTAFF — WLM Temporal Affinity Service . . . . .</b>	<b>897</b>
Environment . . . . .	897
Programming Requirements . . . . .	897
Restrictions. . . . .	898
Input Register Information . . . . .	898
Output Register Information. . . . .	898
Performance Implications . . . . .	898
Syntax . . . . .	898
Parameters . . . . .	899
ABEND Codes . . . . .	901
Return and Reason Codes . . . . .	901
Example . . . . .	903
<b>Appendix E. Notices . . . . .</b>	<b>905</b>
Programming Interface Information . . . . .	906
Trademarks. . . . .	907
<b>Appendix F. Accessibility . . . . .</b>	<b>909</b>
Using assistive technologies . . . . .	909
Keyboard navigation of the user interface. . . . .	909
z/OS information . . . . .	909
<b>Index . . . . .</b>	<b>911</b>

---

## Figures

1. Sequence of functions in a single address space transaction manager. . . . .	18
2. Work manager services for a single address space transaction manager. . . . .	19
3. Work manager and delay monitoring services for a single address space transaction manager. . . . .	21
4. Services for a work manager that uses a database manager. . . . .	22
5. Sequence of function in a multiple address space work manager. . . . .	24
6. Example of services that monitor work across multiple address spaces. . . . .	26
7. Creating an Independent Enclave and Scheduling an SRB . . . . .	37
8. Creating an Enclave and Joining Tasks To It. . . . .	39
9. Using Dependent Enclaves . . . . .	41
10. Using a Multisystem Enclave . . . . .	43
11. Sample JCL for Compiling an ARM-instrumented Application in MVS Batch . . . . .	57
12. Sample JCL for Binding an ARM-instrumented Application in MVS Batch . . . . .	58
13. Services for a queueing manager. . . . .	62
14. Exploiting IWMSINF. . . . .	67
15. Example of routing manager services . . . . .	74
16. Obtaining Scheduling Environments . . . . .	80
17. Manipulating Resource State Settings . . . . .	81
18. Scheduling Environment Flow . . . . .	82
19. Example of using sysplex routing services . . . . .	87
20. Using states for presenting CICS delay information . . . . .	95
21. Combining state information for a sysplex view. . . . .	96
22. Combining state information for a service class. . . . .	97
23. Combining state information across subsystems. . . . .	98
24. Self-defining response time distributions . . . . .	99
25. Mixed-class-indication timestamp in relation to the time interval . . . . .	100
26. IWMSVDEF mapping for a service definition with workload extensions. . . . .	105



# Tables

1. Syntax examples . . . . .	xxxiv
2. Work manager services . . . . .	4
3. Execution delay monitoring services . . . . .	5
4. Enclave services . . . . .	6
5. Queueing manager services . . . . .	8
6. Routing manager services . . . . .	10
7. Scheduling Environment Services. . . . .	10
8. Sysplex routing services . . . . .	11
9. Summary of workload reporting services . . . . .	13
10. Administrative application services . . . . .	14
11. Overview of WLM services supporting 64-bit and 31-bit address space . . . . .	15
12. Enclave Characteristics and Resource Accounting . . . . .	47
13. Starting and Stopping Work Requests . . . . .	53
14. WLM Enclave Services and ARM APIs . . . . .	56
15. Sample Commands for Compiling Applications under z/OS UNIX Systems Services . . . . .	58
16. Sample Commands for Binding Applications . . . . .	58
17. CPU consumption table . . . . .	89
18. Using ENF event code 41 to guide data collection for policy changes . . . . .	92
19. Using IWMRCOLL with the workload reporting services on a single system . . . . .	93
20. Using IWMRQRY with the workload reporting services . . . . .	101
21. SERVD validation reason codes . . . . .	106
22. Return and Reason Codes for the IWMCLSFY Macro . . . . .	147
23. Return and Reason Codes for the IWMCNTN Macro . . . . .	157
24. Return and Reason Codes for the IWMCPAFN Macro. . . . .	165
25. Return and Reason Codes for the IWMCQRY Macro . . . . .	173
26. Return and Reason Codes for the IWMDEXTR Macro . . . . .	182
27. Return and Reason Codes for the IWMDINST Macro . . . . .	191
28. Return and Reason Codes for the IWMEBLK Macro . . . . .	199
29. Return and Reason Codes for the IWMECQRY Macro . . . . .	207
30. Return and Reason Codes for the IWMEDEXTR Macro . . . . .	212
31. Return and Reason Codes for the IWMEGCOR Macro . . . . .	220
32. Return and Reason Codes for the IWMEJOIN Macro . . . . .	227
33. Return and Reason Codes for the IWMELEAV Macro . . . . .	232
34. Return and Reason Codes for the IWMEQTM Macro . . . . .	239
35. Return and Reason Codes for the IWMEREG Macro . . . . .	245
36. Return and Reason Codes for the IWMERES Macro . . . . .	252
37. Return and Reason Codes for the IWMESTOP Macro. . . . .	259
38. Return and Reason Codes for the IWMESTRT Macro. . . . .	268
39. Return and Reason Codes for the IWMEQRY Macro . . . . .	274
40. Return and Reason Codes for the IWMEUBLK Macro. . . . .	281
41. Return and Reason Codes for the IWMEPT Macro . . . . .	289
42. Return and Reason Codes for the IWMEIMPT Macro . . . . .	297
43. Return and Reason Codes for the IWMEEXTR Macro . . . . .	314
44. Return and Reason Codes for the IWMEINTFY Macro . . . . .	323
45. Return and Reason Codes for the IWMERELA Macro . . . . .	334
46. Return and Reason Codes for the IWMEXDC Macro . . . . .	348
47. Return and Reason Codes for the IWMEXFER Macro . . . . .	355
48. Return and Reason Codes for the IWMEXRA Macro . . . . .	361
49. Return and Reason Codes for the IWMEPACT Macro . . . . .	367
50. Return and Reason Codes for the IWMEPQRY Macro . . . . .	375
51. Return and Reason Codes for the IWMECOLL Macro. . . . .	382
52. Return and Reason Codes for the IWMERESET Macro. . . . .	390
53. Return and Reason Codes for the IWMERPT Macro . . . . .	402

	54. Return and Reason Codes for the IWMRQRY Macro . . . . .	410
	55. Return and Reason Codes for the IWMSEDES Macro . . . . .	416
	56. Return and Reason Codes for the IWMSEQRY Macro . . . . .	422
	57. Return and Reason Codes for the IWMSESET Macro . . . . .	428
	58. Return and Reason Codes for the IWMSEVAL Macro . . . . .	434
	59. Return and Reason Codes for the IWMSINF Macro . . . . .	442
	60. Return and Reason Codes for the IWMSRDNS Macro . . . . .	449
	61. Return and Reason Codes for the IWMSRDRS Macro . . . . .	455
	62. Return and Reason Codes for the IWMSRFSV Macro . . . . .	461
I	63. Return and Reason Codes for the IWMSRSRG Macro . . . . .	470
	64. Return and Reason Codes for the IWMSRSRS Macro . . . . .	479
	65. Return and Reason Codes for the IWMUEXPT Macro . . . . .	486
	66. Return and Reason Codes for the IWMUIMPT Macro . . . . .	492
	67. Return and Reason Codes for the IWMWMCON Macro . . . . .	499
	68. Return and Reason Codes for the IWMWSYSQ Macro . . . . .	520
	69. Return and Reason Codes for the IWM4AEDF Macro . . . . .	528
	70. Return and Reason Codes for the IWM4CON Macro . . . . .	545
	71. Return and Reason Codes for the IWM4DIS Macro . . . . .	557
	72. Return and Reason Codes for the IWM4ECRE Macro . . . . .	568
I	73. Return and Reason Codes for the IWM4EDEL Macro . . . . .	578
I	74. Return and Reason Codes for the IWM4HLTH Macro . . . . .	585
I	75. Return and Reason Codes for the IWM4MCHS Macro . . . . .	593
I	76. Return and Reason Codes for the IWM4MCRE Macro . . . . .	602
I	77. Return and Reason Codes for the IWM4MDRG Macro . . . . .	608
I	78. Return and Reason Codes for the IWM4MINI Macro . . . . .	624
I	79. Return and Reason Codes for the IWM4MREG Macro . . . . .	632
	80. Return and Reason Codes for the IWM4QDE Macro . . . . .	638
	81. Return and Reason Codes for the IWM4QIN Macro . . . . .	647
	82. Return and Reason Codes for the IWM4SLI Macro . . . . .	657
	83. Return and Reason Codes for the IWM4SRSC Macro . . . . .	666
	84. Return and Reason Codes for the IWM4SSL Macro . . . . .	673
	85. Return and Reason Codes for the IWM4SSM Macro . . . . .	681
	86. Return and Reason Codes for the IWM4STBG Macro . . . . .	687
	87. Return and Reason Codes for the IWM4STEN Macro . . . . .	694
	88. Return and Reason Codes for the IWM4TAF Macro . . . . .	701
I	89. SMF Type 99 Action Codes . . . . .	707
	90. SERVD validation reason codes . . . . .	751
	91. C Language Interfaces . . . . .	761
	92. C Language Interfaces for WLM Sysplex Routing Services . . . . .	763
	93. Return and Reason Codes for the IWMAEDEF Macro . . . . .	771
	94. Return and Reason Codes for the IWMCONN Macro . . . . .	786
	95. Return and Reason Codes for the IWMDISC Macro . . . . .	796
	96. Return and Reason Codes for the IWMECREA Macro . . . . .	806
I	97. Return and Reason Codes for the IWM4EDEL Macro . . . . .	815
I	98. Return and Reason Codes for the IWM4MCHS Macro . . . . .	822
I	99. Return and Reason Codes for the IWM4MCRE Macro . . . . .	830
I	100. Return and Reason Codes for the IWM4MINI Macro . . . . .	845
	101. Return and Reason Codes for the IWMQDEL Macro . . . . .	850
	102. Return and Reason Codes for the IWMQINS Macro . . . . .	859
	103. Return and Reason Codes for the IWMSLIM Macro . . . . .	868
	104. Return and Reason Codes for the IWMSSEL Macro . . . . .	875
	105. Return and Reason Codes for the IWMSSEM Macro . . . . .	881
	106. Return and Reason Codes for the IWMSTBGN Macro . . . . .	888
	107. Return and Reason Codes for the IWMSTEND Macro . . . . .	895
	108. Return and Reason Codes for the IWMTAFF Macro . . . . .	901



---

## About This Document

This document supports z/OS (5694-A01). This document describes the MVS workload management services. The services are intended for programmers who write authorized programs.

This document is divided into two sections:

- Using the Workload Management Services  
This section provides an overview of the services, and how to use them.
- Workload Management Services Reference  
This section describes each service, its syntax, its parameters, and an example of how to issue it.

---

## Who Should Use This Document

Programmers using assembler language can use the macros described in this document to invoke the services they need. This document includes some guidance information, and detailed information, such as the function, the syntax, and parameters needed to code the macros. To understand the information in this document, programmers should have read *z/OS MVS Planning: Workload Management*.

---

## Where to Find More Information

Where necessary, this document references information in other documents, using shortened versions of the document title. For complete titles and order numbers of the documents for all products that are part of z/OS, see *z/OS Information Roadmap*.

Softcopy Title Used in This Document	Title	Order Number
<i>z/OS MVS Planning: Workload Management</i>	<i>z/OS MVS Planning: Workload Management</i>	SA22-7602
<i>z/OS MVS Initialization and Tuning Guide</i>	<i>z/OS MVS Initialization and Tuning Guide</i>	SA22-7591
<i>z/OS MVS Initialization and Tuning Reference</i>	<i>z/OS MVS Initialization and Tuning Reference</i>	SA22-7592
<i>z/OS MVS Programming: Authorized Assembler Services Guide</i>	<i>z/OS MVS Programming: Authorized Assembler Services Guide</i>	SA22-7608
<i>z/OS MVS Programming: Authorized Assembler Services Reference EDT-IXG</i>	<i>z/OS MVS Programming: Authorized Assembler Services Reference EDT-IXG</i>	SA22-7610
<i>z/OS MVS Programming: Authorized Assembler Services Reference SET-WTO</i>	<i>z/OS MVS Programming: Authorized Assembler Services Reference SET-WTO</i>	SA22-7612
<i>z/OS MVS System Management Facilities (SMF)</i>	<i>z/OS MVS System Management Facilities (SMF)</i>	SA22-7630
<i>z/OS MVS Data Areas, Vol 1 (ABEP-DALT)</i>	<i>z/OS MVS Data Areas, Vol 1 (ABEP-DALT)</i>	GA22-7581
<i>z/OS MVS Data Areas, Vol 2 (DCCB-ITZYRETC)</i>	<i>z/OS MVS Data Areas, Vol 2 (DCCB-ITZYRETC)</i>	GA22-7582
<i>z/OS MVS Data Areas, Vol 3 (IVT-RCWK)</i>	<i>z/OS MVS Data Areas, Vol 3 (IVT-RCWK)</i>	GA22-7583
<i>z/OS MVS Data Areas, Vol 4 (RD-SRRA)</i>	<i>z/OS MVS Data Areas, Vol 4 (RD-SRRA)</i>	GA22-7584
<i>z/OS MVS Data Areas, Vol 5 (SSAG-XTLST)</i>	<i>z/OS MVS Data Areas, Vol 5 (SSAG-XTLST)</i>	GA22-7585
<i>z/OS XL C/C++ Run-Time Library Reference</i>	<i>z/OS XL C/C++ Run-Time Library Reference</i>	SA22-7821

---

## The WLM/SRM Web Page

For the latest information, see the WLM/SRM web page at:

<http://www.ibm.com/servers/eserver/zseries/zos/wlm/>

## Using LookAt to look up message explanations

LookAt is an online facility that lets you look up explanations for most of the IBM® messages you encounter, as well as for some system abends and codes. Using LookAt to find information is faster than a conventional search because in most cases LookAt goes directly to the message explanation.

You can use LookAt from these locations to find IBM message explanations for z/OS® elements and features, z/VM®, VSE/ESA™, and Clusters for AIX® and Linux™:

- The Internet. You can access IBM message explanations directly from the LookAt Web site at [www.ibm.com/servers/eserver/zseries/zos/bkserv/lookat/](http://www.ibm.com/servers/eserver/zseries/zos/bkserv/lookat/).
- Your z/OS TSO/E host system. You can install code on your z/OS or z/OS.e systems to access IBM message explanations using LookAt from a TSO/E command line (for example: TSO/E prompt, ISPF, or z/OS UNIX® System Services).
- Your Microsoft® Windows® workstation. You can install LookAt directly from the *z/OS Collection (SK3T-4269)* or the *z/OS and Software Products DVD Collection (SK3T-4271)* and use it from the resulting Windows graphical user interface (GUI). The command prompt (also known as the DOS > command line) version can still be used from the directory in which you install the Windows version of LookAt.
- Your wireless handheld device. You can use the LookAt Mobile Edition from [www.ibm.com/servers/eserver/zseries/zos/bkserv/lookat/lookatm.html](http://www.ibm.com/servers/eserver/zseries/zos/bkserv/lookat/lookatm.html) with a handheld device that has wireless access and an Internet browser (for example: Internet Explorer for Pocket PCs, Blazer or Eudora for Palm OS, or Opera for Linux handheld devices).

You can obtain code to install LookAt on your host system or Microsoft Windows workstation from:

- A CD-ROM in the *z/OS Collection (SK3T-4269)*.
- The *z/OS and Software Products DVD Collection (SK3T-4271)*.
- The LookAt Web site (click **Download** and then select the platform, release, collection, and location that suit your needs). More information is available in the LOOKAT.ME files available during the download process.

## Using IBM Health Checker for z/OS

IBM Health Checker for z/OS is a z/OS component that installations can use to gather information about their system environment and system parameters to help identify potential configuration problems before they impact availability or cause outages. Individual products, z/OS components, or ISV software can provide checks that take advantage of the IBM Health Checker for z/OS framework. This book refers to checks or messages associated with this component.

For additional information about checks and about IBM Health Checker for z/OS, see *IBM Health Checker for z/OS: User's Guide*. Starting with z/OS V1R4, z/OS users can obtain the IBM Health Checker for z/OS from the z/OS Downloads page at [www.ibm.com/servers/eserver/zseries/zos/downloads/](http://www.ibm.com/servers/eserver/zseries/zos/downloads/).

SDSF also provides functions to simplify the management of checks. See *z/OS SDSF Operation and Customization* for additional information.

---

## How to Read a Syntax Diagram

### 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:

Symbol	Definition
▶—	Indicates the beginning of the syntax diagram.
—→	Indicates that the syntax diagram is continued to the next line.
▶—	Indicates that the syntax is continued from the previous line.
—▶	Indicates the end of the syntax diagram.

#### 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.

**Note:** If a syntax diagram shows a character that is not alphanumeric (for example, parentheses, periods, commas, equal signs, a blank space), enter the character as part of the syntax.

Keywords, variables, and operators may be displayed as required, optional, or default. Fragments, separators, and delimiters may be displayed as required or optional.

Item type	Definition
<b>Required</b>	Required items are displayed on the main path of the horizontal line.
<b>Optional</b>	Optional items are displayed below the main path of the horizontal line.

**Default** Default items are displayed above the main path of the horizontal line.

## Syntax examples

The following table provides syntax examples.

Table 1. Syntax examples

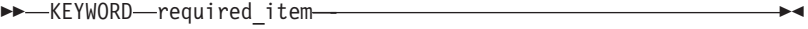
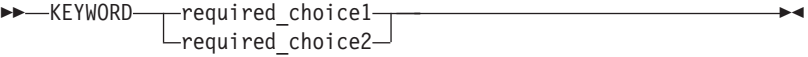
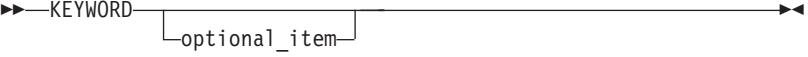
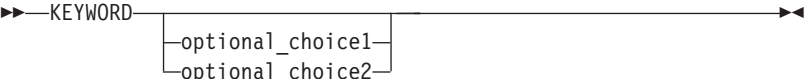
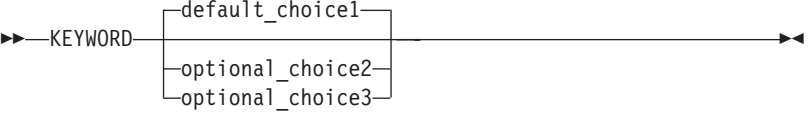

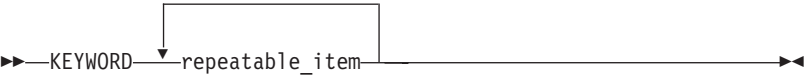
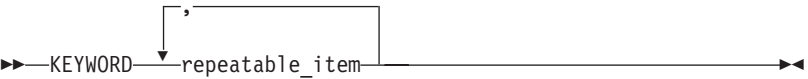
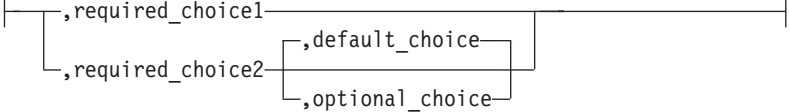
Item	Syntax example
Required item.	
Required items appear on the main path of the horizontal line. You must specify these items.	
Required choice.	
A required choice (two or more items) appears in a vertical stack on the main path of the horizontal line. You must choose one of the items in the stack.	
Optional item.	
Optional items appear below the main path of the horizontal line.	
Optional choice.	
An optional choice (two or more items) appears in a vertical stack below the main path of the horizontal line. You may choose one of the items in the stack.	
Default.	
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.	
Variable.	
Variables appear in lowercase italics. They represent names or values.	
Repeatable item.	
An arrow returning to the left above the main path of the horizontal line indicates an item that can be repeated.	
A character within the arrow means you must separate repeated items with that character.	
An arrow returning to the left above a group of repeatable items indicates that one of the items can be selected, or a single item can be repeated.	

Table 1. Syntax examples (continued)

Item	Syntax example
<p>Fragment.</p> <p>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.</p>	<p>▶▶—KEYWORD—  fragment  ————▶▶</p> <p><b>fragment:</b></p> 



---

# Summary of Changes

## **Summary of Changes for SA22-7619-14 z/OS Version 1 Release 8**

The document contains information previously presented in *z/OS MVS Programming: Workload Management Services, SA22-7619-13* which supports z/OS Version 1 Release 8.

## **Summary of Changes for SA22-7619-13 z/OS Version 1 Release 8**

The document contains information previously presented in *z/OS MVS Programming: Workload Management Services, SA22-7619-12* which supports z/OS Version 1 Release 7. Technical changes include:

- Descriptions of the following five new macros are added: IWMMXDC, IWMMXRA, IWM4MDRG, IWM4MREG, and IWM4HLTH.
- The descriptions of the IWMCLSFY, IWMEQTIME, IWMMEXTR, IWMMNTFY, IWMMRELA, IWMMXFER, IWMRPT, IWMSRSRG, IWM4MCHS, IWM4MCRE, IWM4EDEL, IWM4MINI, and IWM4SRSC macros are updated.

## **Summary of Changes for SA22-7619-12 z/OS Version 1 Release 7**

The document contains information previously presented in *z/OS MVS Programming: Workload Management Services, SA22-7619-11* which supports z/OS Version 1 Release 7.

## **Summary of Changes for SA22-7619-11 z/OS Version 1 Release 7**

The document contains information previously presented in *z/OS MVS Programming: Workload Management Services, SA22-7619-10* which supports z/OS Version 1 Release 7. Technical changes include:

- Performance enhancements of the enclave services. The descriptions of the IWMEBLK, IWMECREA, IWMEGCOR, IWMESTOP, IWMESTRT, IWMEUBLK, IWM4ECRE macros are updated.

## **Summary of Changes for SA22-7619-10 z/OS Version 1 Release 7**

The document contains information previously presented in *z/OS MVS Programming: Workload Management Services, SA22-7619-09* which supports z/OS Version 1 Release 6. Technical changes include:

- A description of the new IWM4SRSC macro is added. The IWM4SRSC service allows to obtain server-specific routing information thus helping to take balanced routing decisions.

- The IWMAEDEF macro is enhanced to support 64-bit environments. The new name is IWM4AEDF.
- The descriptions of the IWMAEDEF, IWMCONN, IWMDISC, IWMECREA, IWMEDELE, IWMMCREA, IWMQDEL, IWMQINS, IWMSLIM, IWMSRSRS, IWMSSEL, IWMSSEM, IWMSTBGN, IWMSTEND, IWMTAFF macros are updated.

**Summary of Changes  
for SA22-7619-09  
z/OS Version 1 Release 6**

The document contains information previously presented in *z/OS MVS Programming: Workload Management Services, SA22-7619-08* which supports z/OS Version 1 Release 6.

**Summary of Changes  
for SA22-7619-08  
z/OS Version 1 Release 6**

The document contains information previously presented in *z/OS MVS Programming: Workload Management Services, SA22-7619-07* which supports z/OS Version 1 Release 6. Technical changes include:

- Enhancements to enclave services to allow work managers to be Enterprise Workload Manger (EWLM) participants. EWLM is a systems service of the IBM Virtualization Engine, and allows system administrators to access and manage resources across multiple platforms. For more information, refer to Chapter 4, “Participating in Enterprise Workload Management,” on page 51.
- The descriptions of the IWMEBLK, IWMEGCOR, IWMESTOP, IWMESTRT, IWMEUBLK, IWM4CON, IWM4DIS, IWM4ECRE, and IWM4EDEL macros are updated.

**Summary of Changes  
for SA22-7619-07  
z/OS Version 1 Release 6**

The document contains information previously presented in *z/OS MVS Programming: Workload Management Services, SA22-7619-06* which supports z/OS Release 5. Technical changes include:

- WLM virtual 64-bit support for Websphere. 15 WLM services are added which can be used in a 31-bit and 64-bit environment. For further information refer to “WLM Services Supporting 64-bit Address Space” on page 14.
- The descriptions of the IWMAEDEF and IWM4QIN macros are updated.

**Summary of Changes  
for SA22-7619-06  
z/OS Version 1 Release 5**

The document contains information previously presented in *z/OS MVS Programming: Workload Management Services, SA22-7619-05* which supports z/OS Version 1 Release 4. Technical changes include:

- The descriptions of the new IWMEBLK, IWMEGCOR, IWMESTOP, IWMESTRT, and IWMEUBLK macros are added.



- The descriptions of the IWMCLSFY, IWMECREA, IWMEDELE, IWMDISC, IWMERES, IWMMCHST, IWMMCREA, IWMMINIT, IWMMNTFY, IWMMRELA, IWMQINS, IWMRCOLL, IWMRESET, IWMSRFSV, IWMRPT, and IWMRQRY macros are updated.

**Summary of Changes  
for SA22-7619-05  
z/OS Version 1 Release 4**

The document contains information previously presented in *z/OS MVS Programming: Workload Management Services, SA22-7619-04* which supports z/OS Version 1 Release 3. Technical changes include:

- The descriptions of the IWMCONN and IWMQINS macros are updated.
- The description of the new IWMAEDEF macro is added. The IWMAEDEF service allows to define dynamic application environments to WLM.

**Summary of Changes  
for SA22-7619-04  
z/OS Version 1 Release 4**

The document contains information previously presented in *z/OS MVS Programming: Workload Management Services, SA22-7619-03* which supports z/OS Release 3. Technical changes include:

- The descriptions of the IWMRCOLL and IWMMCHST macros are updated.
- A number of new SMF type 99 action codes are documented in Appendix A, “SMF Type 99 Action Codes,” on page 707.

**Summary of Changes  
for SA22-7619-03  
z/OS Version 1 Release 4**

The document contains information previously presented in *z/OS MVS Programming: Workload Management Services, SA22-7619-02* which supports z/OS Version 1 Release 3. Technical changes include:

- The description of the IWMCNTN macro is added that allows resource managers to notify WLM work units or transactions involved with resources that are in contention.
- The descriptions of the IWMECREA, IWMEQRY, IWMMCHST, IWMMCREA, IWMMEXTR, IWMMINIT, IWMMNTFY, IWMMRELA, and IWMQINS macros are updated.
- Information is added to indicate this book supports z/OS.e.

This document has been enabled for the following types of advanced searches in the online z/OS Library Center: macros.

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.



---

# Part 1. Using the Workload Management Services



---

## Chapter 1. Introducing the Workload Management Services

The workload management services enable MVS to cooperate with subsystem work managers to achieve installation-defined goals for work, to distribute work across a sysplex, to manage servers and to provide meaningful feedback on how well workload management has achieved those goals. They also allow programs to create an interface to define a service definition.

To change from resource-based performance management to goal-oriented workload management, many transaction managers, data managers, and performance monitors and reporters need to take advantage of the services MVS workload management provides.

This chapter describes the services available for subsystem work managers, performance monitors, and administrative application programs.

---

### Services for Subsystem Work Managers

The workload management services for subsystem work managers allow an installation to process work towards performance goals defined in a service policy. Workload management uses the information provided by the subsystem work managers through the services to match system resources to work to meet goals.

A service policy contains performance goals for all kinds of MVS-managed work expressed in the same terms. A service level administrator identifies and categorizes all of an installation's work and assigns the work performance goals in the workload management service policy. For information about how to set up and use a service policy, see *z/OS MVS Planning: Workload Management*.

The services provide workload management with the information it needs to dynamically adapt to match resources to work to meet the performance goals.

Workload management matches system resources to meet the performance goal assigned to a service class. This management involves handling address space-related resources, such as processor storage, multi-programming level (MPL), dispatching, and I/O queueing.

The services for subsystem work managers fall into the following categories:

- Work manager services
- Execution delay services
- Enclave services
- Queueing manager services
- Routing manager services
- Scheduling environment services
- Sysplex routing services
- Query system information service

### Why Use the Work Manager Services

Work manager services allow MVS to recognize:

- A subsystem work manager and the transactions it processes.
- The service class goals associated with the transactions.
- The address spaces that are processing the transactions.

Based on this information, workload management can determine whether goals are being met, and which work needs resources to meet the goals.

The work manager services allow:

- Your customers to define performance goals to your subsystem work manager's transactions.
- MVS to recognize the goals, and match resources to the work to meet the goals.
- Your customers to get reports from performance monitors like RMF on how well work is executing and whether the goals are being met.

Using the work manager services in your product allows your customers to specify goals for your work the same way they specify them for MVS-managed work.

The work manager services allow workload management to associate incoming work with a service class. When the work is associated with a service class, MVS knows the performance goal and importance level associated with the work, as well as understanding which address spaces are involved in processing the work request.

If your work manager has a client-server structure and has additional objectives such as:

- Dynamic management of server address spaces, or
- Management of server work requests as part of the originating unit of work,
- Resource management and/or reporting of individual requests, or
- Balancing workload among servers across a sysplex,

then consider using either the queueing manager, the routing manager, or enclave services instead of the work manager services.

Table 2 shows a summary of the work manager services.

*Table 2. Work manager services*

<b>Service</b>	<b>Purpose</b>	<b>Information</b>
IWMCLSFY	Associate an arriving work request with a service class defined in a service policy.	Chapter 12, "IWMCLSFY – Classify Work," on page 135
IWM4CON	Obtain token that authorizes caller to use other work manager services, and optionally, to supply additional topology information.	Chapter 65, "IWM4CON – Connecting to Workload Management," on page 533
IWM4DIS	Disconnect from workload management	Chapter 66, "IWM4DIS – Disconnecting from Workload Management," on page 553
IWMWMCON	Override the subsystem name and type previously provided on IWM4CON	Chapter 60, "IWMWMCON – Modify Connect Information," on page 495
IWMWQRY	Obtain a service class goal	Chapter 61, "IWMWQRY – Query Service," on page 503

## Why Use the Execution Delay Monitoring Services

From the execution delay monitoring services, workload management knows how well work is executing, and where any delays are occurring. The execution delay monitoring services are for complex work manager configurations that process across systems in a sysplex, but do not allow MVS to individually manage resource consumption of the transactions. The services allow MVS to recognize additional address spaces that are processing transactions.

When the execution delay monitoring services are used, MVS can allocate resources for address spaces based on the behavior of the transactions being serviced by them. The services also provide execution delay information, so that your customers can determine where work is being delayed. They can then adjust the work manager configuration to consistently meet the goals. Only response time goals can be used with execution delay services. If you need to use velocity goals, discretionary goals, or period switch, consider using enclave services instead. Execution delay monitoring is mutually exclusive with enclaves in the same address space so you must choose whichever function best suits your needs. Enclave services provide more granular resource control and reporting than execution delay monitoring services, but do not provide the capability for the work manager to report its own view of transaction states.

The subsystem work manager uses the execution delay monitoring services to tell workload management about their view of the current state of a work request, such as ready state, idle state, or waiting state. The actual state may be different. For example, a work request may be active from the subsystem's view, but might be delayed by a page fault, or for CPU access. The information is kept in *performance blocks*, also called *monitoring environments*.

The monitoring environments represent work wherever it executes: across multiple dispatchable units, address spaces, and systems.

Table 3 shows a summary of the execution delay monitoring services.

Table 3. Execution delay monitoring services

Service	Purpose	Information
IWMMABNL	Record an abnormal event for work.	Chapter 33, "IWMMABNL – Record Abnormal Event," on page 301
IWM4MCHS	Record the state (such as ready, waiting, idle) of a work request.	Chapter 70, "IWM4MCHS – Monitoring Change State Of Work Unit," on page 587
IWM4MCRE	Create a monitoring environment, also called performance block.	Chapter 71, "IWM4MCRE – Create Delay Monitoring Environment," on page 595
IWMMMDELE	Delete the monitoring environment.	Chapter 34, "IWMMDELE – Delete Monitoring Environment," on page 305
IWMMESTR	Extract information from the monitoring environment	Chapter 35, "IWMMESTR – Monitor Environment Extract Service," on page 309
IWM4MINI	Initialize monitoring environment with information about a work request.	Chapter 73, "IWM4MINI – Initialize Monitoring Environment," on page 611
IWMMNTFY	Record the completion of an execution phase of processing a work request.	Chapter 36, "IWMMNTFY – Notify of Work Execution Completion," on page 317
IWMMRELA	Relate two monitoring environments that are associated with the same work request.	Chapter 37, "IWMMRELA – Relate Monitoring Environment Service," on page 327
IWMMSWCH	Record that the information for a work request may be reflected in another monitoring environment.	Chapter 38, "IWMMSWCH – Switch Monitoring Environment," on page 337
IWMXFER	Record that information for a work request is reflected in another related monitoring environment.	Chapter 40, "IWMXFER – Transfer Monitoring Environment," on page 349
IWMRPT	Report the response time of a completed work request.	Chapter 46, "IWMRPT – Report on Work Request Completion," on page 393
IWMWQWRK	Identify where transactions are executing.	Chapter 62, "IWMWQWRK – Query Work Service," on page 507

## Why Use the Enclave Services

An **enclave** is an anchor for a transaction that can be spread across multiple dispatchable units in multiple address spaces. These multiple address spaces can even span across multiple systems in a parallel sysplex. The value of using an enclave to represent a transaction is that the resources used to process the transaction can be accounted to the transaction itself, rather than to the address space or spaces that the transaction runs in. In addition, you can assign a performance goal to the enclave, which means that as a transaction consumes system resources, it can switch periods to run with a new goal.

Any number of tasks and SRBs can be grouped together in an enclave:

- Enclave SRBs offer the advantage that they are preemptable and won't tie up the system.

SRBs in enclaves work well for higher volume, small requests, as SRBs have very little overhead compared to tasks. The subsystem can create an enclave using the IWM4ECRE macro, and then schedule SRBs to run in the enclave using the IEAMSCHD macro.

- Tasks in enclaves automatically associate the enclave with the address spaces where they are dispatched, so workload management can manage the storage of those address spaces to meet the goal of the enclave. The enclave can perform functions that require a task environment, such as supervisor calls. Tasks can dynamically leave and join an enclave as they finish one piece of work and begin another. The subsystem creates an enclave using the IWM4ECRE macro, and then the task joins the enclave using the IWMEJOIN macro.

### Comparison to Other Services

Enclaves should not be run in the same address space with execution delay monitoring environments. Unpredictable workload management actions could result.

Although enclaves have some characteristics that are similar to those of execution delay monitoring, there are some important differences. These differences should be considered before choosing which set of services to use. Enclaves support all types of performance goals; delay monitoring supports only response time goals. Enclaves allow period switching; delay monitoring does not. Enclaves can span address spaces in multiple systems in a parallel sysplex; delay monitoring cannot. Enclaves allow the separate management of work units that run in the same address space but have different performance goals; a delay monitoring environment can only be managed at the address space level. One advantage of delay monitoring is that it does enable a work manager to report its own view of transaction states.

Enclaves are required to be used with queueing manager services. For more information, see “Why Use the Queueing Manager Services” on page 8.

Table 4 shows a summary of the enclave services.

Table 4. Enclave services

Service	Purpose	Information
IEAMSCHD	Schedule an SRB into the enclave.	<i>z/OS MVS Programming: Authorized Assembler Services Guide</i>
IWM4ECRE	Create an enclave.	Chapter 67, “IWM4ECRE – Create an Enclave,” on page 561



Table 4. Enclave services (continued)

Service	Purpose	Information
IWMECQRY	Query the classification information associated with an enclave.	Chapter 19, "IWMECQRY – Query Enclave Classification Attributes," on page 203
IWM4EDEL	Delete a previously created enclave.	Chapter 68, "IWM4EDEL – Delete an Enclave," on page 573
IWMEDREG	Deregister an enclave	Chapter 20, "IWMEDREG — WLM Enclave Deregister Service," on page 209
IWMEJOIN	Join an enclave (task only). Once a task has joined an enclave, all future processing is on behalf of the transaction represented by the enclave.	Chapter 22, "IWMEJOIN – Join an Enclave," on page 223
IWMELEAV	Leave an enclave (task only).	Chapter 23, "IWMELEAV – Leave an Enclave," on page 229
IWMEREG	Register an enclave	Chapter 25, "IWMEREG — WLM Enclave Register Service," on page 241
IWMESQRY	Query whether or not the current dispatchable unit is associated with an enclave.	Chapter 29, "IWMESQRY – Query Enclave State," on page 271
IWMEXPT	Export an enclave to all systems in a parallel sysplex.	Chapter 31, "IWMEXPT – WLM Export Service," on page 285
IWMIMPT	Import an enclave that has been exported to all systems in a parallel sysplex.	Chapter 32, "IWMIMPT – WLM Import Service," on page 293
IWMRQRY	Obtain information about enclave resource consumption and delays.	Chapter 47, "IWMRQRY – Collecting Address Space Delay Information," on page 405
IWMUEXPT	Undo an export of an enclave to all systems in a parallel sysplex.	Chapter 58, "IWMUEXPT – WLM Undo Export Service," on page 483
IWMUIMPT	Undo an import of an enclave.	Chapter 59, "IWMUIMPT – WLM Undo Import Service," on page 489
SYSEVENT ENCASSOC	Indicate what address space an enclave is associated with so the address space can be managed to the performance goals of the enclave.	<i>z/OS MVS Programming: Authorized Assembler Services Reference SET-WTO</i>
SYSEVENT ENCSTATE	Indicate that an enclave will be idle for an extended period of time, exempting the enclave from active resource management.	<i>z/OS MVS Programming: Authorized Assembler Services Reference SET-WTO</i>

## Enterprise Workload Manager (EWLM)

With enclave services, work managers that use enclaves to manage and report individual business transactions can become Enterprise Workload Manager (EWLM) participants. EWLM allows you to define business-oriented performance goals for an entire domain of servers across a variety of platforms (z/OS, AIX, i5/OS, Windows, Solaris and Linux), and then get an end-to-end view of actual performance relative to those goals. You can use enclave services to:

- Connect a calling address space to WLM as an EWLM participant
- Classify work requests in order to assign an EWLM transaction class
- Indicate the start and end of an EWLM work request
- Block work requests while waiting for the completion of a work request in another application.

## Why Use the Queueing Manager Services

A **queueing manager** is a subsystem or application that queues work requests to workload management for execution by one or more server address spaces.

Queueing manager services allow MVS to:

- Dynamically start and stop server address spaces based on workload.
- Control the number of server instances per server address space.
- Manage the work queues associated with the server address spaces to meet the performance goals set by the customer.

With the dynamic management of server address spaces, an installation does not need to calculate the proper number of address spaces to process work, nor do they have to monitor workload fluctuations that change the number of address spaces needed. Customers can segregate work requests into different server address spaces if this is important for security or integrity.

Enclaves are required to be used with the queueing manager services. This means that customers can define velocity and discretionary goals for work as well as response time goals. Multiple period control is also available for work running in enclaves.

Queueing manager services may provide incentive to subsystems who run with multiple tasks in one address space to switch to multiple address spaces. The queueing manager services make it easier for installations to isolate individual work requests from each other, by running only one in each execution address space, with workload management managing the number of execution address spaces.

Table 5 shows a summary of the queueing manager services.

*Table 5. Queueing manager services*

Service	Purpose	Information
IWM4AEDF	Define a dynamic application environment	Chapter 64, "IWM4AEDF — Defining Dynamic Application Environments to Workload Management," on page 523
IWM4CON	With the QUEUE_MANAGER=YES parameter, establish the caller as a queueing manager so it can begin queueing work requests to its server address spaces.	Chapter 65, "IWM4CON – Connecting to Workload Management," on page 533
IWM4CON	With the SERVER_MANAGER=YES parameter, establish the caller as a server address space so it can begin receiving work requests from the queueing manager.	Chapter 65, "IWM4CON – Connecting to Workload Management," on page 533
IWM4DIS	Remove the caller as a queueing manager or server manager.	Chapter 66, "IWM4DIS – Disconnecting from Workload Management," on page 553
IWM4ECRE	Create an enclave. This can be done by the queue manager itself or by its caller. It can be a dependent or independent enclave.	Chapter 67, "IWM4ECRE – Create an Enclave," on page 561
IWM4EDEL	Delete an enclave.	Chapter 68, "IWM4EDEL – Delete an Enclave," on page 573
IWMESQRY	Query whether or not the current dispatchable unit is associated with an enclave.	Chapter 29, "IWMESQRY – Query Enclave State," on page 271

Table 5. Queueing manager services (continued)

Service	Purpose	Information
IWM4QDE	Delete a work request from the queue for an execution address space.	“IWMQDEL – Deleting a Request from the Queue for An Execution Address Space” on page 846
IWM4QIN	Insert a work request onto workload management queues so its execution in a server address space can be managed by workload management. The enclave token obtained with the IWM4ECRE service is passed into workload management by IWM4QIN.	Chapter 76, “IWM4QIN – Inserting a Request Onto the Queue for An Execution Address space,” on page 641
IWMSINF	Obtain the number of server instances to be started by workload management.	Chapter 52, “IWMSINF — WLM Server Manager Inform Service,” on page 437
IWM4SLI	Immediately after invoking IWM4CON, optionally establish a maximum and/or minimum number of server instances that can be started for a given application environment.	Chapter 77, “IWM4SLI – Application Environment Limit Service,” on page 653
IWM4SSL	Select a work request from workload management queues for execution in a server address space. This must be done under a task.	Chapter 79, “IWM4SSL – Selecting a Request from a Caller’s Work Manager Queue,” on page 669
IWM4SSM	Select the next secondary work request from the queue associated with the caller’s server task.	Chapter 80, “IWM4SSM – WLM Server Select Secondary Service,” on page 677
IWM4STBG	Join the invoking task to the enclave associated with the work request represented by WUTOKEN (which was obtained on a prior call to IWM4SSL) and optionally check the authorization of the request. The server address space is beginning to process the work request. This must be done under a task.	Chapter 81, “IWM4STBG – Beginning a Request from a Caller’s Work Manager Queue,” on page 683
IWM4STEN	Leave the enclave that was joined in IWM4STBG. The server address space has completed its processing of the work request.	Chapter 82, “IWM4STEN – End a Request from a Caller’s Work Manager Queue,” on page 691
IWM4TAF	Tell workload management when a temporal affinity begins and when it ends.	Chapter 83, “IWM4TAF — WLM Temporal Affinity Service,” on page 697

## Why Use the Routing Manager Services

A **routing manager** is a subsystem that establishes and manages connections between a client and a server address space.

Routing manager services perform two main functions:

- Automatically starting and maintaining server address spaces as needed by the workload across the sysplex.
- Balancing the workload among the servers in the sysplex by deciding on the best server and providing the server routing information when a server is requested by the routing manager.

Table 6 on page 10 shows a summary of the routing manager services.

Table 6. Routing manager services

Service	Purpose	Information
IWM4CON	With the ROUTER=YES parameter, establish the caller as a routing manager so it can begin requesting server routing information through IWMSRFSV.	Chapter 65, "IWM4CON – Connecting to Workload Management," on page 533
IWM4CON	With the SERVER_MANAGER=YES and SERVER_TYPE=ROUTING parameters, establish the caller as an eligible server for requests coming from a routing manager. Workload management will balance the workload among the eligible servers.	Chapter 65, "IWM4CON – Connecting to Workload Management," on page 533
IWM4DIS	Remove the caller as a routing manager.	Chapter 66, "IWM4DIS – Disconnecting from Workload Management," on page 553
IWM4ECRE	Create an enclave. This is done in a server address space.	Chapter 67, "IWM4ECRE – Create an Enclave," on page 561
IWM4EDEL	Delete an enclave.	Chapter 68, "IWM4EDEL – Delete an Enclave," on page 573
IWMSRFSV	Find the best server for a work request. If no server exists for a request, start one.	Chapter 55, "IWMSRFSV – Finding a Sysplex Routing Server," on page 457

## Why Use the Scheduling Environment Services

A scheduling environment is a list of resource requirements, allowing you to ensure that units of work are sent to systems that have the appropriate resources to handle them. Resources can represent actual physical entities, such as a data base or a peripheral device, or they can represent intangible qualities such as a certain period of time (like second shift or weekend).

These resources are listed in the scheduling environment according to whether they must be set to ON or set to OFF. A unit of work can be assigned to a specific system only when all of the required resource states are satisfied.

Table 7 shows a summary of the scheduling environment services.

Table 7. Scheduling Environment Services

Service	Purpose	Information
IWMSEDES	Determine if a scheduling environment is available on a specified system	Chapter 48, "IWMSEDES – Scheduling Environments Determine Execution Service," on page 413
IWMSEQRY	Obtain scheduling environment definitions and status	Chapter 49, "IWMSEQRY – Scheduling Environments Query Service," on page 419
IWMSESET	Modify the state setting of a resource	Chapter 50, "IWMSESET – Scheduling Environments Set Resource," on page 425
IWMSEVAL	Validate a scheduling environment name	Chapter 51, "IWMSEVAL – Scheduling Environments Validate Service," on page 431

## Why Use the Sysplex Routing Services

The sysplex routing services allow work associated with a server to be distributed across a sysplex. They are intended for use by clients and servers when the incoming work requests have not been classified by workload management at the time the routing decision is being made.

The sysplex routing services enable distributed client/server environments to balance work among multiple servers. These services help distributed programs make the routing decisions, rather than having each installation make these decisions. Unlike the routing manager services described earlier, sysplex routing services do not automatically start server address spaces as needed.

A client is any subsystem work manager, application or product, in the network that requests a service. The service could be a request for data, a program to be run, or access to a database or application. In terms of the sysplex routing services, a client is any program routing work to a server. A server is any subsystem address space that provides a service on an MVS image.

The sysplex routing services provide information for more intelligent routing. They do not route or distribute work requests. The server must use its existing routing mechanisms.

Table 8 shows a summary of the sysplex workload balancing services.

*Table 8. Sysplex routing services*

Service	Purpose	Information
IWMSRDNS	Provide the caller with list of location names for all registered servers known to the system on which the service is invoked.	Chapter 53, "IWMSRDNS – Get Sysplex Routing Location List," on page 445
IWMSRDRS	Deregister a server	Chapter 54, "IWMSRDRS – Deregister a Server for Sysplex Routing," on page 451
IWMSRSRG	Register an eligible server.	Chapter 56, "IWMSRSRG – Register a Server for Sysplex Routing," on page 465
IWMSRSRS	Provide the caller with a list of registered servers and the number of requests that should be routed to each server.	Chapter 57, "IWMSRSRS – Determine Where to Route Work," on page 473
IWM4SRSC	Provide the caller with server-specific routing information to allow for balanced routing decisions.	Chapter 78, "IWM4SRSC – Obtain Server-Specific Routing Information," on page 661

## Why Use the Query System Information Service

The Query System Information service, IWMWSYSQ, returns a list of systems running in goal mode and information related to available CPU capacity and resource constraints. Applications that schedule work across multiple systems in an MVS sysplex can use this service to:

- Locate the "best" (fastest or most idle) system in a sysplex for scheduling specific work
- Avoid scheduling additional work to systems already critically overloaded
- Factor workload management business importance level information into scheduling decisions.

The output of this service is a data area mapped by the IWMWSYSI macro, that provides a point-in-time snapshot of each system workload management is managing in the sysplex. A scheduling application can interpret and use this information to schedule one or more types of work to systems with specific operating characteristics.

Refer to Chapter 63, "IWMWSYSQ – Querying System Information," on page 515 for more information about this service.

---

## Services for Performance Monitors

The workload reporting services are intended for use by monitoring or reporting products to collect performance data. These services replace some of the existing methods of collecting data, and provide as complete a picture of performance information as possible.

A workload management ISPF application contains an installation's goals for work in a service policy. The reporting services access the service policy information, and report on how well the installation is doing in processing towards the goals in the policy. The services report information based on the service classes defined in the service policy. They also provide delay information on work managed by subsystems using the execution delay monitoring services.

Because the system collects performance data continually, there is no set reporting interval. So, unlike earlier releases of MVS, multiple performance monitors can request the services at the same time. And, performance monitors can collect the data based on their own reporting intervals. When the performance monitor invokes a service to collect performance data, the data is provided in a cumulative fashion.

When a significant change occurs in workload management, such as a policy activation, the data collection is stopped and re-started. At such times, performance monitors should also stop and re-start their reporting intervals. For each time that data collection is stopped and re-started in workload management, an ENF signal notifies listeners of the change.

## Why Use the Workload Reporting Services

The workload reporting services provide information for performance monitors to report on how well an installation is doing in meeting performance goals.

Prior to z/OS R3, workload reporting services were available to systems running in either goal or compatibility mode. Some of the collected data was different for each mode. The performance monitor should realize the system will now be running exclusively in goal mode, and be able to locate and use the collected performance data appropriately.

For goal mode, with the cooperation of subsystem work managers, the service can provide more performance data than previously reported. They provide information about work that is processed by many address spaces, and allow for a view of subsystem transactions, not just address spaces and enclaves. The data includes:

- Response time information
- Response time distributions
- Execution delay state information for transactions
- Information about service classes that different address spaces are serving.

The services allow a performance monitor to show the goal for a service class period, how well the system is doing to meet the goal, and if it is not meeting the goal, why it is delayed. The performance monitor can show this goal vs. actual data in terms that are consistent for all MVS-managed work.

Table 9 on page 13 shows an overview of the workload reporting services, and where the information about them is documented.

Table 9. Summary of workload reporting services

Service	Purpose	Information
IWMPQRY	<ul style="list-style-type: none"> <li>Provides the active service policy.</li> <li>Use it with IWMRCOLL for goal vs. actual information</li> </ul>	Chapter 43, "IWMPQRY – Query Active Service Policy," on page 371
IWMRCOLL	Collects: <ul style="list-style-type: none"> <li>Workload activity information</li> <li>Response time information</li> <li>General delay information</li> <li>Execution delay state information</li> </ul>	Chapter 44, "IWMRCOLL – Collecting Workload Activity Data," on page 377
IWMRQRY	Provides address space related information: <ul style="list-style-type: none"> <li>Server information</li> <li>Velocity information</li> <li>General delay information               <ul style="list-style-type: none"> <li>MPL delay</li> <li>Swap-in</li> <li>Resource group capping</li> <li>CPU delay</li> </ul> </li> <li>Enclave information</li> </ul>	Chapter 47, "IWMRQRY – Collecting Address Space Delay Information," on page 405
SYSEVENTs REQASD and REQFASD	Provide information about an address space: <ul style="list-style-type: none"> <li>Whether it is a server</li> <li>Whether its goal is being honored</li> <li>Whether it was quiesced</li> <li>Service class, report class</li> <li>Performance group, report performance group</li> </ul>	<i>z/OS MVS Programming: Authorized Assembler Services Reference SET-WTO</i>
SYSEVENT REQSRMST	To quickly check: <ul style="list-style-type: none"> <li>Active service policy</li> <li>Installed service definition</li> </ul>	<i>z/OS MVS Programming: Authorized Assembler Services Reference SET-WTO</i>

## Getting Information from SMF Type 99

SMF record type 99 provides detailed audit information. You can use the type 99 records for analyzing performance characteristics of work. The records contain performance data for each service class period, a trace of SRM actions, the data SRM used to decide which actions to take, and the internal controls SRM uses to manage work.

This can help you determine in detail what SRM is doing to meet your work's goals with respect to other work, and the types of delays the work is experiencing.

### Attention:

Be aware that the SMF type 99 records are written frequently. The SMF type 99 records are for detailed audit information only. Before migrating to z/OS V1R3 and switching your systems into goal mode, you should make sure you do not write SMF type 99 records unless you want them.

Chapter 11, "Using SMF Record Type 99," on page 113 explains how to use the information provided in SMF type 99 records. For a mapping of the records, see *z/OS MVS System Management Facilities (SMF)*.

---

## Services for Application Programs

The administrative application services are intended for programs which provide a user interface to define and edit a workload management service definition.

Table 10 shows a summary of the administrative application services.

Table 10. Administrative application services

Service	Purpose	Information
IWMCQRY	Query the classification rules in effect.	Chapter 15, "IWMCQRY – Query Classification Attributes," on page 169
IWMDINST	Install a service definition on the WLM couple data set.	Chapter 17, "IWMDINST – Install Service Definition," on page 185
IWMDEXTR	Extract a service definition from the WLM couple data set.	Chapter 16, "IWMDEXTR – Extract Service Definition," on page 177
IWMPACT	Activate a service policy.	Chapter 42, "IWMPACT – Activate Service Policy," on page 363

---

## WLM Services Supporting 64-bit Address Space

Several WLM services support 64-bit environments. These services run in both, 31-bit and 64-bit address mode.

To use 64-bit services, change the names of the services in your application, for example, change IWMCONE to IW4CON. The prefix of all 64-bit services names is IW4.

The services that run in 64-bit address mode in general support the same parameters as their equivalents for 31-bit address mode. Note that the only exception is the PLISTVER parameter, which has slightly changed. The 64-bit services only support PLISTVER=0, in case a PLIST Version is explicitly used. The following example shows how to use the PLISTVER keyword for 31-bit services:

```
...
12345678 SPACE 1 DS 0H
          IWMxxxxx ETOKEN=ETOKEN
          RSNCODE=RSNCODE,
          PLISTVER=2
```

```
...
ETOKEN  DS    F
RSNCODE DS    F
```

where xxxxx is the name of the 31-bit service.

The following example shows how to use the PLISTVER keyword for 64-bit services:

```
...
12345678 SPACE 1 DS 0H
          IW4xxxxx ETOKEN=ETOKEN
          RSNCODE=RSNCODE,
          PLISTVER=0
```

```
...
ETOKEN  DS    F
RSNCODE DS    F
```



where xxxx is the name of the 64-bit service.

Table 11 lists the WLM services supporting 64-bit address space and their equivalents for 31-bit address space only:

Table 11. Overview of WLM services supporting 64-bit and 31-bit address space

WLM Service Name (31-bit only)	WLM Service Name (31-bit and 64-bit)	Purpose
IWMAEDEF. For further information refer to “IWMAEDEF — Defining Dynamic Application Environments to Workload Management” on page 765.	IWM4AEDF. For further information refer to Chapter 64, “IWM4AEDF — Defining Dynamic Application Environments to Workload Management,” on page 523.	Defining a dynamic application environment to Workload Manager
IWMCONN. For further information refer to Appendix D, “WLM Services supporting 31-bit address space only,” on page 765.	IWM4CON. For further information refer to Chapter 65, “IWM4CON – Connecting to Workload Management,” on page 533.	Connecting to Workload Manager
IWMDISC. For further information refer to Appendix D, “WLM Services supporting 31-bit address space only,” on page 765.	IWM4DIS. For further information refer to Chapter 66, “IWM4DIS – Disconnecting from Workload Management,” on page 553.	Disconnecting from Workload Manager.
IWMECREA. For further information refer to Appendix D, “WLM Services supporting 31-bit address space only,” on page 765.	IWM4ECRE. For further information refer to Chapter 67, “IWM4ECRE – Create an Enclave,” on page 561.	Creating an enclave
IWMEDELE. For further information refer to Appendix D, “WLM Services supporting 31-bit address space only,” on page 765.	IWM4EDEL. For further information refer to Chapter 68, “IWM4EDEL – Delete an Enclave,” on page 573.	Deleting an enclave
IWMMCHST. For further information refer to Appendix D, “WLM Services supporting 31-bit address space only,” on page 765.	IWM4MCHS. For further information refer to Chapter 70, “IWM4MCHS – Monitoring Change State Of Work Unit,” on page 587.	Changing state of work request service
IWMMCREA. For further information refer to Appendix D, “WLM Services supporting 31-bit address space only,” on page 765.	IWM4MCRE. For further information refer to Chapter 71, “IWM4MCRE – Create Delay Monitoring Environment,” on page 595.	Create monitoring environment service
IWMMINIT. For further information refer to Appendix D, “WLM Services supporting 31-bit address space only,” on page 765.	IWM4MINI. For further information refer to Chapter 73, “IWM4MINI – Initialize Monitoring Environment,” on page 611.	Monitoring environment initialization
IWMQDEL. For further information refer to Appendix D, “WLM Services supporting 31-bit address space only,” on page 765.	IWM4QDE. For further information refer to Chapter 75, “IWM4QDE – Deleting a Request from the Queue for An Execution Address Space,” on page 635.	Deleting a request from the queue for an execution address space
IWMQINS. For further information refer to Appendix D, “WLM Services supporting 31-bit address space only,” on page 765.	IWM4QIN. For further information refer to Chapter 76, “IWM4QIN – Inserting a Request Onto the Queue for An Execution Address space,” on page 641.	Inserting a request to the queue for an execution address space
IWMSLIM. For further information refer to Appendix D, “WLM Services supporting 31-bit address space only,” on page 765.	IWM4SLI. For further information refer to Chapter 77, “IWM4SLI – Application Environment Limit Service,” on page 653.	Application environment limit service
–	IWM4SRSC. For further information refer to Chapter 78, “IWM4SRSC – Obtain Server-Specific Routing Information,” on page 661.	Obtaining server-specific routing information
IWMSSEM. For further information refer to Appendix D, “WLM Services supporting 31-bit address space only,” on page 765.	IWM4SSM. For further information refer to Chapter 80, “IWM4SSM – WLM Server Select Secondary Service,” on page 677.	WLM server select secondary service
IWMSTBGN. For further information refer to Appendix D, “WLM Services supporting 31-bit address space only,” on page 765.	IWM4STBG. For further information refer to Chapter 81, “IWM4STBG – Beginning a Request from a Caller’s Work Manager Queue,” on page 683.	Beginning a request from a caller’s work manager queue

Table 11. Overview of WLM services supporting 64-bit and 31-bit address space (continued)

WLM Service Name (31-bit only)	WLM Service Name (31-bit and 64-bit)	Purpose
IWMSTEND. For further information refer to Appendix D, "WLM Services supporting 31-bit address space only," on page 765.	IWM4STEN. For further information refer to Chapter 82, "IWM4STEN – End a Request from a Caller's Work Manager Queue," on page 691.	End a request from a caller's work manager queue
IWMTAFF. For further information refer to Appendix D, "WLM Services supporting 31-bit address space only," on page 765.	IWM4TAF. For further information refer to Chapter 83, "IWM4TAF — WLM Temporal Affinity Service," on page 697.	WLM temporal affinity service

---

## Chapter 2. Using The Subsystem Work Manager Services

You can use many different combinations of the workload management services. Which ones and which combination you choose to use depends on the benefit you expect from using them, whether your programming environment allows you to use them, and the structure of the subsystem work manager using the services. The following section describes some suggested uses of the MVS workload management services.

If you need to manage a transaction separately from the address space in which it runs, or you want to use velocity goals, discretionary goals, or period control, use enclaves. For more information about using enclave services, see Chapter 3, “Creating and Using Enclaves,” on page 31.

---

### Considerations Before Using the Services

Before you use the subsystem work manager and the execution delay monitoring services, you should consider the following:

- What “type” of subsystem work manager you are.
  - Transaction processing system
  - Data or resource manager
- What benefits do you expect to reap from using the WLM services?
  - Do you want your customers to be able to specify goals for your transactions?
  - Do you plan on getting reporting information on goals defined in the MVS workload management service definition?
- What kinds of address spaces does the subsystem work manager consist of?
- What is the definition of a transaction or work request, from your customer’s perspective?
- What kind of functions do the address spaces provide?
  - Control address spaces?
  - Transaction level dispatching?
  - Other execution regions?
  - Other supporting address spaces?
- What environments does the subsystem work manager run in?
  - Authorization, including PSW key
  - Dispatchable unit mode
  - Cross memory mode
  - AMODE
  - ASC mode
  - Interrupt status
  - Locks
- Does the subsystem work manager use other products, such as a data manager?
- Does the work cross MVS system (MVS image) boundaries?
- Does the work cross MVS sysplex boundaries?

---

## Suggested Services for a Single Address Space Transaction Manager

A single address space subsystem work manager performs functions similar to those shown in Figure 1. It goes through address space initialization and start-up routines. It receives a work request, processes the work request, receives another work request, and so on. At some point, it processes some address space clean-up and termination routines.

### Single Address Space Transaction Manager

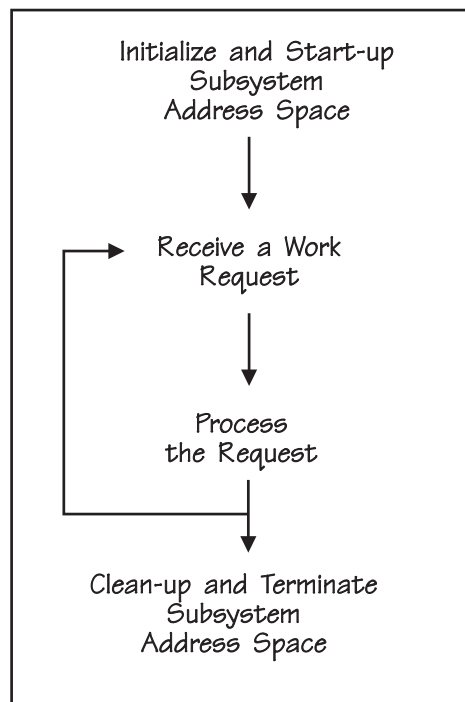


Figure 1. Sequence of functions in a single address space transaction manager.

If you have a single address space transaction manager, and would like the following functions from workload management:

- Associate work coming into the subsystem with a service class.
- Have MVS match resources to the work to meet the service class goal.
- Provide goal vs. actual information for reporting
- Provide response time information for work requests

consider using the services shown in Figure 2 on page 19. They show when a single address space manager could invoke the appropriate workload management services. They are the minimum set of services a work manager can use to achieve the objectives listed above.

**Note:** You can instead use enclave services for a single address space transaction manager. For more information about using enclave services, see Chapter 3, “Creating and Using Enclaves,” on page 31.

## Single Address Space Transaction Manager Work Manager Services

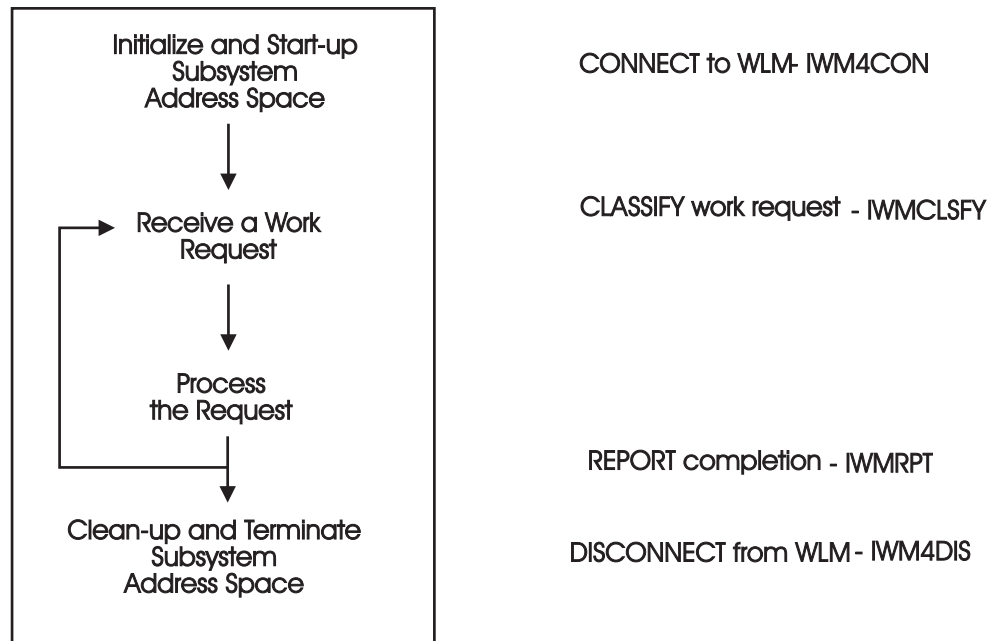


Figure 2. Work manager services for a single address space transaction manager.

You issue the IWM4CON service at address space initialization time. This connect service returns a token required by the IWMCLSFY and IWMRPT services. When the address space receives a work request, it should issue the classify (IWMCLSFY) macro to associate arriving work with a service class. The subsystem work manager can issue IWMCLSFY in either problem state or supervisor state, in any PSW key. The PSW key, however, must be compatible with the key specified when IWM4CON was issued.

IWMCLSFY passed WLM information identifying the work request. This information includes the following:

- Subsystem environment and name (used on IWM4CON)
- Transaction/job name
- Transaction/job class
- Userid
- Accounting information
- LU name
- Network id

These are called work *qualifiers*. IWMCLSFY also supports a product-specific parameter, called the subsystem parameter. If none of the qualifier types listed above define your subsystem work manager's work requests, you can use the subsystem parameter.

The transaction manager should document which work qualifiers they use on the IWMCLSFY service, so that a customer knows how to define the *classification rules* defined in the workload management ISPF application. Also, since your transaction manager may not support all of the qualifier types, you should recommend that your customers customize the list of qualifiers for your subsystem type in the WLM ISPF application.

After receiving the work request and classifying it, the transaction manager then processes the work request. When it completes the request, it should issue IWMRPT. The report service provides the arrival time and, optionally, completion information about the work request. Only normal completions are included in the response time information, so the information is not skewed by abnormal completions. The transaction manager should issue IWMRPT only once per work request.

Then, at address space clean up and termination time, the transaction manager should issue a IWM4DIS, to disconnect from workload management services.

## Using the Execution Delay Monitoring Services

If you have a single address space subsystem work manager, and would like the following functions from workload management:

- Associate work coming into the subsystem with a service class
- Goals vs. actual information for reporting
- Response time information for work requests
- Execution delay information about work for reporting and for MVS management purposes

consider using the services shown in Figure 3 on page 21.

The execution delay monitoring services support response time goals only. If you want your customers to assign velocity or discretionary goals, or if you want to support period switching, consider using enclaves instead. For information about using enclave services, see Chapter 3, “Creating and Using Enclaves,” on page 31

It is important to use the IWM4CHST service together with the IWMRPT service, otherwise the delay information is not meaningful.

## Single Address Space Transaction Manager Work Manager & Delay Monitoring Services

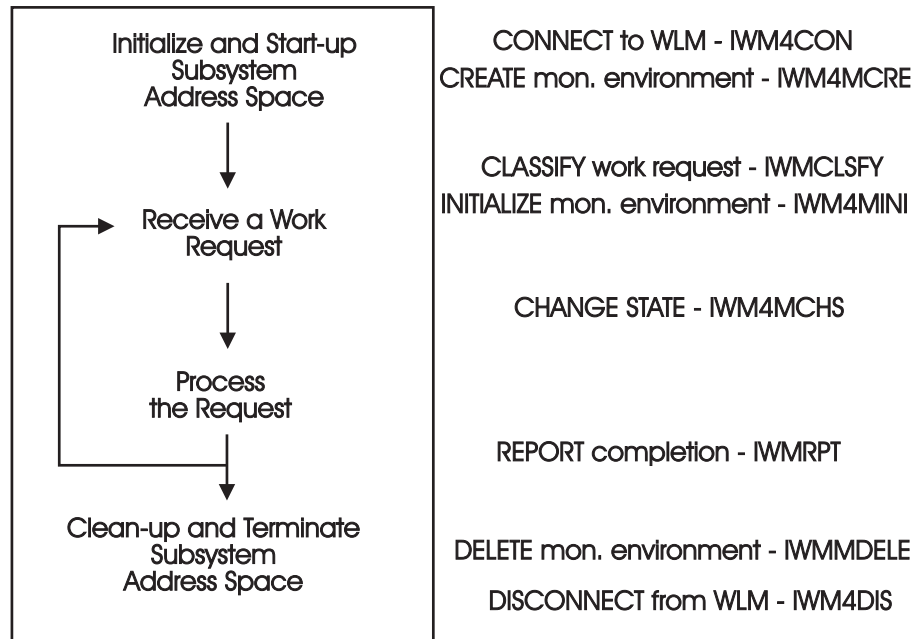


Figure 3. Work manager and delay monitoring services for a single address space transaction manager.

At address space initialization, the address space issues the IWM4CON service to establish authorization for subsequent services. It then issues the IWM4MCRE (create) service. Create establishes a monitoring environment to keep track of the execution delays encountered by a work request. If the transaction manager sets up multiple tasks to process work, you should create one monitoring environment per task, assuming each task is dedicated to one work request. Similarly, if the task processes multiple work requests at the same time, then it should issue one IWM4MCRE for each work request that may be running under that task at one time. You can use the REQUEST=MULTIPLE parameter on the IWM4MCRE service to create a pool of monitoring environments at initialization time. This saves the repeated system overhead of issuing a single IWM4MCRE service for each monitoring environment needed.

The IWM4MCRE service also defines the PSW key in which the transaction manager is to run. Since monitoring environments are not initially associated with a work request, the IWM4MCRE sets the state of the monitoring environment to “free”.

When you create a monitoring environment, workload management establishes recovery at both the task and address space level. If the address space or the task that created the monitoring environment fails, workload management cleans up the resources associated with the monitoring environment.

When the transaction manager receives a work request, it should issue the IWMCLSFY service to associate an incoming work request with a service class. At that time, the transaction manager should issue an IWM4MINI to initialize the monitoring environment. The IWM4MINI service with the MODE=RESET parameter sets the state of the monitoring environment to “active”, and associates the monitoring environment with the work request.

Whenever that work request encounters a different state, such as waiting on a conversation, waiting on a lock, or for I/O, the transaction manager should issue the IWM4MCHS (change state of work request) service. Since IWM4MCHS is an inline expansion, there is very little overhead, and you can issue it frequently. Workload management can then update the monitoring environment to reflect these changes, and represent the execution delays the work request encountered.

When the transaction manager has completed processing the work request, it should issue the IWMRPT service. The transaction manager should delete all created monitoring environments at address space clean-up and termination, and disconnect from workload management services.

## Suggested Services for a Work Manager Calling a Data Manager

If you have a work manager that calls a data manager, and you are on the same MVS image, you could use the combination of services suggested in this section. If you would like to do the following:

- Associate work coming into the subsystem with a service class
- Goals vs. actual information for reporting
- Response time information for work requests
- Execution delay information about work
- Track work from a work manager to a data manager

consider using the services shown in Figure 4.

### Single Address Space Transaction Manager Work Manager TCB calls Database Manager

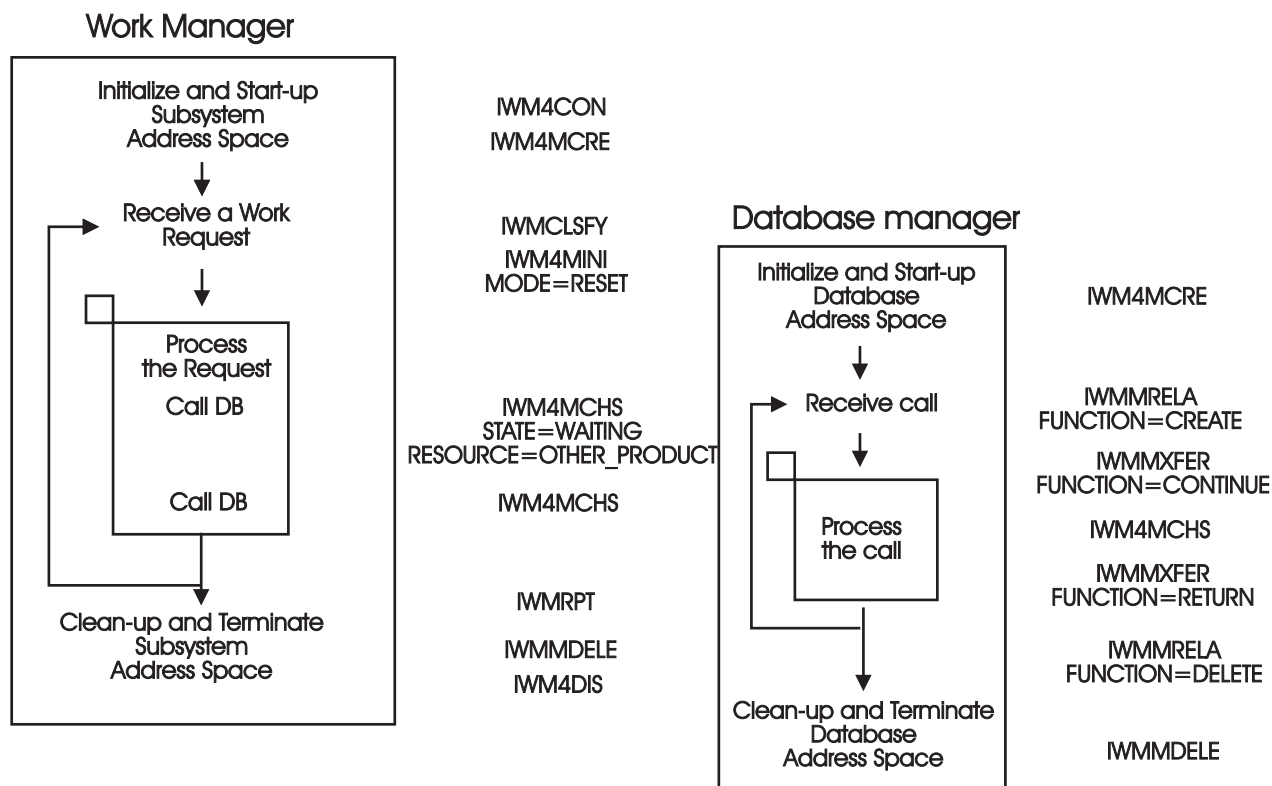


Figure 4. Services for a work manager that uses a database manager.



Since the transaction manager is using the work manager services, it must issue the IWM4CON at address space initialization. In this example, the transaction manager is using monitoring environments, so it issues an IWM4MCRE at address space initialization time to create the monitoring environment. Similarly, the database manager is also using monitoring environments, so it issues IWM4MCRE at its address space initialization.

Assuming that the database manager uses a dedicated dispatchable unit, it should also create one monitoring environment per task or SRB that it uses. The transaction manager and the database manager could be running in different, or the same tasks. The tasks in the figure (one in the work manager, and one in the database manager) could be the same one.

However, the database manager, instead of using IWM4MINI, issues IWMMRELA (relate monitoring environment) when it is called by the transaction manager task. The relate monitoring environment service associates the database manager's monitoring environment for the work request to the transaction manager's monitoring environment for the same work request. Because the relate service requires the token and key identifying the transaction manager's monitoring environment, the transaction manager should pass the token and key to the database manager. In this example, the transaction manager monitoring environment is called the *parent* environment, and the database manager that issues the relate service is called the *dependent* environment.

After the database manager has related to the work manager, it can be called to process a specific data base request. Each such call should begin with a transfer (IWMMXFER) service with a FUNCTION=CONTINUE parameter, and end with a transfer (IWMMXFER) with a FUNCTION=RETURN parameter.

When the work manager issues a call to the database manager, the database manager issues a IWMMXFER. The work manager state can be either active, or waiting throughout the transfer. The IWMMXFER FUNCTION=CONTINUE indicates that the real state for the work request now resides in the data manager monitoring environment. From that point on, the database manager should use change state (IWM4MCHS) as its view of the work request changes. You should issue IWMMRELA and IWMMXFER in pairs for each data base call or return.

The transfer with the FUNCTION=RETURN parameter resets the dependent monitoring environment state to free. At this point, workload management recognizes that the dependent monitoring environment no longer represents the work request. The parent and dependent monitoring environments are still related, for any future transfers, such as with a second call to the database manager. When the database manager is done with any work requests that require related monitoring environments between the parent and the dependent, it should issue the IWMMRELA with the FUNCTION=DELETE parameter. This disassociates the parent and dependent monitoring environments.

---

## Services for Multiple Address Space Work Managers

The structure of the subsystem work manager dictates which workload management services can be used. A multiple address space work manager normally consists of three kinds of address spaces: router, execution, and supporting. The router address space receives incoming work requests, and passes them off to execution address spaces, which might use the services of supporting

address spaces. Figure 5 shows the sequence of functions in a multiple address space work manager.

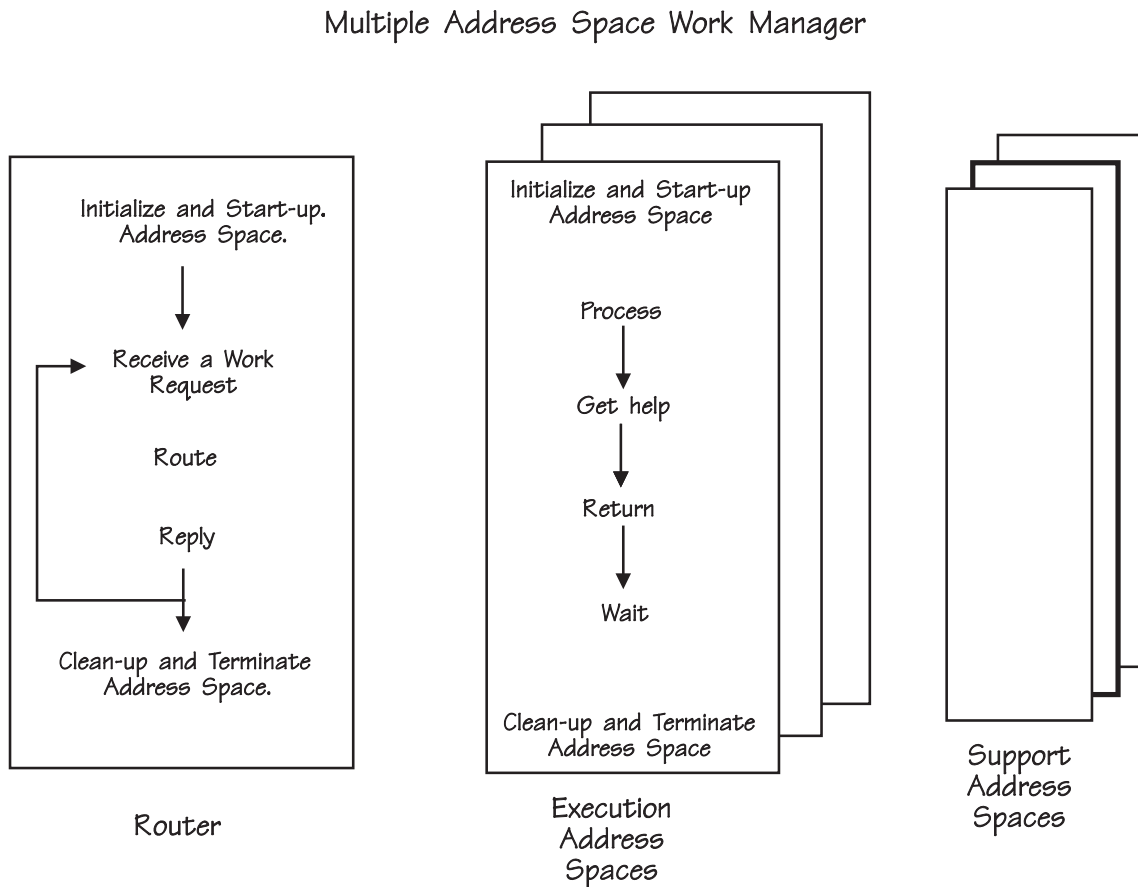


Figure 5. Sequence of function in a multiple address space work manager.

There are several groups of services that are useful to a multiple address space work manager:

- Chapter 3, “Creating and Using Enclaves,” on page 31  
The enclave services let you manage transactions across multiple address spaces in the same service class as the original request. The customer can assign a response time, discretionary, or a velocity goal to work, and can define period switching.
- Chapter 5, “Using the Queueing Manager Services,” on page 61  
The queueing manager services make it possible for the system to dynamically start and stop server address spaces based on the workload, and manage the work queues associated with the server address spaces to meet service class goals. The customer can assign a response time, discretionary, or a velocity goal to work, and can define period switching.
- “Execution Delay Monitoring Services for Multiple Address Space Work Managers” on page 25.

The execution delay monitoring services let you associate a service class with work, and the customer can assign a response time goal. You can also get response time and delay information about how well the work did to meet the goal. However, if you want to have the advantages of enclaves (such as having a single transaction that spans multiple address spaces and is managed to the goal of the originating address space), you should use enclave services rather than execution delay monitoring services.

## Execution Delay Monitoring Services for Multiple Address Space Work Managers

The structure of the multiple address space subsystem work manager determines which workload management services you can use to monitor work in multiple address spaces across a sysplex. The relate (IWMMRELA), transfer (IWMMXFER), and switch (IWMMSWCH) services provide a way for the work manager to indicate that a transaction is continuing execution somewhere else.

Some questions to help determine which services are appropriate include:

- Are the router, execution, and support address spaces all on one MVS image, or can they be distributed across MVS images?

How many MVS images are involved influences the choice of the relate/transfer pair or switch service. IWMMXFER requires both monitor environments to be on the same image, and knowledge of the parent monitoring environment.

IWMMSWCH says the continuation of this work is “somewhere else,” either waiting within the MVS image, or in the sysplex, or the network. In addition, it indicates that the work request is waiting for the continuation to return.

- What are the addressability requirements?

Connect (IWM4CON) and create (IWM4MCRE) identify the key in which future services are issued, and IWMMRELA requires addressability to the parent monitoring environment. IWMMXFER requires updating the dependent monitoring environment, and requires addressability and key update access to the parent monitoring environment.

- What are the dispatchable units in the servers and support address spaces?

You must specify the dispatchable unit type on the INIT and RELATE services.

- When are the “arrival time” and the work qualifiers (name, userid, etc.) known?

The arrival time for the work request is required for INIT, and the work qualifiers are required for CLASSIFY.

- What is the current communication between the router, servers, and the support address spaces, so that new data can be passed?

The participating subsystem work managers may want to pass the service class token returned from the IWMCLSFY service, together with the work request using their own communication methods.

Figure 6 on page 26 shows some suggested services for a multiple address space work manager that take into account the previously discussed considerations.

## Multiple Address Space Work Manager

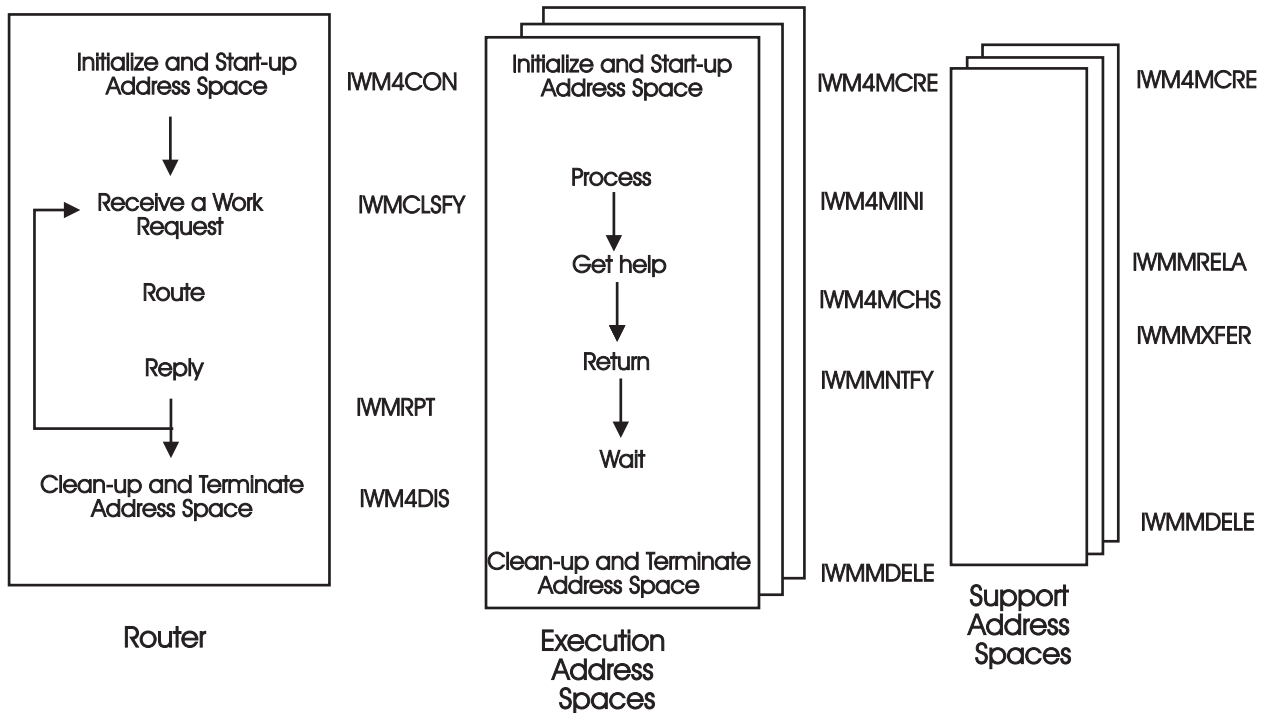


Figure 6. Example of services that monitor work across multiple address spaces.

In the figure, the router could be like a CICS TOR that routes the work off to an AOR, the execution address space, for processing. There may be some supporting address space involved that help process the work requests. The router would issue the IWM4CON macro to connect to workload management. It would also issue the IWMCLSFY service to associate the arriving work with a service class. Since it also receives the work request back once it has been processed, it also issues the IWMRPT service to report the completion, and the IWM4DIS at address space termination.

The execution address spaces would issue the IWM4MCRE to create the monitoring environments and record information about the work with the IWM4MCHS service. It would issue the IWM4MNTFY to signify that the execution phase of work request execution has completed. Then, at address space termination, it would issue the IWM4MDELE service to delete the monitoring environments.

The supporting address spaces would also use monitoring environments, so they would issue IWM4MCRE and IWM4MDELE. To show that the information they are keeping in the performance block reflects the same work request as the monitoring environment created by the execution address spaces, they issue the IWM4MRELA and IWM4MXFER services.

IWM4MRELA and IWM4MXFER require the supporting address space to be on the same MVS image, and that the monitoring token be passed from the execution address space to the supporting address space.

---

## Services for Work Managers that Distribute Work Requests

Some work managers distribute work across systems in a sysplex. Distributed work may originate from one subsystem work manager, and be processed by another. For example, a work manager may send work to a data base manager for processing. Other work managers may split up complex work into smaller pieces and distribute the pieces to other systems in the sysplex for processing.

Whether the work is distributed, or split and distributed, you may still want to keep the work classified according to the subsystem originating the work request. Not all classification information may be available to the receiving subsystem. For example, suppose JES has distributed a batch job to a data base system. The data base system issues IWMCLSFY when it receives the batch job. Because it is not the same subsystem environment, the batch job is now classified into a service class representing the data base work, and not to a service class representing batch jobs.

A work manager can use the IWMWMCON macro when it receives work and wants it to be classified using the originator's subsystem environment, and not its own. The IWMWMCON macro lets a caller modify the subsystem type and subsystem name previously provided on the IWM4CON macro.

Once the receiving subsystem has issued the IWMWMCON macro, the subsystem can issue the IWMCLSFY macro. The work is then classified according to the modified environment. Note that any other macro that requires subsystem name, subsystem type, or service class name is affected by the change. Those macros are:

- **IWMCLSFY**, which returns a service class based on the modified subsystem environment attributes. See Chapter 12, "IWMCLSFY – Classify Work," on page 135 for a complete description.
- **IWM4ECRE**, which creates an enclave based on the parameter list from IWMCLSFY. See Chapter 3, "Creating and Using Enclaves," on page 31.
- **IWMRPT**, which reports on the completion of the work associated with the service class received from IWMCLSFY. See Chapter 46, "IWMRPT – Report on Work Request Completion," on page 393 for a complete description.

For details on the IWMWMCON macro, see Chapter 60, "IWMWMCON – Modify Connect Information," on page 495.

## Determining the Subsystem Name and Type

A caller must provide a subsystem name and type on the IWMWMCON macro. To determine the subsystem name or type, a caller can use the REQASCL SYSEVENT. The REQASCL SYSEVENT provides information about an address space's classification information. The originating subsystem should issue REQASCL, and pass the information with the work request.

For information about how to use the REQASCL SYSEVENT, see *z/OS MVS Programming: Authorized Assembler Services Reference SET-WTO*.

## Using IWMWMCON When Distributing Work in a Sysplex

The expected use of the IWMWMCON service is in a multiple system work environment, where work is received by a work router and distributed to other systems in the sysplex. In this case, the enclave or address space transaction of the originator cannot be used to manage the work request, because a transaction has a single system scope. When a request or a part of a request is distributed to a

different system, it must run under a new transaction. The example below shows how IWMWMCON can be used in conjunction with the IWM4ECRE service to create an independent enclave on the system receiving the work request. For details on using enclaves, see Chapter 3, “Creating and Using Enclaves,” on page 31.

### **Example of Using IWMWMCON**

Suppose a subsystem work manager called DISS splits complex work requests into pieces and distributes the split work to six data base manager address spaces, called DB1 through DB6, each running on a separate system in the sysplex. DISS might communicate with the data base address spaces through a shared queue on DASD or a coupling facility, or through sysplex services. The DISS subsystem does the following:

1. Receives a work request.
2. Determines the work requestor's classification attributes using the SYSEVENT REQASCL macro.
3. Splits the request and distributes pieces to DB1 through DB6, passing the information returned by REQASCL.

Each data base subsystem, DB1 through DB6, does the following:

1. Receives the split work request along with its classification attributes.
2. Obtains a latch or lock prior to issuing IWMWMCON to serialize the use of the work manager connect environment.
3. Modifies the connect environment by issuing IWMWMCON with the subsystem name and type passed by DISS.
4. Builds a classification parameter list with the attributes passed by DISS using the modify form of the IWMCLSFY macro.
5. Creates an independent enclave using the IWM4ECRE macro to manage the split work request.
6. Restores the previous connect environment using the IWMWMCON macro with the previous subsystem type and name.
7. Releases the latch or lock being used to serialize the connect environment.
8. Processes the work by joining tasks to the enclave and/or scheduling SRBs into it.
9. When the work is finished, deletes the enclave using the IWM4EDEL macro.

### **Considerations for Mixed Releases**

The example above assumes that all systems are running OS/390 R3 or later. If your subsystem needs to also support earlier levels of OS/390, then the data base address spaces running on the down-level systems may need to operate differently than shown above.

First, you need to be careful when passing classification information between systems at different levels. The subsystem issuing the IWMCLSFY macro needs to use a parameter list compatible with the level where it is running.

Also, there is a restriction on pre-OS/390 R3 systems. You need to run enclave transactions in a service class or performance group separate from address space transactions. So, if the work request is running under the originator's address space transaction when it is received by DISS, the data base address space must ensure that the independent enclave it creates for the request is classified differently.

One way to force separate classification of the enclaves is to use a new subsystem type and name when invoking IWMWMCON from the data base address space. Extending the example above, assume DB7 is a data base address space running on a pre-OS/390 R3 system. When DB7 receives a request from DISS it does the following:

- Like steps 1 and 2 above for DB1 through DB6, receives a request and obtains a latch or lock.
- When calling IWMWMCON, passes in a new subsystem type, such as DBR, rather than the originator's subsystem type. DB7 preserves the originator's subsystem type, by putting it in the subsystem name field. The originator's subsystem name is not used. The subsystem type is preserved because it is a higher level attribute than subsystem name.
- Provides other classification attributes unchanged via the modify form of IWMCLSFY.
- Continues with steps 5 through 9 above.

This means that when DB7 creates an enclave, it is classified into a service class or performance group based on the classification rules defined for subsystem type DBR in the active service policy, rather than those defined for the originator's subsystem type.

A subsystem that makes use of a unique subsystem type to classify remote work as in this mixed release example, must explain this in the subsystem reference information so the installation knows how to properly define classification rules for the subsystem.





---

## Chapter 3. Creating and Using Enclaves

An **enclave** is a transaction that can span multiple dispatchable units (SRBs and tasks) in one or more address spaces and is reported on and managed as a unit. The enclave is managed separately from the address spaces it runs in. CPU and I/O resources associated with processing the transaction are managed by the transaction's performance goal, accounted to the transactions, and reported to the transaction. A program can create an enclave, schedule SRBs into it, or join tasks to it. A multisystem work manager can process a transaction on multiple systems by using a multisystem enclave.

Use the following services to work with enclaves:

- The **IWM4ECRE** macro allows you to create an enclave.
- The **IWMEREG** macro allows you to register an enclave to prevent it from premature deletion.
- The **IWMEDREG** macro allows you to deregister an enclave.
- The **IEAMSCHD** macro allows you to schedule an SRB into the enclave.  
For information about using the IEAMSCHD macro, see *z/OS MVS Programming: Authorized Assembler Services Guide*.
- The **SYSEVENT ENCASSOC** macro allows an enclave running SRBs to be associated with an address space so the server address space's storage-related resources can be managed to the enclave's performance goal.  
For information about using the SYSEVENT ENCASSOC macro, see *z/OS MVS Programming: Authorized Assembler Services Guide*.
- The **SYSEVENT ENCSTATE** macro allows the creator of an enclave to notify SRM when the enclave is idle, so that its state is sampled correctly.  
For information about using the SYSEVENT ENCSTATE macro, see *z/OS MVS Programming: Authorized Assembler Services Guide*.
- The **IWMEJOIN** macro allows a task to join an enclave.
- The **IWMELEAV** macro allows a task to leave an enclave.
- The **IWMEXPT** macro allows you to export an enclave to all systems in a parallel sysplex.
- The **IWMUEXPT** macro allows you to undo an export.
- The **IWMIMPT** macro allows you to import an enclave that has been exported.
- The **IWMUIMPT** macro allows you to undo an import.
- The **IWMECQRY** macro allows a program to query the classification information associated with an enclave.
- The **IWMESQRY** macro provides a program with information about whether the current dispatchable unit is associated with an enclave.
- The **IWM4EDEL** macro allows a program to delete a previously created enclave.

---

### Why Would You Use an Enclave?

Use an enclave when you have a transaction that spans multiple tasks or SRBs in one or more address spaces, and you want to manage it as a unit. An enclave allows you to manage and report on resource consumption in the enclave based on a performance goal unrelated to the performance goal(s) of the address space(s) in which the enclave's dispatchable units execute.

An independent enclave represents a complete transaction. Its performance goal is assigned based on the service class to which it is classified when the enclave is created. Each independent enclave starts in period 1 of its service class and switches periods based on the service consumed by the dispatchable units belonging to the enclave.

A dependent enclave represents the continuation of an existing address space transaction under a new set of dispatchable units. Its performance goal is inherited from the existing address space transaction based on the service class (or PGN) and period being used to manage the address space at the instant the dependent enclave is created. CPU service consumed by a dependent enclave is treated as if it were consumed by the address space transaction, and can cause the address space along with the dependent enclave to switch into later periods.

If your work manager does **not** use enclaves, work can only be managed on an address space basis, tied to the address space the work runs in. If you have a transaction that spans multiple address spaces, use an enclave to manage the transaction as a unit.

If you have an address space that executes multiple transactions, use enclaves to isolate the transactions so they can be reported on and managed individually.

## SRBs in Enclaves

Enclave SRBs offer advantages over local and global SRBs in that they are preemptable, and can be run at a lower major dispatching priority than tasks in the same address space.

SRBs in enclaves work well for transactions having short durations, not issuing supervisor calls, and not otherwise requiring a task environment. SRBs have very little startup overhead compared to tasks. The subsystem can create an enclave using the IWM4ECRE macro, and then schedule SRBs to run in the enclave using the IEAMSCHD macro.

The SYSEVENT ENCASSOC macro is used to indicate that an enclave and an address space are related for storage management purposes. The ENCASSOC sysevent is necessary only when SRBs are used. A task that joins an enclave automatically associates the home address space with the enclave. See *MVS Programming: Authorized Assembler Services Reference, Volume 4* for more information about the SYSEVENT macro.

For more information about SRBs and how to use them, see *z/OS MVS Programming: Authorized Assembler Services Guide*. For more information about the IEAMSCHD macro, see *z/OS MVS Programming: Authorized Assembler Services Reference EDT-IXG*.

## Tasks in Enclaves

Using tasks in enclaves offers all the advantages of enclaves and allows the enclave to perform functions that require a task environment, such as supervisor calls. Unlike SRBs, tasks can dynamically leave and join an enclave as they finish one piece of work and begin another.

A subsystem can create an enclave using the IWM4ECRE macro, join the task to the enclave using the IWMEJOIN macro, process the work request, and remove the task from the enclave using the IWMELEAV macro. If a task joins an enclave and

subsequently attaches subtasks, the subtasks are automatically joined to the enclave. The interactions between enclaves and attach/detach are summarized as follows:

- Subtasks attached while the mother task belongs to an enclave inherit membership in the same enclave.
- Subtasks that already exist when the mother task joins an enclave are **not** automatically made part of the enclave although they may explicitly join and leave the enclave using IWMEJOIN and IWMELEAV.
- Tasks which inherit membership in an enclave can only leave the enclave by terminating or by deletion of the entire enclave.
- Mother tasks with subtasks that inherited enclave membership cannot leave the enclave until all such subtasks terminate.

## Comparison of Enclaves and Execution Delay Services

You cannot use tasks in enclaves and execution delay services in the same address space. This same restriction applies to SRBs in enclaves if you use the SYSEVENT ENCASSOC to associate the enclave with an address space. When deciding which set of services to use for a work manager, you should consider the following advantages of enclaves over execution delay services:

- Isolation of transactions  
Enclaves allow separate dispatching priorities to be assigned to work running in the same address space. Therefore, workload management can manage this work to different performance goals. Without enclaves, all work in an address space runs at the same major dispatching priority.
- Period control  
Enclaves can run in a service class with multiple periods. Because resource consumption is tracked for individual enclaves, the enclave can move from one period to the next as it consumes CPU resource. The goals for the periods can be chosen to favor short transactions over long ones within a single address space.
- Full goal support  
Enclaves support response time, velocity, and discretionary goals, whereas transactions reported using execution delay services can be managed only to response time goals.
- Server address space management  
Enclaves are independent from an address space, so a transaction that moves from the originating address space to one or more server address spaces can be managed as a single transaction.
- Multisystem scope  
Enclaves can span address spaces on multiple systems in a parallel sysplex, whereas transactions supported using execution delay services are constrained to a single system.

---

## Creating an Enclave

A subsystem creates an enclave using the IWM4ECRE service. You can define **independent** or **dependent** enclaves.

## Independent Versus Dependent Enclaves

You use independent and dependent enclaves for different purposes, as follows:

- **Independent Enclaves**

Use an independent enclave to represent a new transaction. The TYPE=INDEPENDENT parameter on IWM4ECRE is the default. An independent enclave must be classified into a service class or performance group when it is created, so the caller must provide classification qualifiers as input to IWM4ECRE. The home address space when IWM4ECRE is issued is the **owner** of an independent enclave. CPU service consumed by the enclave is accumulated in the SMF 30 record of the owning address space and the SMF 72 record of the enclave's service class or performance group period.

For examples showing how to use independent enclaves, see "Scheduling an SRB in an Independent Enclave" on page 36 and "Joining Tasks to an Independent Enclave" on page 38.

- **Dependent Enclaves**

Use a dependent enclave when you have an existing address space defined with its own performance goal that you wish to extend to programs running under dispatchable units in other address spaces. The home address space at the time IWM4ECRE is invoked becomes the **owner** of the enclave. Specify TYPE=DEPENDENT on IWM4ECRE to indicate a dependent enclave. A dependent enclave derives its performance goal from the owning address space, and all CPU service consumed by the enclave is accumulated in the SMF 30 record of the owning address space and the SMF 72 record of the owning address space's service class or performance group period.

The TYPE=MONENV parameter creates a dependent enclave owned by the address space of a specified monitoring environment. Note that this dependent enclave is managed to the goal established for the owning address space, not the response time goal that might have been established for the monitoring environment.

For an example showing how to use dependent enclaves, see "Using Dependent Enclaves" on page 40.

For more specific differences between independent enclaves and dependent enclaves, see Table 12 on page 47.

## Registering an Enclave

Enclave transactions do not only exist within a subsystem, but also across subsystems. Enclaves can be deleted by any subsystem at any time. So, it might happen that a subsystem deletes an enclave that is still used by another subsystem. To avoid premature deletion, you can register an enclave. The registration indicates to the system that an enclave must not be deleted until the registering subsystem deregisters it.

The new service, IWMEREG, allows an enclave to be registered in order to prevent it from premature deletion until the enclave is deregistered. The new service, IWMDREG, allows the registration for an enclave to be undone and deleted

The registration is owned by the job step task of the home address space at the time IWMEREG is invoked. If the job step task or the address space terminates, the system implicitly deregisters the enclave.

Only subsystems which utilize enclaves created by other subsystems need to register interest in an enclave while using it. If a subsystem only uses enclaves that it created itself, there is no need to register interest in the enclave.

## Multisystem Enclaves

**Note:** The use of multisystem enclaves requires the definition of a coupling facility structure named SYSZWLW\_WORKUNIT in the CFRM policy. Once the CFRM policy with this structure definition is activated, then WLM will automatically connect to the structure, enabling the use of multisystem enclaves. See *z/OS MVS Planning: Workload Management* for more information.

Some work managers split large transactions across multiple systems in a parallel sysplex, improving the transaction's overall response time. These work managers can use multisystem enclaves to provide consistent management and reporting for these types of transactions.

Among the benefits of using multisystem enclaves:

- All parts of a split transaction are managed using the same service class. If the service class has multiple periods, the CPU usage of the entire transaction is used to switch periods.
- The enclave owner's SMF 30 record includes CPU time accumulated by all of the multisystem enclaves it owns, for *all* systems on which they executed. Remote system service is reported by individual system within the SMF 30 record.

A multisystem enclave begins as either an independent or dependent enclave on a single system. This enclave is called the "original" enclave. If the work manager decides to involve other systems in the processing of the work unit, it issues IWMEXPT to "export" the enclave to other systems in the parallel sysplex. The export token it receives back from IWMEXPT is a sysplex-wide unique name that it must now pass along with the work request to other systems.

Each work manager in the supporting address spaces on other systems can now issue IWMIMPT to "import" the enclave onto its system. It passes the export token and receives a special enclave token that is valid for its system only. This new, supporting enclave is called a "foreign" enclave. The original enclave and the foreign enclaves are all referred to as one unit called a multisystem enclave.

When work has completed in a foreign enclave, the supporting work manager issues IWMUIMPT to "unimport" the enclave, and then signals its completion to the original work manager. When all of the supporting work managers have unimported their enclave, the original work manager issues IWMUEXPT to "unexport" the original enclave. When all work is finished, the original work manager that created the original enclave deletes it.

If your subsystem uses an enclave that it did not create for its processing, then you should use the registration services (IWMEREG, IW MEDREG) to protect the enclave against deletion by its owner while your subsystem is using it. The IWMUIMPT service delays the physical deletion of an enclave as long as the enclave is registered by any subsystem.

Each work manager must first connect to WLM using the IWM4CON service, specifying EXPTIMPT=YES to enable exporting and importing. IWMEXPT, IWMIMPT, IWMUIMPT, and IWMUEXPT must all be invoked from the address space that connected.

WLM will automatically undo a work manager's export and import requests when:

- The work manager disconnects from WLM

- The work manager's connecting task or address space ends
- The work manager's system fails.

If an export is undone, whether by the original work manager's request or due to WLM's recovery action, before all of the supporting work managers have completed their work in the foreign enclaves, the outstanding imports are handled as follows:

- An outstanding import on the same system as the original enclave is automatically undone. (When a work manager on the same system as the original work manager attempts to import the original enclave, it receives the original enclave token. It's not really "exported" at all.) The only effect will be a warning return code when the work manager attempts to unimport the enclave.
- An outstanding import on a foreign system will remain in effect. WLM provides no notification to the supporting work manager that the export has ended. The supporting work manager must learn of the failure through its own mechanisms and then terminate the work on its own.
- New import requests are rejected. The supporting work manager should terminate any work being done on behalf of the original work manager.

As a transaction flows from one work manager to another, it is possible that more than one work manager will split its processing across multiple systems. In this way, an original enclave can be exported multiple times, both by the original work manager and by other work managers. Each export request is tracked separately, and requires a corresponding unexport request. Multiple concurrent exports all share the same export token.

If a work manager on the original system attempts to import the original enclave, it will receive the enclave token of the original enclave. The work manager can schedule SRBs into or join tasks to the original enclave just as it would any other enclave on the same system.

Just as an enclave can be exported multiple times, it can also be imported multiple times by one or more supporting work managers. Each import request is tracked separately, and requires a corresponding unimport request. Multiple concurrent imports on a single system all share a single foreign enclave.

A foreign enclave cannot be exported — in other words, once an enclave has been imported onto a foreign system, it cannot be exported again *from that system*. If a work manager invokes IWMEXPT for a foreign enclave, it will receive a warning code along with the existing export token for that enclave.

For an example showing how to use multisystem enclaves, see "Using a Multisystem Enclave" on page 42.

## Scheduling an SRB in an Independent Enclave

Suppose an address space representing a subsystem uses the specialized processing services of a supporting address space to satisfy a work request. The subsystem creates an independent enclave that is used by an SRB executing in the supporting address space on behalf of the work request. Part of the work request executes under an SRB in the subsystem address space, so that SRB uses the same enclave.

Figure 7 on page 37 shows the two address spaces.

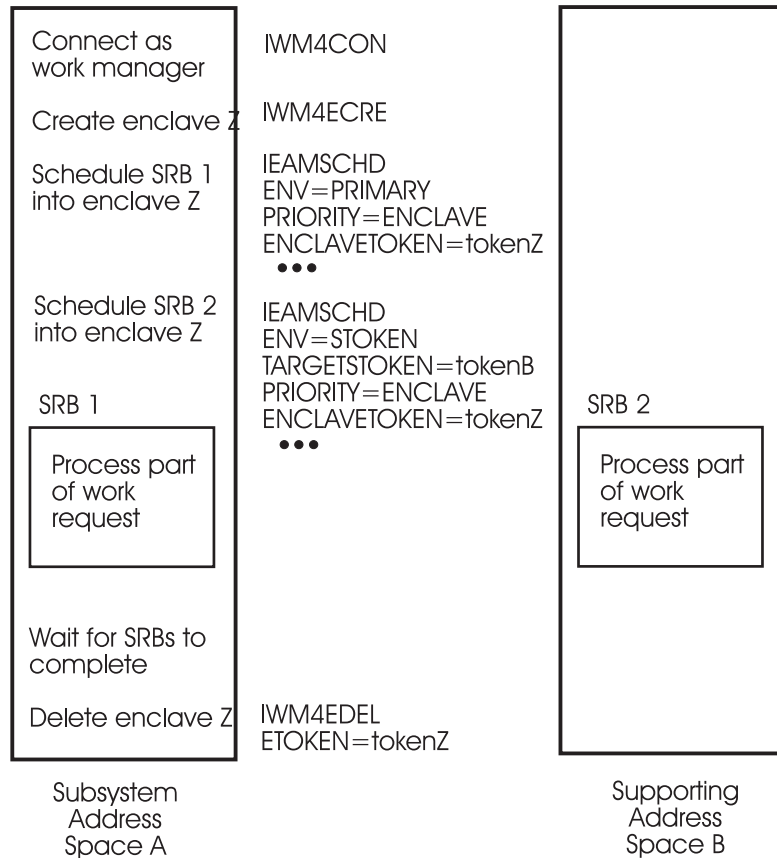


Figure 7. Creating an Independent Enclave and Scheduling an SRB

In the figure,

- **Connect as a work manager**

Subsystem address space A issues IWM4CON to connect to workload management with WORK\_MANAGER=YES specified or defaulted. This makes work management services, including enclave services, available to the connecting address space.

- **Create enclave Z**

Subsystem address space A wants to manage multiple SRBs (SRBs 1 and 2) as a unit, so address space A creates an independent enclave Z by issuing IWM4ECRE. Subsystem address space A is the home address space, and is the owner of enclave Z. Any work that the subsystem and its supporting address space B process can be managed as an enclave. Classification information is passed in with IWM4ECRE so workload management can assign the enclave to a service class or performance group and manage to those goals.

**Note:** If enclave SRB 2 issues SYSEVENT ENCASSOC, no work consuming significant CPU resource should run in the supporting address space outside of the enclave. Similarly, if enclave SRB1 issues SYSEVENT ENCASSOC, no work consuming significant CPU resource should run in the subsystem address space outside of the enclave. For more information, see “Performance Management of Address Spaces with Enclaves” on page 45.

- **Schedule SRB 1 into enclave Z**

The subsystem address space then schedules an SRB, SRB 1, to execute in its own address space and to be managed as part of enclave Z, using the IEAMSCHD macro:

```
IEAMSCHD ENV=PRIMARY,  
          EPADDR=entry_point_address,  
          PRIORITY=ENCLAVE,  
          ENCLAVETOKEN=tokenZ
```

Where the subsystem address space has defined:

```
tokenZ          FL4          The enclave token for enclave Z
```

- **Schedule SRB 2 into enclave Z**

The subsystem address space then schedules an SRB (SRB 2) to execute in supporting address space B and to be managed as part of enclave Z, using the IEAMSCHD macro:

```
IEAMSCHD ENV=STOKEN,  
          TARGETSTOKEN=tokenB,  
          EPADDR=entry_point_address,  
          PRIORITY=ENCLAVE,  
          ENCLAVETOKEN=tokenZ
```

- **Wait for the SRBs to complete and delete enclave Z**

The subsystem waits for the SRBs to complete the request, then deletes the enclave using the IWM4EDEL macro and returns to the caller.

## Joining Tasks to an Independent Enclave

Again, an address space representing a subsystem is using a supporting address space for part of the processing for a unit of work. The subsystem address space creates an independent enclave, and a task in the supporting address space joins the enclave. A task in subsystem address space can also join the enclave when it is processing on behalf of the unit of work.

Figure 8 on page 39 shows how this works.



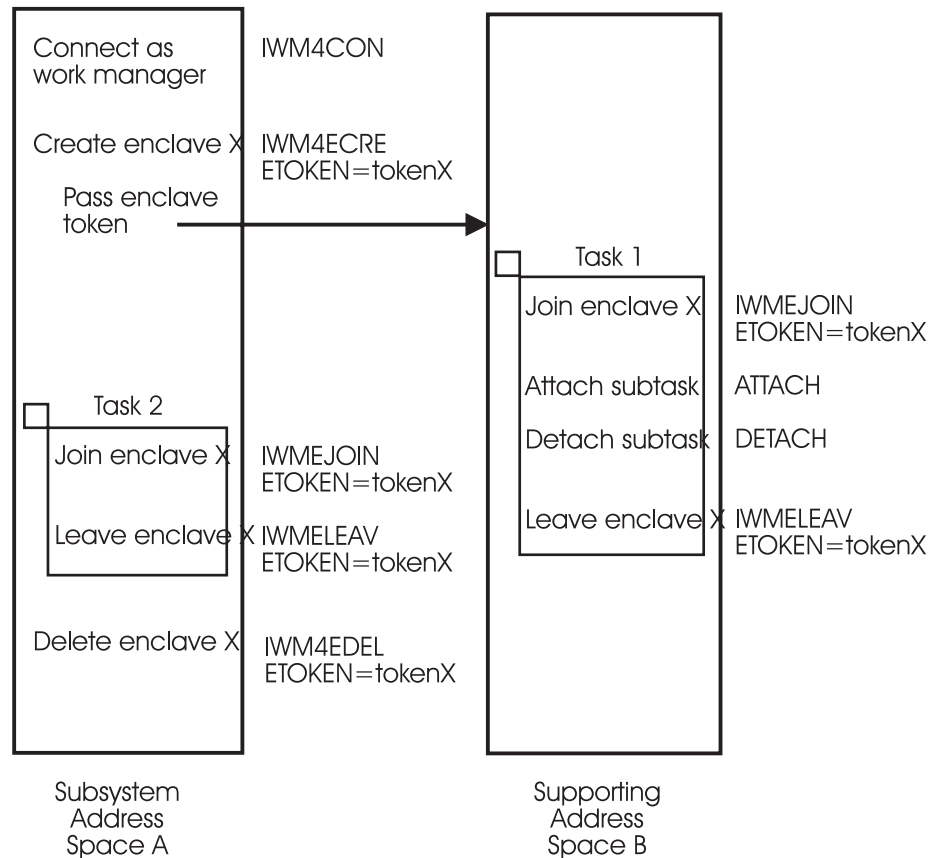


Figure 8. Creating an Enclave and Joining Tasks To It

In the figure,

- **Connect as work manager**

Subsystem address space A issues IWM4CON to connect to workload management with WORK\_MANAGER=YES specified or defaulted. This makes work management services, including enclave services, available to the connecting address space.

- **Create enclave X**

Subsystem address space A wants to manage work in multiple address spaces as a unit, so address space A creates independent enclave X by issuing IWM4ECRE. Subsystem address space A is the home address space, and is the owner of the enclave. IWM4ECRE passes back the enclave token *tokenX* to the subsystem. Any work that the subsystem and its supporting address space B process can be managed together as an enclave. Classification information is passed in with IWM4ECRE so workload management can assign the enclave to a service class or performance group and manage to those goals.

**Note:** No work consuming significant CPU service should run in either the subsystem or the supporting address space outside of an enclave. For more information, see “Performance Management of Address Spaces with Enclaves” on page 45.

- **Task 1: join enclave X**

The subsystem passes the work to its supporting address space B along with the enclave token *tokenX* from IWM4ECRE. Before Task 1 in address space B runs

the work passed in by the subsystem, it joins the enclave by issuing the IWMEJOIN service with enclave token *tokenX*. Now the work running under Task 1 is managed to the goal of the enclave.

Although this example shows only one enclave, the subsystem can create an enclave for each new unit of work that arrives. These enclaves can be running work simultaneously in the subsystem and the supporting address space, with each unit of work being managed to its own unique goal.

- **Task 1: attach subtask; detach subtask**

In address space B, Task 1, which now belongs to enclave X, issues the ATTACH macro to create a subtask. This subtask will also, automatically, be part of enclave X and be managed to the enclave's goal. When the subtask is detached, it automatically leaves the enclave. The subtask cannot use IWMELEAV to do this. Each subtask attached by Task 1 after it joins the enclave must be detached before Task 1 leaves the enclave.

- **Task 1: leave enclave X**

Task 1 finishes its processing and leaves the enclave by issuing the IWMELEAV service with the enclave token *tokenX* specified. Any processing in Task 1 after it leaves the enclave is managed to the goal of the address space, not of the enclave.

- **Task 2: join enclave X**

If the subsystem itself has work to do on behalf of the unit of work, it can join a task, Task 2, to the same enclave as the supporting address space used. It uses the same enclave token, *tokenX*, on IWMEJOIN. The work in Task 2 is now managed to the enclave's goal. Task 1 and 2 can be run concurrently or in sequence. At any point in time, an enclave can have multiple tasks and/or SRBs running in it across multiple address spaces, and they are all managed to the same enclave goal.

- **Task 2: leave enclave X**

Task 2 finishes its processing for the transaction and leaves the enclave. The task is now managed to the address space goal.

- **Delete enclave X**

The transaction is now complete, so the subsystem deletes the enclave using IWM4EDEL with enclave token *tokenX*. Note that the address space that deletes the enclave need not be the same one that created it.

## Using Dependent Enclaves

When a unit of work is processed in multiple address spaces, you can use dependent enclaves to tie the work done in a supporting (server) address space back to the originating client address space. The dependent enclave represents the continuation of an existing address space transaction under a new set of dispatchable units in another address space.

Figure 9 on page 41 shows how this works.

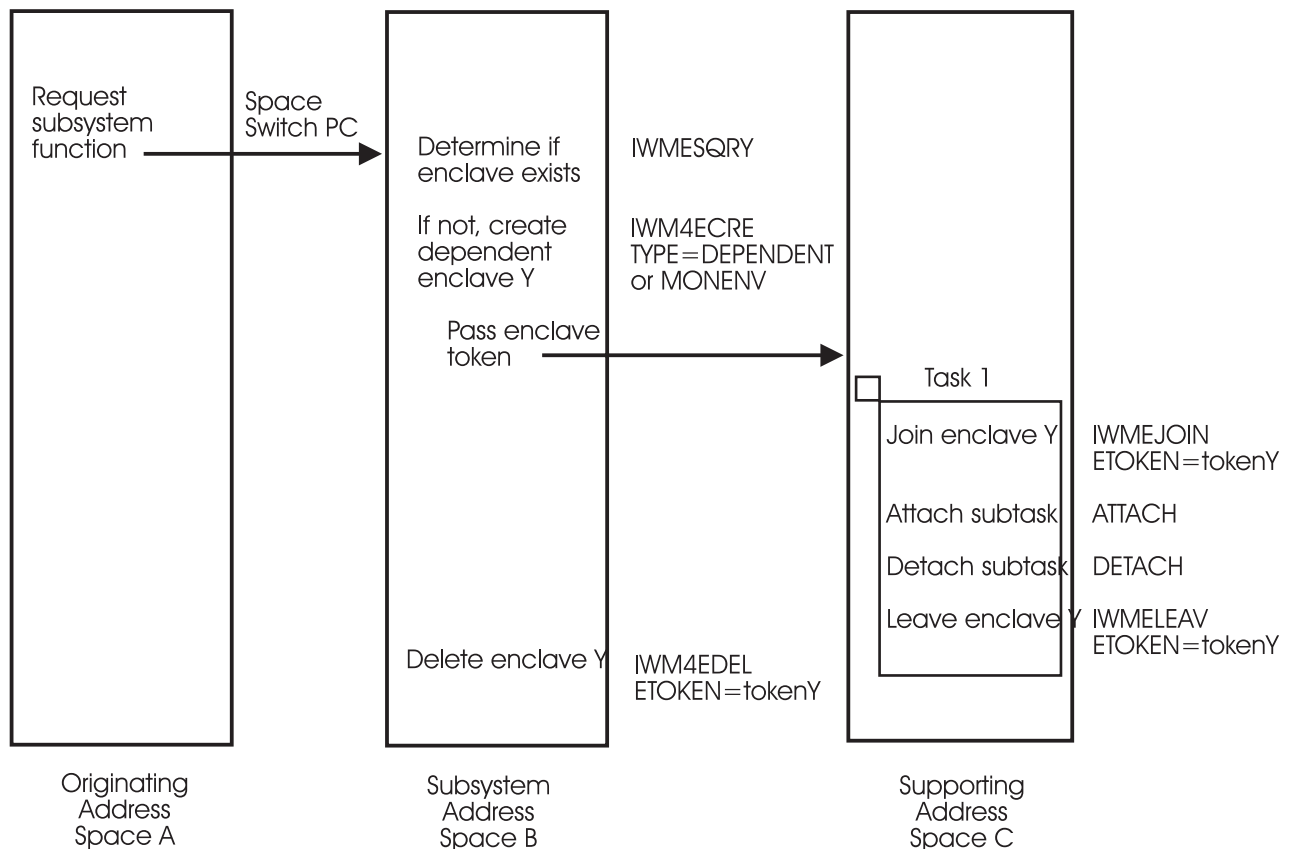


Figure 9. Using Dependent Enclaves

In the figure,

- **Request subsystem function**

The originating address space, address space A, sends a work request to subsystem address space B, for example, by issuing a space-switching PC.<sup>1</sup> Address space A could be a TSO, batch job, or started task. The address space's performance goal is used to manage the transaction while running in address space A and also when running in the dependent enclave.

- **Determine if an enclave exists**

Subsystem address space B uses IWMESQRY to determine whether the caller is already in an enclave. If it is in an enclave, it would use that enclave and schedule SRBs to the enclave, or join tasks to the enclave.

- **Create dependent enclave Y**

If the caller is not in an enclave, the subsystem creates a dependent enclave using IWM4ECRE with the TYPE=DEPENDENT parameter. The home space when IWM4ECRE is issued, in this example address space A, is the owner of the dependent enclave. No classification information is required on IWM4ECRE for a dependent enclave. The service class or performance group of the owning address space A is used to manage the work in the enclave. The subsystem does not need to connect to workload management (using IWM4CON) to create a dependent enclave.

1. A space-switching PC isn't required — it is used here only as an example. A nonspace-switching PC or other linkage can be used so long as the originating space remains the home space.

- **Pass enclave token**

The subsystem posts Task 1 in the supporting address space, address space C, to join the enclave, passing it the enclave token *tokenY* passed back by IWM4ECRE. The subsystem can also schedule SRBs into the same enclave. The work running in the dependent enclave executes in address space C but is managed to the goal of the owning address space A.

- **Join enclave Y**

Task 1 joins enclave Y using the enclave token passed from the subsystem. The work running under Task 1 is now managed to the goal of address space A.

**Note:** No work consuming significant CPU service should run in the supporting address space outside of an enclave. The same restriction applies to the subsystem address space if it runs work in enclaves. For more information, see “Performance Management of Address Spaces with Enclaves” on page 45.

- **Attach subtask; detach subtask**

Task 1 may attach one or more subtasks while it is joined to enclave Y. These subtasks are automatically joined to enclave Y and also managed to address space A's goal.

- **Leave enclave Y**

When Task 1 completes the work request, it leaves the enclave. It reverts back to being managed to address space C's goal.

- **Delete the enclave**

The subsystem waits for the tasks, and any SRBs, to complete the request, then deletes the enclave (if it created it) and returns to the caller.

## Using a Multisystem Enclave

In this case, a work manager will process a work request using one or more supporting address spaces on different systems in a parallel sysplex.

Figure 10 on page 43 shows how this works.

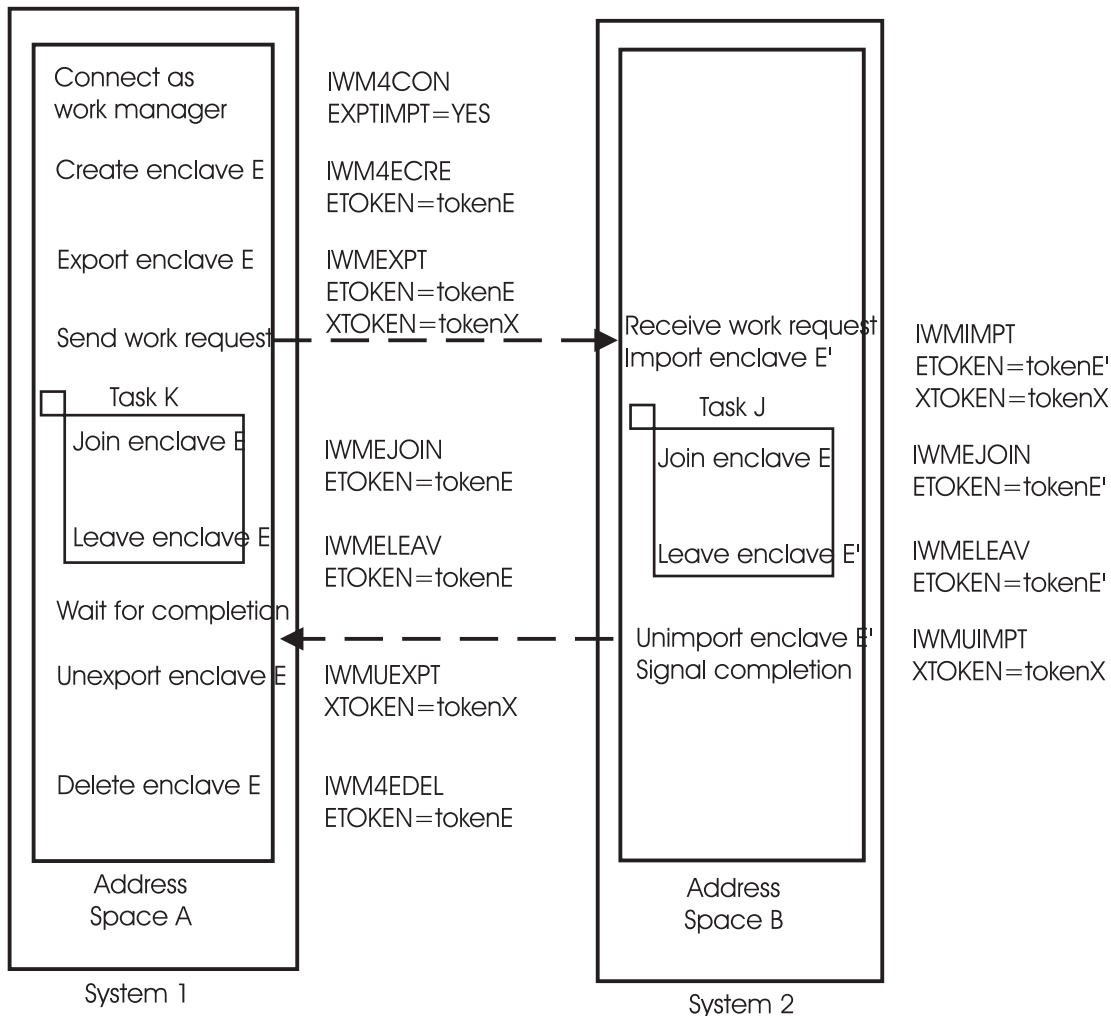


Figure 10. Using a Multisystem Enclave

In the figure,

- **Connect as work manager**

The work manager issues IWM4CON, with WORK\_MANAGER=YES specified or defaulted so that it can create independent enclaves, and EXPTIMPT=YES specified to allow for exporting the enclaves.

**Note:** It is assumed here that the CFRM policy already contains the coupling facility structure SYSZWLM\_WORKUNIT, which is required for the use of multisystem enclaves. If the coupling facility structure is not available, IWM4CON will succeed, but export and import requests will return errors. See *z/OS MVS Planning: Workload Management* for more information.

- **Create original enclave E**

The work manager creates an independent enclave in address space A on system 1 by issuing IWM4ECRE. Address space A is the owner of the original enclave. Classification information is passed in with IWM4ECRE so workload management can assign the enclave to a service class or performance group. IWM4ECRE passes back the enclave token *tokenE* to the work manager.

- **Address Space A on System 1: Export enclave E**  
The work manager exports enclave E to all other systems in the parallel sysplex by issuing IWMEXPT with the enclave token *tokenE*. IWMEXPT passes back the export token *tokenX*. The work manager can now pass this sysplex-wide unique export token to supporting address spaces on other systems, using its own communication mechanism.
- **Address Space B on System 2: Import enclave E'**  
Once an enclave has been exported, a work manager in the supporting address space B can import the enclave by issuing IWMIMPT with the export token *tokenX*. A foreign enclave E' is created. It receives back an enclave token *tokenE'* that is valid on system 2 only.  
  
Although this example shows only one supporting address space on one separate system, the enclave can be imported by several address spaces on several different systems in the parallel sysplex. These foreign enclaves can all be running work simultaneously, with each unit of work being managed to the goals of the original enclave.
- **Task J: join foreign enclave E'**  
Task J in address space B on system 2 joins the foreign enclave by issuing IWMEJOIN with the enclave token *tokenE'*.
- **Task J: leave foreign enclave E'**  
Task J in address space B on system 2 leaves the foreign enclave by issuing IWMELEAV with the enclave token *tokenE'*. Any further processing in Task J after it leaves the enclave is now managed to the goal of the address space B, not of the enclave.
- **Task K: join original enclave E**  
If work is to be done in the original enclave at the same time that work is being done in the foreign enclaves, task J in address space A on system 1 can join the original enclave by issuing IWMEJOIN with the enclave token *tokenE*. The work in Task K is now managed to the goals of the original enclave. At any point in time, a multisystem enclave can have multiple tasks and/or SRBs running in it across multiple address spaces on multiple systems, and they are all managed to the original enclave's goal.
- **Task K: leave original enclave E**  
Task K in address space A on system 1 leaves the original enclave by issuing IWMELEAV with the enclave token *tokenE*. Any further processing in Task K after it leaves the enclave is now managed to the goal of the address space A, not of the enclave.
- **Address Space B on System 2: Unimport enclave E'**  
Once the task has left the foreign enclave, the work manager in the supporting address space B unimports the enclave by issuing IWMUIMPT with the export token *tokenX*. This deletes the foreign enclave E'. The supporting work manager now reports its completion and any results to the original work manager using its own communication mechanism.
- **Address Space A on System 1: Unexport enclave E'**  
After every supporting work manager has reported its completion, the original work manager unexports the enclave by issuing IWMUEXPT with the export token *tokenX*.
- **Delete enclave X**  
The transaction is now complete, so the work manager deletes the enclave using IWM4EDEL with enclave token *tokenE*.

## Performance Management of Address Spaces with Enclaves

How workload management manages an address space where an enclave (either dependent or independent) is running depends on the characteristics of the enclave. Some address spaces with work running in enclaves are managed to the performance goal of the enclave. Other address spaces with work running in enclaves are managed to the performance goal of the address space transaction. In either case, workload management manages the dispatching priority of the enclave to meet the performance goal of the enclave.

An address space is managed (in terms of MPL level, paging, dispatching and I/O priorities) to the goal of the enclave if one of the following are true:

- The address space is running a task that has joined an enclave using the IWMEJOIN or IWM4STBG macro,

- OR -

- The address space is running an SRB that has issued the SYSEVENT ENCASSOC to associate the enclave with an address space.

**Note:** An address space must be non-swappable if it has enclave SRBs dispatched and SYSEVENT ENCASSOC has **not** been issued.

For an address space that is being managed to an enclave's goal, be aware of the following:

- It is assumed that no work consuming significant CPU service is running in the address space outside of an enclave. The CPU consumption of work running outside of enclaves is not included when workload management assesses the impact of CPU adjustments for the enclave work.
- A page fault incurred by a dispatchable unit (task or SRB) while running in the enclave is accounted for in the enclave's service class period as server private area page delay rather than as a cross-memory page delay.

## Using ENQ/DEQ or Latch Manager Services with Enclaves

There are some considerations to be aware of when using enclaves for tasks or SRBs that serialize on resources using the ENQ macro or the latch manager callable services. A task cannot change its transaction status, that is, cannot join or leave an enclave, while holding a resource using ENQ or the latch manager; an SRB cannot issue SYSEVENT ENCASSOC while holding a resource using the latch manager. Otherwise, enqueue promotion processing may not work properly. The recommended sequence is:

- Task: Join an enclave using IWMEJOIN or IWM4STBG. SRB: Associate enclave with an address space using SYSEVENT ENCASSOC.
- Obtain resource with ENQ or latch manager.
- Release resource.
- Task: Leave an enclave using IWMELEAV or IWM4STEN. SRB: Disassociate enclave from the address space using SYSEVENT ENCASSOC.

In addition, to ensure correct enqueue promotion processing, a task executing in an enclave should **not** make the following types of ENQ requests:

- Directed enqueues, that is, issuing the ENQ macro with the TCB= parameter
- Matching task enqueues, that is, issuing the ENQ macro with the MASID= and MTCB= parameters

---

## Enclave Resource Accounting

The accounting for resources consumed by an enclave depends on whether it is an independent, dependent, or a foreign enclave.

A dependent enclave is a logical continuation of the transaction already active in a client's address space. Therefore, CPU and MSO service for a dependent enclave is included in the SMF 30 record of the owning address space, and in the SMF 72 record for the address space's transaction. MSO service for the enclave is calculated based on the frame count of the owning address space, not on frame usage in the address space(s) where the enclave is executing.

For an independent enclave, CPU service is included in the SMF 30 record of the owning address space, and in the SMF 72 record for the enclave's service class or performance group period. MSO service is not calculated for an independent enclave.

For both dependent and independent enclaves, IOC service is included in the SMF 30 and 72 records associated with the address space where the enclave work is executing. SRB service for enclaves is always zero.

For a foreign enclave, CPU time is included in the SMF 30 record of the owning address space on the originating system. It is reported separately from local CPU time. CPU service is also included in the SMF 72 record on each system where the enclave executed.

Since CPU time used by foreign enclaves is included in the owner's SMF 30 record, it is *not* included in the SMF 30 records on the other systems where it actually executed. In order for those other systems to have some record of the CPU time used by foreign enclaves, an SMF 97 record is written for each SMF global recording interval. This SMF 97 record identifies the CPU time used by foreign enclaves during that interval, broken down by originating system. The installation can review the originating system's SMF 30 records to identify the specific jobs that consumed the CPU time in the foreign enclaves. Note that because data is collected asynchronously for the SMF 30 records, and because SMF intervals can vary from system to system, it may not be possible to exactly match SMF 30 times with SMF 97 times from one global interval to another.

Table 12 on page 47 compares control characteristics and resource accounting for independent, dependent and foreign enclaves.



Table 12. Enclave Characteristics and Resource Accounting. (a.s. = address space)

	Independent enclave	Dependent enclave	Foreign enclave
Dispatchable unit type	SRBs and/or tasks	SRBs and/or tasks	SRBs and/or tasks
New transaction?	yes	no	no
Owner	home space at the time IWM4ECRE is issued	home space at the time IWM4ECRE is issued (for TYPE=MONENV, owner is the a.s. associated with the monitoring environment — see note 1)	owner of the original enclave
Server	a.s. where enclave work is dispatched	a.s. where enclave work is dispatched	a.s. where enclave work is dispatched
Service class/report class	Assigned based on attributes passed to IWM4ECRE (see note 2)	Same as owner.	Same service & report class as original enclave.
CPU time	owner's SMF30Cpt (total) owner's SMF30Enc (ind. enclave only)	owner's SMF30Cpt (total) owner's SMF30Det (dep. enclave only)	owner's SMF30MRI (for foreign ind. enclave) owner's SMF30MRD (for foreign dep. enclave)
CPU service by a.s.	owner's SMF30Csu (total) owner's SMF30Esu (ind. enclave only)	owner's SMF30Csu (total)	CPU time/SMF30MRA/256 * CPU coefficient (CPU coefficient can be obtained from SMF 72 record)
CPU service by period	enclave's R723Ccpu or SMF72Cts	owner's R723Ccpu or SMF72Cts	enclave's R723Ccpu or SMF72Cts
DASD I/O connect time by a.s. (see note 3)	owner's SMF30Eic (ind. enclave only)	owner's SMF30Aic (dep. enclave + a.s.)	n/a
DASD I/O connect time by period (see note 3)	enclave's R723Cict or SMF72lct	owner's R723Cict or SMF72lct	enclave's R723Cict or SMF72lct
DASD I/O counts by a.s.	owner's SMF30Eis (ind. enclave only)	owner's SMF30Ais (dep. enclave + a.s.)	n/a
DASD I/O counts by period	enclave's R723Circ or SMF72lrc	owner's R723Circ or SMF72lrc	enclave's R723Circ or SMF72lrc
Page delay samples, with storage mgt. (see note 4)	enclave's R723Cspv	owner's R723Cspv	enclave's R723Cspv
Page delay samples, without storage mgt. (see note 4)	enclave's R723Caxm	owner's R723Caxm	enclave's R723Caxm
IOC service	server's SMF 30 and 72 records	server's SMF 30 and 72 records	server's SMF 30 and 72 records
SRB service	n/a	n/a	n/a
MSO service	n/a	owner's SMF30Mso, based on owner's frame count	n/a

**Notes:**

1. The address space associated with the monitoring environment is one of the following:
  - The address space related to the monitoring environment via the IWMMRELA service

- If there is no related space, the home space at the time IWM4MINI was issued
2. The attributes passed to IWM4ECRE are used with the classification rules in the active service policy to assign a service class and/or report class to the enclave.
  3. Connect time is used as an example here. Other measures associated with I/O in the SMF records are:
    - DASD I/O disconnect time in fields SMF30EID, SMF30AID, R723CIDT, and SMF72IDT.
    - DASD I/O wait time in fields SMF30EIW, SMF30AIW, R723CIWT, and SMF72IWT.
  4. Storage management is in effect for an enclave if either of the following is true:
    - The enclave includes one or more tasks.
    - The enclave includes at least one SRB which has issued the SYSEVENT ENCASSOC to associate itself with an address space.

---

## Managing the Performance of Work in Enclaves

This section describes how to classify the work running in enclaves. It is broken into two topics:

- Using Independent Enclaves
- Using Dependent Enclaves

### Using Independent Enclaves

You define a service class and a goal for work that is processed by the subsystem using the independent enclaves. Workload management then dynamically manages resource controls based on the goal.

For more information about defining performance characteristics for enclaves, see *z/OS MVS Planning: Workload Management*.

#### Example

Suppose your installation has a subsystem called DDF that uses enclaves for its distributed work requests. Your installation is running in goal mode with an active policy. To define the performance characteristics for the work scheduled to an enclave, you do the following:

- Define a workload and a service class for DDF work using the WLM ISPF application:

```
Service Class: DDF_ALL
Goal:          5 second response time
Importance:    3
```

- Using the WLM ISPF application, define a classification rule for the DDF subsystem type where all work goes into the DDF\_ALL service class.

```
Subsystem Type. . . . .: DDF

                        -----Class-----
                        Service      Report
DEFAULTS: DDF_ALL_____
```

- Install the service definition.
- Activate the service policy.

## Using Dependent Enclaves

Dependent enclaves are managed to the performance goal of the owning address space, so there is no need to separately classify dependent enclaves, or to define separate service classes or performance groups for them.

---

## Querying an Enclave's Classification Information

A caller can use the IWMECQRY macro to determine the classification information about an enclave. The classification information is that information passed on the IWMCLSFY macro for an independent enclave or inherited from the owning address space for a dependent enclave. For details on IWMCLSFY, see Chapter 12, "IWMCLSFY – Classify Work," on page 135.

### Example

To determine the classification attributes associated with an enclave represented by ETOKEN, first issue IWMECQRY to determine the length of the storage required to contain the classification information. The length of the area is dependent on the MVS release. Specify:

```
IWMECQRY ETOKEN=etoken,  
         ANSAREA=ansarea,  
         ANSLEN=anslen,  
         QUERYLEN=querylen
```

where the calling program has defined the following, and *ansarea* and *anslen* are set to zero:

etoken	DS	FL4	enclave token
ansarea	DS	A	Area to contain address of classification
anslen	DS	A	Length of the answer area
querylen	DS	A	Length of storage required

Obtain the amount of storage passed back in *querylen* and set *anslen* equal to *querylen*. Set *ansarea* to point to the storage and issue IWMECQRY again for the enclave classification information:

```
IWMECQRY ETOKEN=etoken,  
         ANSAREA=ansarea,  
         ANSLEN=anslen,  
         QUERYLEN=querylen
```

---

## Querying a Dispatchable Unit's Enclave Status

A caller can use the IWMESSQRY macro to determine whether the current dispatchable unit is associated with an enclave. If the dispatchable unit is associated with an enclave, the service returns the enclave token.

---

## Deleting an Enclave

A caller can delete an enclave using the 4EDEL macro. If the enclave is registered, it is only logically deleted. That is, it remains available until it is no longer registered by any subsystem.

When the enclave is deleted, the following occurs for each remaining dispatchable unit:

- **SRBs:**  
Each SRB belonging to the enclave is changed to a preemptable SRB and run at the dispatching priority of the current home address space (the address space into which the SRB was scheduled). The subsystem can purge SRBs using the PURGEDQ macro when the enclave still exists in the system. In most cases, this prevents the SRB from existing beyond the life of the enclave. For information on how to use PURGEDQ, see *z/OS MVS Programming: Authorized Assembler Services Guide*.
- **Tasks:** If an enclave ends with tasks still joined to the enclave, the tasks revert back to ordinary non-enclave tasks.

A foreign enclave is deleted using the IWMUIMPT macro.

For information about the IWM4EDEL macro, see Chapter 68, “IWM4EDEL – Delete an Enclave,” on page 573.

### Example

To delete an enclave, specify:

```
IWM4EDEL ETOKEN=etoken,
         RETCODE=retcode,
         RSNCODE=rsncode
```

Where the calling program has defined the following:

etoken	DS	FL4	Enclave token
retcode	DS	CL4	Return code
rsncode	DS	CL4	Reason code

---

## Chapter 4. Participating in Enterprise Workload Management

Enterprise Workload Manager (EWLM) is a systems service of the IBM Virtualization Engine™. Virtualization Engine is a set of technologies and systems services that allow system administrators to access and manage resources across multiple platforms (z/OS, AIX, i5/OS, Windows, Solaris and Linux). The EWLM systems service allows you to define business-oriented performance goals for an entire domain of servers across multiple platforms, and then provides an end-to-end view of actual performance relative to those goals.

For example, a multi-platform environment might include:

- A Web server tier running Windows
- An application server tier running AIX
- A database tier running z/OS

To ensure that work requests are performing as expected in this environment, you must be able to track the performance of those requests across server and subsystem boundaries. EWLM allows you to do that.

EWLM uses middleware that has been instrumented with the Open Group's Application Response Measurement (ARM) 4.0 standard. ARM 4.0 provides a set of interfaces that an application calls; these are then used by EWLM to calculate the response time and status of work processed by the application.

In a multi-tiered environment, what is perceived as one transaction may be implemented as many sub-transactions across several different applications. To relate the sub-transactions with one another, ARM 4.0 uses *correlators*. Relationships are established by passing the correlator of a parent sub-transaction to its child sub-transactions. The correlator for a parent sub-transaction, that is, a correlator that is passed from the sending application tier, is called a parent correlator. A correlator for its child sub-transaction is called a current correlator.

Ordinarily, applications that participate in EWLM use standard instrumentation based on the ARM 4.0 Java, C or C++ language bindings. However, because applications that use the WLM enclave services are already instrumented, WLM provides support for EWLM through a set of new and enhanced enclave services.

The WLM support for EWLM is effective only if the ARM agent is enabled and an EWLM policy is installed. Operators can control the state of the ARM agent by disabling or enabling the function from an operator console at any time with the F WLM,AM command. Additionally, the current state of the ARM agent can be displayed by the operator command D WLM,AM. For more information, see *z/OS MVS System Commands*.

For more information on Virtualization Engine and EWLM, including using the ARM APIs to instrument an application, see the @server Information Center on the Internet at [http://publib.boulder.ibm.com/eserver/v1r1/en\\_US/index.htm?info/icmain.htm](http://publib.boulder.ibm.com/eserver/v1r1/en_US/index.htm?info/icmain.htm)

For more information on enclaves, see Chapter 3, "Creating and Using Enclaves," on page 31.

---

## Enclave Services and EWLM

The enclave services related to EWLM are:

- IWM4CON . This service connects a calling address space to WLM and optionally indicates that this work manager will participate in EWLM.
- IWM4DIS. This service allows the caller to disconnect from WLM. If the caller is connected as an EWLM participant, participation with EWLM is terminated.
- IWMCLSFY. This service associates a service class and possibly a report class with an arriving work request. With respect to EWLM, this service allows you to specify the correlator of a parent work request to start a new sub-work request, either implicitly, when creating an enclave, or explicitly, with the use of the IWMESTRT service.
- IWM4ECRE. This service creates an enclave. An optional parameter indicates whether the work manager starts a new EWLM work request implicitly.
- IWM4EDEL. This service stops all existing work requests that are still active and then deletes an enclave.
- IWMESTRT. This service allows work managers that participate in EWLM to explicitly indicate the start of an EWLM work request.
- IWMESTOP. This service allows work managers that participate in EWLM to explicitly indicate the end of an EWLM work request and all of its sub-work requests.
- IWMEBLK. This service allows work managers that participate in EWLM to indicate that processing of a work request is blocked while it waits for a sub-work request in another application to complete.
- IWMEUBLK. This service allows work managers that participate in EWLM to indicate that processing of a work request is no longer blocked.
- IWMEGCOR. This service allows work managers that participate in EWLM to retrieve the correlator for a given work request handle or the maximum length of a correlator.

## Modelling Your Business Transactions

With the introduction of support for EWLM, there are two ways of modelling your business transactions:

- Using enclaves without EWLM participation
- Mapping EWLM work requests, either closely or loosely, to enclaves

### Using Enclaves without EWLM Participation

With this model, a business unit of work is represented by the lifetime of the enclave under which it will be processed, that is, from the time the enclave is created until it is deleted. At the time the enclave is created, the business unit of work is classified and assigned a WLM service class. The work manager does not further interact with EWLM to identify when such a business unit of work starts and stops, and the work manager is not an EWLM participant.

### Mapping EWLM Work Requests to Enclaves

With the EWLM model, a business transaction is measured by the EWLM ARM agent and may be referred to as a *work request*. A work request lasts from the time it is started until the time that it is stopped. Only EWLM participants can start and stop work requests. Work requests are mapped to an enclave, as the enclave is the MVS means to manage (with SRM) transactions directly across address space boundaries.

Work requests may be started and stopped explicitly by the EWLM-participant application. In this case, the work requests can be said to be loosely mapped to enclaves. Alternatively, work requests may be started implicitly when the enclave is created, and stopped implicitly when the enclave is deleted. In this case, the work requests can be said to be closely mapped to enclaves. Note that such work requests do not differ from the non-EWLM model, except that:

- EWLM is notified about the work request's start and stop time
- EWLM may optionally classify the work request based on its end-to-end policy

It is also possible to combine the explicit and implicit starting and stopping of work requests. For example, a work request may be started explicitly but stopped implicitly. The possibilities, along with the associated WLM services, are shown in Table 13.

*Table 13. Starting and Stopping Work Requests*

Start	WLM Service	Stop	WLM Service
Implicit	IWM4ECRE	Implicit	Work request is stopped by IWM4EDEL
	IWM4ECRE	Explicit	IWMESTOP
Explicit	IWM4ECRE with ESTRT=EXPLICIT and subsequent IWMESTRT	Implicit	Work request is stopped by IWM4EDEL
	IWM4ECRE with ESTRT=EXPLICIT and subsequent IWMESTRT	Explicit	IWMESTOP
Explicit	IWM4ECRE with ESTRT=EXPLICIT_SINGLE and subsequent IWMESTRT	Implicit	Work request is stopped by IWM4EDEL
	IWM4ECRE with ESTRT=EXPLICIT_SINGLE and subsequent IWMESTRT	Explicit	IWMESTOP

An EWLM participant can process multiple single work requests under the same enclave. The underlying assumption is that an enclave can only process one work request at a time. Within a work request, however, processing of sub-work requests is allowed. The ARM transaction model requires that nested sub-work requests be completed before the nesting work request completes. WLM enforces this requirement.

## Connecting with WLM as an EWLM-Participant

A work manager becomes an EWLM participant when it connects to WLM through the IWM4CON service with the following options:

- WORK\_MANAGER=YES
- EWLM=YES

An optional GROUPNM parameter specifies an application group to which the work manager belongs.

The IWM4CON service connects a subsystem to WLM so that workload manager services can be used. Note that the default for the EWLM parameter is NO, which indicates that:

- The work manager interacts only with WLM. No interaction with EWLM takes place.

- The use of the IWMESTRT and IWMESTOP services, as well as the use of the ESTRT parameter on the IWM4ECRE service, is not permitted.

For more information, see Chapter 65, “IWM4CON – Connecting to Workload Management,” on page 533.

## Disconnecting from WLM

The IWM4DIS service allows the caller to disconnect from the workload management services, and thus also terminate EWLM participation. For more information, see Chapter 66, “IWM4DIS – Disconnecting from Workload Management,” on page 553.

## Creating an Enclave

The IWM4ECRE service creates an enclave. An optional parameter indicates whether the work manager implicitly starts an EWLM work request:

- ESTRT=IMPLIED specifies that a work request is started implicitly when the enclave is created.
- ESTRT=EXPLICIT specifies that the work manager explicitly indicates the start of an EWLM work request by invoking the IWMESTRT service.
- ESTRT=EXPLICIT\_SINGLE specifies the same as option ESTRT=EXPLICIT and, in addition, the application ensures that only one work request is active. No nested calls to IWMESTRT are allowed.

If the ESTRT=EXPLICIT\_SINGLE option is specified, the CPU consumption on all EWLM enclave services (IWMEGCOR, IWMESTRT, IWMESTOP, IWMEBLK, IWMEUBLK) will be reduced.

For further details refer to the corresponding macro descriptions.

- ESTRT=NEVER specifies that this enclave will never use any EWLM-related enclave services (IWMEGCOR, IWMESTRT, IWMESTOP, IWMEBLK, IWMEUBLK) after the enclave has been created, even if the work manager has registered (IWM4CON or IWMCONN) with EWLM=YES. Also, IWM4ECRE will not start an EWLM work request on the enclave and will not do any EWLM-related processing.

The use of the ESTRT parameter is allowed only when the work manager previously connected to WLM with IWM4CON EWLM=YES.

An optional parameter, WORKREQ\_HDL, allows the caller to get the work request handle of an implicitly created work request. The handle can be used subsequently with any of the other new services (IWMEBLK, IWMEUBLK, IWMESTOP and IWMEGCOR). The WORKREQ\_HDL parameter is valid only with ESTRT(IMPLIED).

For more information, see Chapter 67, “IWM4ECRE – Create an Enclave,” on page 561.

## Deleting an Enclave

The IWM4EDEL service stops all existing work requests that are still active and then deletes an enclave.

## Classifying Work Requests

For WLM management and reporting purposes, classification is done when an enclave is created. The resulting service class is assigned to the enclave and remains until the enclave is deleted.



For EWLM purposes, classification for an EWLM-participant work request is always done at the first hop, that is, the first application tier in a domain that accepts a request. Subsequent tiers usually do not classify the work request again unless the domain border has been crossed or different EWLM policies are installed on the sending and receiving tier. However, each time a new work request is started, a new correlator is created that must be obtained and forwarded to each subsequent application tier, if any. To obtain the correlator, you use the IWMEGCOR service.

If the work manager represents the first hop, the classification structure mapped by the IWMCLSFY service is used to classify the work request. To specify the EWLM correlator associated with the parent correlator, you use the EWLM\_CORR parameter on the IWMCLSFY service.

The parent correlator is the correlator that is passed from the sending application tier. It contains, among other information, the transaction class of the work request. When a sub-work request is started, the parent correlator must be used to indicate to EWLM that a sub-work request is to be started, and the current correlator can be derived from the parent correlator. Otherwise, the current correlator is generated by classifying the new work request.

z/OS passes classification attributes to ARM as property names. These attributes are used in case EWLM classification on z/OS is required, for example, because no correlator was passed into an EWLM participant, or the work manager represents the first hop. For EWLM classification it is critical that these names are in upper case in the EWLM policy. The attribute value, however, can be upper or lower case.

The classification attributes passed to ARM are:

- ACCTINFO
- COLLECTION
- CONNECTION
- CORRELATION
- LUNAME
- NETID
- PACKAGE
- PERFORM
- PLAN
- PRCNAME
- PRIORITY
- PROCESSNAME
- SCHEDENV
- SUBCOLN
- SUBSYSPM
- TRXCLASS
- TRXNAME
- USERID

For details on the attributes, see Chapter 12, “IWMCLSFY – Classify Work,” on page 135.

## Starting and Stopping Work Requests Explicitly

WLM services allow work managers that are EWLM participants to explicitly indicate the start and end of an EWLM work request:

- IWMESTRT, for start
- IWMESTOP, for stop

Required parameters for each service specify the enclave under which the work request is processed and the handle that represents the work request. On IWMESTRT you also specify classification attributes and a parent correlator when a sub-work request should be started.

For more information, see Chapter 28, “IWMESTRT – Starting a Work Request,” on page 263 and Chapter 27, “IWMESTOP – Stopping a Work Request,” on page 255.

## Continuing a Work Request at Another Application

To continue a work request at another application, a work manager must pass the correlator of the current work request to that application. To retrieve the correlator for a given work request handle, you use the IWMEGCOR service. The service can also be used to return the maximum length of a correlator. The receiving application would in turn use this correlator as the parent correlator to start a sub-work request.

## Blocking/Unblocking Work Requests

Services are provided for the blocking and unblocking of work requests. You would block the processing of a work request when it is waiting for a work request in another application to complete.

- IWMEBLK indicates that processing of a work request is blocked.
- IWMEUBLK indicates that processing of a work request is no longer blocked.

For more information, see Chapter 18, “IWMEBLK – Work Request Blocked,” on page 195 and Chapter 30, “IWMEUBLK – Work Request No Longer Blocked,” on page 277.

## Enclave Services and the ARM API

For a work manager that is an EWLM participant, the enclave services internally invoke the ARM services. A work manager becomes an EWLM participant when it connects to WLM through the IWM4CON service with EWLM=YES. Table 14 shows the ARM services that are invoked by the enclave services.

Table 14. WLM Enclave Services and ARM APIs

Enclave Service	Description	ARM Service	Parameters
IWM4CON	Connect to WLM, identify the work manager as an EWLM participant	arm_register_application	app_name = SUBSYS
		arm_register_transaction	
		arm_start_application	app_group_name = GROUPNM app_instance_name = SUBSYSNM
IWM4ECRE with ESTRT(IMPLIED)	Create an enclave	arm_start_transaction	
IWMESTRT	Start a work request	arm_start_transaction	
IWMESTOP	Stop a work request	arm_stop_transaction	
IWMEBLK	Indicate that processing of a work request is blocked	arm_block_transaction	

Table 14. WLM Enclave Services and ARM APIs (continued)

Enclave Service	Description	ARM Service	Parameters
IWMEUBLK	Indicate that processing of a work request is no longer blocked	arm_unblock_transaction	
IWMEDELE	Delete an enclave	arm_stop_transaction (if applicable)	
IWM4DIS	Disconnect from WLM	arm_destroy_application	

## Instrumenting a C Application for ARM

Instrumenting an application for ARM is described in the EWLM topic of the @server Information Center, on the Internet at [http://publib.boulder.ibm.com/eserver/v1r1/en\\_US/index.htm?info/icmain.htm](http://publib.boulder.ibm.com/eserver/v1r1/en_US/index.htm?info/icmain.htm). The sections that follow describe how to compile, bind and run an ARM-instrumented application on z/OS.

## Compiling an ARM-instrumented Application

When compiling an ARM-instrumented application, the application should use `#include "arm4.h"` to include the ARM header file. As a result, the `arm4.h` header file library, `SYS1.SIEAHDRV.H`, must be included in the `SEARCH/LSEARCH` path.

### Using the C/C++ Compiler in BATCH

Figure 11 shows sample JCL that illustrates the required options to compile an ARM-instrumented 31-bit XPLINK application in MVS batch.

```

/MYJOB jobcard
/**
//          JCLLIB ORDER=CBC.SCCNPRC
/**
//COMPILE EXEC CBCC,
//          CPARM='OPTF(DD:OPTIONS)',
//          INFILE='mysource(mymember)',
//          OUTFILE='myobjectdeck(mymember),DISP=SHR'
//COMPILE.OPTIONS DD *
LSEARCH(//'SYS1.SIEAHDRV.+')
SEARCH(//'SYS1.SIEAHDRV.+')
LANGLVL(EXTENDED)
XPLINK                               Non-XPLINK: omit, 64BIT: LP64
SOURCE
other options

```

Figure 11. Sample JCL for Compiling an ARM-instrumented Application in MVS Batch

For a 31-bit non-XPLINK application, omit XPLINK (third line from the bottom in the example).

For a 64-bit application, specify the LP64 option instead of XPLINK. (The LP64 option is supported by the z/OS V1R6 or higher C/C++ compiler.)

### Using the C/C++ Compiler under z/OS UNIX System Services

Run the `c89` or `c++` z/OS UNIX System Services shell command to compile your program and store the object. For C++ compiles add the `-+` option. Table 15 on page 58 shows examples of compiling under z/OS UNIX System Services in the various environments.

Table 15. Sample Commands for Compiling Applications under z/OS UNIX Systems Services

Environment	Sample Command
31-bit XPLINK	<code>c++ -c -o mymain.o -Wc,"DLL,XPLINK,LANGLVL(EXTENDED)" \ -I/"'SYS1.SIEAHRV.+' mymain.c</code>
31-bit non-XPLINK	<code>c++ -c -o mymain.o -Wc,"DLL,LANGLVL(EXTENDED)" \ -I/"'SYS1.SIEAHRV.+' mymain.c</code>
64-bit	<code>c++ -c -o mymain.o -Wc,"DLL,LP64,LANGLVL(EXTENDED)" \ -I/"'SYS1.SIEAHRV.+' mymain.c</code>

## Binding an ARM-instrumented Application

### Using the Binder in BATCH

Figure 12 shows sample JCL that illustrates the required options to compile an ARM-instrumented 31-bit XPLINK application in MVS batch.

```
//MYJOB      jobcard
//*
//          JCLLIB ORDER=CBC.SCCNPRC
//*
//BIND      EXEC CBCXB, Non-XPLINK: CBCB, 64BIT: CBCQB
//          BPARAM='CALL,NOMAP,RENT,DYNAM=DLL',
//          OUTFILE='myloadlib,DISP=SHR',
//          INFILE='myobject(mymain)'
//BIND.ARMSIDE DD DISP=SHR,DSN=SYS1.SIEASID
//BIND.MYOBJ DD DISP=SHR,DSN=myobject
//SYSIN DD * INCLUDE MYOBJ(mymem2, ...)
//          INCLUDE ARMSIDE(LARM43X)
//          ENTRY CEESTART 64BIT: CELQSTRT
//          NAME mymain(R)
```

Figure 12. Sample JCL for Binding an ARM-instrumented Application in MVS Batch

For 31-bit non-XPLINK applications, use the procedure CBCB rather than CBCXB (line 5 in the example), and specify SIEASID member LARM431 rather than LARM43X (third line from the bottom in the example).

For 64-bit applications (compiled with compiler option LP64) use the procedure CBCQB rather than CBCXB (line 5 in the example), and specify SIEASID member LARM464 rather than LARM43X (third line from the bottom). In addition, specify an ENTRY point for the binder of CELQSTRT instead of CEESTART (second line from the bottom in the example).

### Using the Binder under z/OS UNIX System Services

Run the `c89 / c++` command specifying the appropriate side-deck for the ARM DLL and the correct binder options. Table 16 shows sample commands for 31-bit and 64-bit environments.

Table 16. Sample Commands for Binding Applications

Environment	Sample Command
31-bit XPLINK	<code>c++ -o mymain -Wl,"DLL,XPLINK" mymain.o mymem2.o \ /usr/lib/libarm4_3x.x</code>
31-bit non-XPLINK	<code>c++ -o mymain -Wl,"DLL" mymain.o mymem2.o \ /usr/lib/libarm4_31.x</code>
64-bit	<code>c++ -o mymain -Wl,"LP64,DLL" mymain.o mymem2.o \ /usr/lib/libarm4_64.x</code>

## Running an ARM-Instrumented Application

The ARM-instrumented application can run:

- In batch or as a started task
- Under z/OS UNIX System Services

### Running an Application in Batch or as a Started Task

To run an ARM application in batch or as a started task, you must ensure that either:

- The data set SYS1.SIEALNKE is in the system LINKLIST. This is the default, and is the recommended approach.
- The library is in the application's STEPLIB or JOBLIB, as appropriate.

### Running an Application under z/OS UNIX System Services

To run an ARM application under z/OS UNIX System Services, you must ensure that the path /usr/lib is in the application's LIBPATH.



---

## Chapter 5. Using the Queueing Manager Services

The queueing manager services are intended for queueing managers to use to manage server (execution) address spaces and the work requests they process to meet service class performance goals. Through queueing manager services, workload management maintains the queues for passing work requests from the queueing manager to its servers. A **queueing manager** is a subsystem that queues work requests to workload management for execution in server address spaces.

Workload management dynamically starts and maintains server address spaces as required to meet the queueing manager's workload. Therefore, installations do not have to manage the address spaces manually, nor do they have to monitor workload fluctuations that change the number of address spaces needed for the work to meet its goals. Workload management automatically adjusts to changes in the workload.

For queue managers using the services, workload management spreads the work across multiple address spaces, providing workload isolation and greater scalability based on workload demands. For a queueing manager that queues and executes work all in the same address space, sometimes encountering storage overlay problems, the services provide an incentive to change to a multiple address space configuration.

This chapter describes how to use the queueing manager services, and suggests the information you should provide to your customers so they can properly set up the required service definitions.

**Note:** The queueing manager services uses application environments. (See "Updating a Service Definition with Application Environment Information" on page 69 for a discussion of defining policy for application environments.)

---

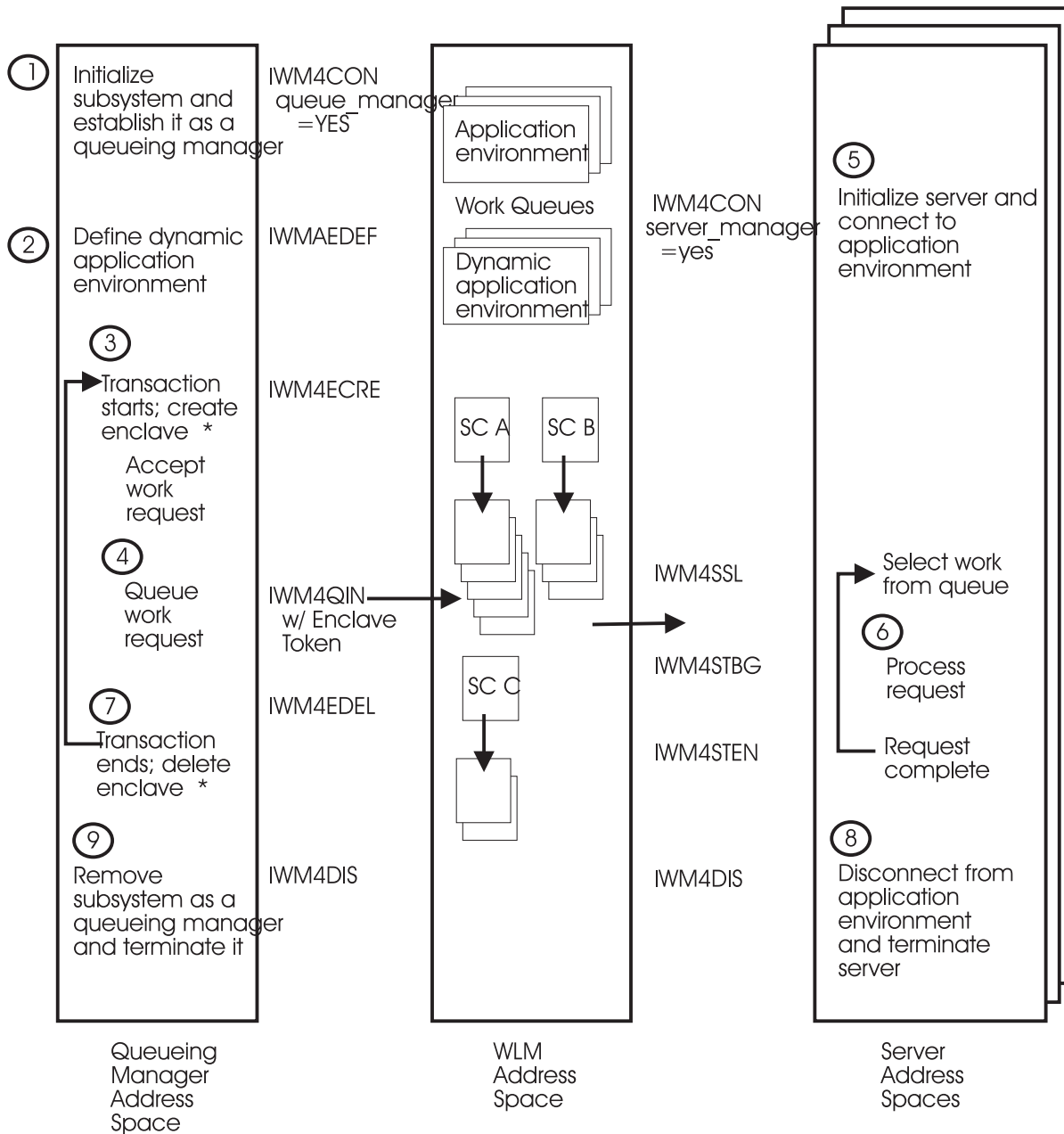
### Example of Using the Queueing Manager Services

This section describes how a queueing manager can use the services to:

- Isolate different types of requests into separate server address spaces for integrity, security, and operational reasons.
- Classify work into service classes according to business goals.
- Manage the application execution in the server address space as a continuation of the originating unit of work.
- Specify dynamic management of server address spaces, and other resources to meet the service class goals of the work requests.
- Report service class goals against actual service class performance
- Report response time information.

Figure 13 on page 62 shows an example of how a queueing manager can use the services. The example shows the services used to achieve the objectives listed above. The services are intended to be used with the enclave services, and must be used independently of the execution delay monitoring services.

## Queueing Manager Model



\* Enclave can be created and deleted directly by the queueing manager as shown, or by its caller.

Figure 13. Services for a queueing manager

The figure shows a queueing manager address space starting up, connecting to workload management, and queueing work requests to workload management. The server address spaces are created dynamically by workload management as needed. When the server initializes, it connects to workload management, which allows it to select work from the queues. The server address space indicates to workload management when each request starts and ends, so the work request can be properly managed and its performance statistics reported. The following steps describe the flow illustrated in Figure 13.



### 1. **Establish queueing manager**

The queueing manager address space starts up through either a manual start or customer automation. During its initialization, it issues the IWM4CON service with the QUEUE\_MANAGER=YES parameter, and provides the subsystem type (SUBSYS parameter), the subsystem name (SUBSYSNM parameter), and, optionally, the node name (NODENM parameter) to identify the type of work associated with the queueing manager. If you need to create an independent enclave, then the queueing manager should also specify WORK\_MANAGER=YES (the default) on IWM4CON. If you are creating dependent enclaves, you do not need to specify WORK\_MANAGER=YES.

If the queueing manager needs to take some action when workload management deletes work requests that the queueing manager previously queued, specify a connect exit routine on the QMGR\_EXIT@= parameter. Workload management deletes all queue requests when the queue manager disconnects from workload management or the application environment is deleted. This exit gets control when workload management has deleted a work element from its queue. Input to the exit is mapped by the list form of the IWMQCXIT macro. For information about the IWMQCXIT exit, see “Using the Queueing Manager Connect Exit” on page 69. Workload management provides the exit with the information passed to workload management when a work element is queued and an indication that the work element has been removed from the queue.

### 2. **Define dynamic application environment**

Optionally, you can define a dynamic application environment using the IWM4AEDF service. The scope of this newly defined DAE only applies to the queueing manager that invoked this service. For further information, refer to Chapter 64, “IWM4AEDF — Defining Dynamic Application Environments to Workload Management,” on page 523.

### 3. **Create enclave**

The queueing manager receives a work request and must use an enclave to manage it. Depending on the environment, the queueing manager can use an existing enclave or create a new enclave, either dependent or independent, using the IWM4ECRE service.

If running under the requestor’s dispatchable unit, IWMESQRY can be used to determine whether the requestor belongs to an enclave and if so, which enclave. If running under a different dispatchable unit, it is the subsystem’s responsibility to pass information on any existing enclave along with the work request. For example, the subsystem could pass information it obtained from IWMESQRY while running under the requestor’s dispatchable unit.

If the requestor does not belong to an enclave but has an address space transaction (for example, it is a TSO user or a batch job), the queueing manager can create a dependent enclave to represent a continuation of the requestor’s transaction. This requires that the requestor be the home address space.

If there is no existing enclave or address space transaction, such as when the requestor is on another system, the queueing manager must create an independent enclave to begin a new transaction. This requires the queueing manager to classify the work request.

### 4. **Queue request**

The queueing manager uses the IWM4QIN service to add the work request to a workload management queue. The application environment, enclave token, and optional userid for resource access control are provided as input to workload management. The service class is determined from the enclave token, and the

request is added to a queue associated with that service class within the specified application environment. You can optionally pass information to the server address space when it selects this work request. Workload management does not read or modify this data in any way.

Workload management stages work requests between the queueing manager address space and the server(s), but the queueing manager is still responsible for managing the flow of work requests and handling timeout and abnormal conditions where servers are failing to properly process requests. Workload management detects and reacts to certain error conditions such as JCL errors in the procedure used to start the server and repeated, unexpected terminations of the server address space. For more information, see “Defining Application Environments” in *z/OS MVS Planning: Workload Management*.

If the queueing manager needs to remove a work request that it previously queued through the IWM4QIN service before it has been processed by the server, it uses the IWM4QDE service. This service is provided for exceptional circumstances, such as:

- Timeout of the work request
- An external request to cancel a queued request
- Queueing manager recovery.

A queueing manager must not insert requests for a dynamic and static application environment with the same application environment name concurrently.

#### 5. **Establish server address space**

When the first request is queued to an application environment, workload management detects that there are no active servers for the request, and automatically starts one. The MVS procedure name and start parameters are taken from the application environment definition in the service definition. As the workload fluctuates, workload management adjusts the number of server address spaces so the goals of the work are met.

When the server initializes, it must establish itself as a server address space using the IWM4CON service with `SERVER_MANAGER=YES` parameter, and indicate which application environment it is servicing. The subsystem type and name specified on the server connect must match the values specified on the associated queueing manager connect.

Workload management creates as many server address spaces as are needed to meet the goals of the work running in the servers, unless the application has limited the number of server instances that workload management can create using IWM4SLI.

Immediately after invoking IWM4CON, you have the option of using IWM4SLI to control the number of server instances that WLM will create. Use the `AE_SERVERMAX` parameter to establish a maximum number — this is particularly useful for applications such as MQSeries Workflow that connect to backend applications supporting a limited number of parallel connections. Use the `AE_SERVERMIN` parameter to establish a minimum number — this allows an application to keep a number of servers active, even during low utilization periods. In addition, you can specify `AE_SPREADADMIN=YES` to ensure that the defined minimum number of servers are distributed evenly across all of the service classes used to execute work requests in the application environment.

There will be at least as many servers for an application environment as there are unique service classes associated with the work requests, even if the workload is low. This is so workload management can separately manage work with different service class goals.

Applications can optionally give workload management the control about the number of server instances per server address space. Directly after IWM4CON, you can use IWMSINF to obtain recommendations from workload management about the number of server instances to be started. WLM will pass the number of instances to be started in addition to the already running server instances to the server address space. The caller must have previously connected to WLM using the IWM4CON service specifying SERVER\_MANAGER=YES, SERVER\_TYPE=QUEUE, and MANAGER\_TASKS=YES. See “Managing the Number of Server Instances per Server Address Space” on page 67 for a more detailed explanation on using IWMSINF.

Because the workload management services used by server address spaces are available to problem programs, a SAF call is made as part of the connection process to allow the installation to protect against malicious use or damage due to incorrect startup definitions. This SAF check permits the server address space to be totally unauthorized (no APF authorization required). The RACF-supported resource for this call is of the class SERVER and has the name:

```
sstype.subsys.applenv[.nodenm]
```

where:

- *sstype* is the subsystem type
- *subsys* is the subsystem name of the queueing manager address space
- *applenv* is the name of the application environment being serviced
- *nodenm* is an optional qualifier of the queueing manager

Workload management picks up these values from the IWM4CON parameters, SUBSYS=, SUBSYSNM=, APPLENV=, and NODEDM respectively. For more information, see “Workload Management Migration” in *z/OS MVS Planning: Workload Management*.

## 6. Process work request

The server uses the IWM4SSL service to remove a work request from the queue associated with its application environment and service class. If no requests are queued, workload management suspends the task issuing the IWM4SSL until work is available. After a request is selected, the server uses the IWM4STBG service to indicate the processing of the request is beginning. At this point the task joins the enclave identified when the work was queued through IWM4QIN. When the server is finished processing the request, it uses the IWM4STEN service to cause the task to leave the enclave.

Almost all of the processing in a server address space should be on behalf of a work request and occur between the IWM4STBG and IWM4STEN calls. A server address space should not have other tasks performing unrelated processing as this could interfere with effectively managing the enclave to its goals.

If the queueing manager provided a userid on the IWM4QIN service, IWM4STBG sets up a SAF environment for the work request to control access to resources.

Alternative task structures:

This example shows a single task issuing the IWM4SSL, IWM4STBG, and IWM4STEN services: one task within the server address space selects work from the queue, processes the request, then loops back to select more work. Queueing manager services does not impose any rules limiting the ways a server can organize its tasks. The following task structures are also possible:

- Multiple tasks with single thread execution.

One task selects work from the queue, selects a “worker” subtask to process the request, posts the worker task, and then waits to be posted by the worker task when the processing of the request is complete.

- Multiple tasks with parallel work selection and execution.

There is a set of “worker” subtasks within the server address space, each of whom selects work from the queue, processes the request, and loops back to obtain a new request.

At the time the server connects to workload management (IWM4CON), the server must specify how many tasks will execute work in parallel (PARALLEL\_EU).

#### 7. **Delete enclave**

The server address space informs the queueing manager using its own interface that a work request is complete. If the queueing manager had previously created an enclave for the work request, it determines if there are other work requests active for the same enclave. If there are none, the queueing manager deletes the enclave and returns to the originator of the request.

#### 8. **Terminate server address space**

Under normal circumstances, it is expected that a server address space will continue to select work until told by workload management that the space is no longer needed through a return and reason code (x'0C14') on the IWM4SSL service. The server is then expected to complete any work requests already selected, clean up, disconnect from workload management through the IWM4DIS service, and terminate. Possible reasons for this IWM4SSL return code are:

- The operator entered one of the following commands for the application environment:
  - VARY WLM,APPLENV=,QUIESCE
  - VARY WLM, DYNAPPL=,QUIESCE or
  - VARY WLM,APPLENV=,REFRESH
  - VARY WLM,DYNAPPL=,REFRESH
- A policy was activated from a new service definition which no longer includes the application environment associated with the server.
- WLM determines that the server is not needed to meet the goals of the enclave's service class.

The server should make reasonable recovery attempts when errors occur. If the server encounters a failure that forces it to terminate, it should disconnect from workload management as part of its shutdown. If the server address space terminates without explicitly disconnecting from workload management, workload management detects the termination and performs the IWM4DIS processing at that time. If workload management detects five unexpected server disconnects or address space terminations within ten minutes, it will put the application environment into a STOPPED state. This means that no new servers are created for the application environment until the VARY WLM,APPLENV=,RESUME or VARY WLM, DYNAPPL=RESUME command is issued for it. Existing server address spaces continue to run work while the application environment is in the STOPPED state, as long as they are operating normally.

#### 9. **Terminate queueing manager address space**

When the queueing manager terminates normally, it is expected to quiesce its activities, drain its queues of pending work, and disconnect from workload

management using the IWM4DIS service. If the queueing manager address space terminates without first disconnecting from workload management, workload management detects the termination and performs the IWM4DIS processing at that time.

Workload management returns control immediately to the caller of IWM4DIS, and asynchronously performs server clean up. It purges all queued, but unselected, requests and attempts to terminate all associated server address spaces. Because workload management depends on server address spaces to terminate voluntarily when requested, and because workload management must wait for a server to request work prior to telling it to terminate, server address spaces could remain for a significant amount of time after the queueing manager terminates.

## Managing the Number of Server Instances per Server Address Space

A server address space contains one or multiple server instances which all select work requests from a work queue. Without using the new interface IWMSINF, the server address space must tell WLM during start up how many server instances will be started (parameter: PARALLEL\_EU on IWM4CON).

The new interface IWMSINF allows the application to obtain the number of server instances from WLM. The following Figure 14 shows an example on how the server manager address space can use IWMSINF.

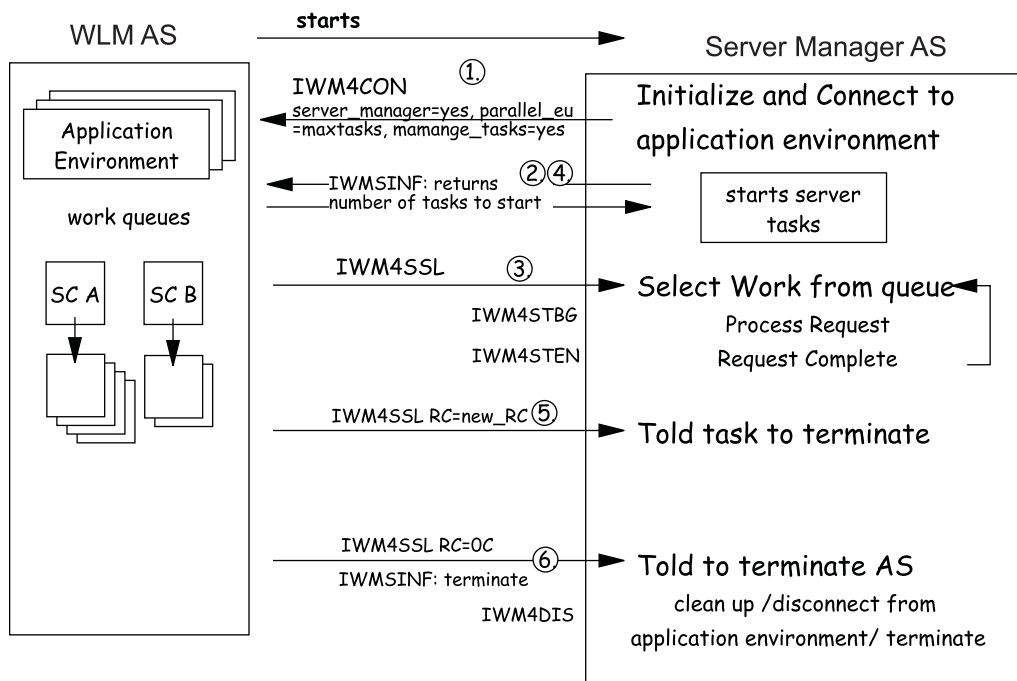


Figure 14. Exploiting IWMSINF

1. To use the new interface, the server manager address space connects to WLM with IWM4CON specifying the option `MANAGE_TASK=YES`. All server address spaces which connect to the same application environment must specify the same setting of `MANAGE_TASK`. The options `SERVER_MANAGER=YES` and `SERVER_TYPE=QUEUE` have to be defined, too.
2. After connecting the server manager address space must invoke the new service IWMSINF to obtain the number of server instances to be started initially.

3. The server address space starts the server instances which select work from WLM by using IWM4SSL.
4. The server manager address space reinvokes IWMSINF again to "listen" for recommendations from WLM for the number of tasks to start.
5. IF WLM wants to terminate one or multiple server instances, it resumes IWM4SSL with a new return code telling the server instance to terminate.
6. If WLM wants to terminate the server address space, it resumes IWM4SSL and IWMSINF with return code that tells the server to terminate.

**Note:** WLM can only manage the number of server instances per server address space if there is a linear relationship between the number of instances in an address space and the virtual storage consumed in the address space. This implies that all instances allocate about the same amount of virtual storage.

## Directing Work Requests to a Specific Server Region

Existing workload manager interfaces allow a control region to queue work requests to a pool of server regions for a service class. The underlying assumption is that each work request represents one or multiple contiguous transactions. This transaction is represented by an enclave which is created when the work request is inserted and which is removed when the application completed its processing for the work request. It is assumed that no information is left in any temporary structure in the system for following work requests.

But there are cases where information must be preserved across multiple "independent" work requests. The information left behind lives only in the virtual storage of the address space. Following work requests requiring this information must now be directed to the server region which has this information.

The solution is that the server region is able to obtain a region token at select time (IWM4SSL) and passes this region token to the queue manager. The queue manager is now able to route subsequent requests directly to this server region by specifying the region token on IWM4QIN. A new interface (IWM4TAF) allows the server or control region to mark the server region as being needed by follow-on work requests. WLM will ensure that server region stays alive until all temporal affinities have been removed.

### Notes:

1. The requests which are directly routed to server regions are outside of the control scope of WLM. Therefore WLM is not able to manage the number of server regions properly if the majority of requests is directly routed to the regions and not queued for being picked up by the WLM managed server pool. It is assumed that requests which are outside of the scope of WLM represent only a minor portion of all work requests processed by the application.
2. The application should carefully use the new interface IWM4TAF. WLM will not terminate a server region if it is marked of having a temporal affinity. This can significantly delay the behaviour of WLM operator commands such as refresh and quiesce for these application environments. It is assumed that temporal affinities live only a short period (a few minutes) in the system and that they do not represent the majority of the work requests of the application (see also Note 1.)

---

## Updating a Service Definition with Application Environment Information

When a customer installs a subsystem that makes use of the queueing manager services, the service administrator must define one or more **application environments** in the workload management service definition. An application environment is a group of application functions requested by a client which execute in a server address space. Workload management dynamically adjusts the number of address spaces servicing the application environment to meet the goals of the work.

If you use the queueing manager services, make sure to document the information needed by the customer's service administrator to define the application environments. For example, you should provide the following:

- A technique for grouping work into application environments.  
Each application environment should represent a named group of server functions that require access to the same application libraries. Having a named group facilitates library security, application program change control, performance management, and system operation.  
For example, a set of related payroll applications might be grouped into one application environment because of their similar run-time requirements. The customer can name the application environment for these payroll applications in the service definition. Workload management then starts and stops server address spaces to process the work in the payroll applications.
- The queueing manager subsystem type  
The queueing manager specifies the subsystem type in the SUBSYS parameter of the IWM4CON service. A service administrator defines the subsystem type when specifying the application environment in the service definition. Make sure you do not use a subsystem type already in use by another subsystem.
- Samples of JCL start procedures and start parameters for a server address space.  
You should provide your customer with sample procedures and start parameters for the server address spaces.

For more information on how to define an application environment and a list of subsystem types currently used, see *z/OS MVS Planning: Workload Management*.

**Note:** When defining an application environment, you must specify whether or not workload management can start multiple address spaces for the subsystem. In the case of a queueing manager, you can only choose Option 1, Managed by WLM, or Option 2, Limited to a single address space per system. For more information, see the "Defining Application Environments" chapter in *MVS Planning: Workload Management*.

---

## Using the Queueing Manager Connect Exit

A queueing manager can optionally provide the name of an exit routine on the QMGR\_EXIT@ parameter of IWM4CON when it connects to workload management. Through this exit, the queueing manager is informed when workload management has had to delete queued work requests associated with the queueing manager. Workload management deletes all queued requests when:

- The installation deletes the application environment (that is, activates a service policy that does not contain the application environment).

- The queuing manager address space disconnects.

When the exit routine is invoked, register 1 contains the address of a parameter list mapped by the list form of the IWMQCXIT macro. The parameter list includes an indicator of what action workload management has taken and the input values specified previously by the queuing manager when it queued the work request using IWM4QIN.

The execute and standard form of IWMQCXIT are intended for use only by the operating system.

## Exit Routine Environment

The queuing manager connect exit routine receives control in the following environment:

**Authorization:**

Supervisor state and PSW key 0

**Dispatchable unit mode:**

Task

**Cross memory mode:**

any PASN, any HASN, any SASN

**AMODE:**

31-bit

**ASC mode:**

Primary

**Interrupt status:**

Enabled for I/O and external interrupts

**Locks:**

No locks held

**Serialization:**

None

**Location:**

The connect exit must be a resident routine callable from any address space and must remain available after the queuing manager disconnects or terminates. Input parameter list is in the primary address space.

The input parameter list is in pageable storage addressable in the primary address space, but should not be changed by the exit.

**Exit Recovery:**

The system may discontinue calling the exit upon repetitive, abnormal completions, that is, where an error within the exit percolates to the system recovery routine. The exit may optionally establish a functional recovery routine (FRR) or ESTAEX for any needed recovery or cleanup of its resources.

## Register Usage

Upon entry to the exit, the register contents are as follows:

- Register 0 = not defined
- Register 1 contains the address of the input parameter list
- Register 2-13 = not defined



- Register 14 = return address
- Register 15 - entry point address

Upon entry to the exit, the access register contents are undefined.

Upon return from the exit, the register contents are expected to be:

- Register 0 = reason code if GR15 return code is non-zero
- Registers 1-14 = not defined (need not be restored to value on entry).
- Register 15 = return code

Upon return from the exit, the access register contents are unchanged.

## Restrictions

The exit routine should not invoke functions or suspend execution which could prevent return to the caller for a protracted period. This includes the use of system services which either explicitly or implicitly give control back to the system. In this context, “protracted period” means durations of one second or longer. When such processing is required, the exit should use asynchronous techniques.

### Example

Following is an example of invoking the IWMQCXIT macro instruction and the resulting parameter list mapping. This parameter list is passed to the queueing manager connect exit specified on the QMGR\_EXIT@ parameter of IWM4CON.

```

IWMQCXIT MF=(L,MYQCXITPL)

MYQCXITPL DS 0D                ++ IWMQCXIT PARAMETER LIST
MYQCXITPL_XVERSION DS XL1      ++ INPUT
MYQCXITPL_XRSV0001 DS CL1      ++ RESERVED
MYQCXITPL_XPLISTLEN DS XL2     ++ INPUT
MYQCXITPL_XACTION DS BL.8      ++ ACTION INDICATORS
MYQCXITPL_XACTION_QDEL EQU B'10000000'
                                ++ INDICATES QUEUE ELEMENT DELETED
MYQCXITPL_XRSV0005 DS CL2      ++ RESERVED
MYQCXITPL_XSECUSER_OPTIONS DS BL.8
                                ++ OPTIONS USED ON IWM4QIN
MYQCXITPL_XSECUSER_YES EQU B'10000000'
                                ++ SECUSER=YES USED ON IWM4QIN
MYQCXITPL_XETOKEN DS CL8       ++ ETOKEN VALUE FROM IWM4QIN
MYQCXITPL_XUSERDATA DS CL16    ++ USERDATA FROM IWM4QIN
MYQCXITPL_XRSV0020 DS CL4      ++ RESERVED
MYQCXITPL_XAPPLENV_ADDR DS A   ++ ADDR OF APPLICATION ENVIRONMENT
MYQCXITPL_XUSERID DS CL8       ++ USERID FROM IWM4QIN
MYQCXITPL_XRSV0030 DS CL8      ++ RESERVED
MYQCXITPL EQU *-MYQCXITPL      ++ LENGTH OF PARAMETER LIST

```



---

## Chapter 6. Using the Routing Manager Services

A **routing manager** is a subsystem that establishes and manages connections between a client and a server address space. The routing manager handles these connections rather than individual work requests; the requests are processed only after they arrive in the server address space. The routing manager is responsible for balancing the client connections across a set of eligible servers, with the assistance of the routing manager services.

Routing manager services perform two main functions:

- Automatically starting and maintaining server address spaces as needed by the workload across the sysplex. Installations then do not have to manage the address spaces manually, nor do they have to monitor workload fluctuations that change the number of address spaces needed. Workload management automatically adjusts to changes in the workload.
- Balancing the workload among the servers in the sysplex by deciding on the best server and providing the server routing information when a server is requested by the routing manager.

Routing manager services differ from sysplex routing services in the following ways:

- Routing manager services provide automatic management of address spaces.
- Routing manager services include the server's performance index when selecting the best available servers.
- With routing manager services, workload management decides on a server and passes its identity to the routing manager, instead of offering the routing manager a choice of several servers.

This chapter presents a model of how the routing manager services are intended to be used. Routing manager services combines the use of the "find server" function with application environments and enclaves. See "Updating a Service Definition with Application Environment Information" on page 69 for a discussion of defining policy for application environments.

**Note:** When defining an application environment, you must specify whether workload management can start multiple or single address spaces for the subsystem. In the case of a routing manager, both Option 1, Managed by WLM, and Option 3, Limited to a single address space per sysplex are valid. For more information, see the "Defining Application Environments" chapter in *MVS Planning: Workload Management*.

It is strongly recommended that server address spaces associated with a routing manager use enclave services to manage the work requests according to goals set by the customer. Be aware that enclaves are mutually exclusive with the execution delay monitoring services described in earlier chapters. For a discussion of enclaves and a comparison to delay monitoring, see Chapter 3, "Creating and Using Enclaves," on page 31.

---

### A Routing Manager Model

This section presents a model of a routing manager which uses workload management services to accomplish the following objectives:

- Isolate different types of requests into separate server address spaces for integrity, security, and operational reasons.

- Have MVS adjust the number of server address spaces to meet the goals of the workload.
- Have MVS balance the workload across a sysplex by selecting the best system on which to start a server.
- Associate work coming into the server with a service class goal.
- Report goals versus actuals.
- Report response time information.

Figure 15 shows the suggested services for a routing manager for managing server address spaces and balancing workload.

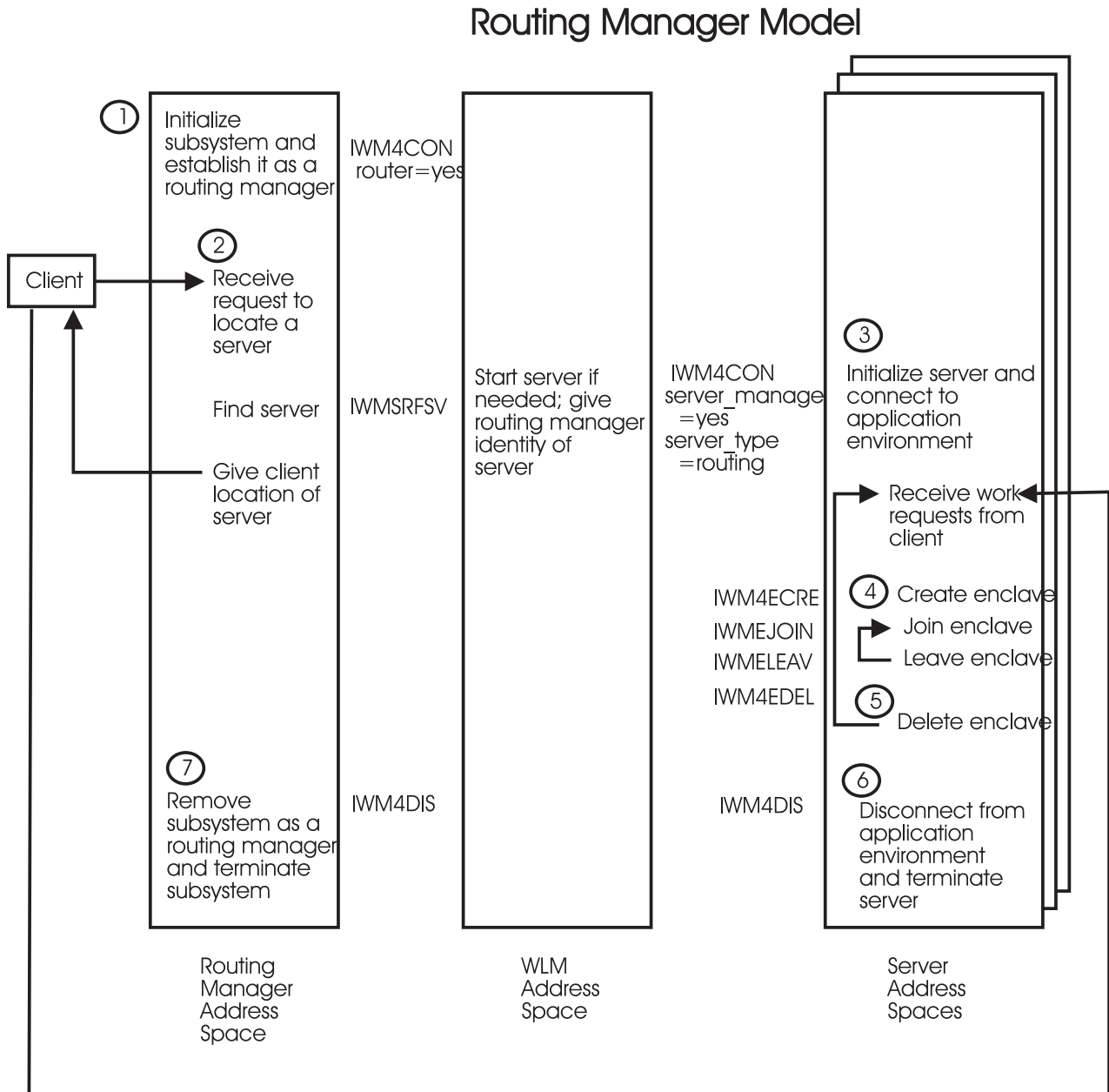


Figure 15. Example of routing manager services

Figure 15 shows a routing manager connecting to workload management and establishing itself as a routing manager. It then issues a “find server” for each client

request for server location. Workload management is then able to create server address spaces as needed and keep track of the servers across the sysplex. When the client contacts the server, the server creates an enclave on behalf of the client so the work request can be managed to the customer goals.

The detailed sequence of events in Figure 15 on page 74 is:

**1. Establish routing manager**

The routing manager address space starts up through either a manual start or customer automation. During its initialization, it issues the IWM4CON service with the ROUTER=YES parameter to indicate the intent to use routing manager services. The routing manager supplies the subsystem type (SUBSYS parameter) and subsystem name (SUBSYSNM parameter). These parameters identify the type of work associated with the routing manager.

**2. Request server routing information**

Once initialized in step 1, the routing manager is ready to receive requests to locate servers. In response to such a request, the routing manager calls the IWMSRFSV service, passing the desired application environment name (via the APPLENV= parameter). The output returned will be the routing information needed for the client to contact the selected server directly. (The specific content of this information is defined by the client and passed by the server via the SERVER\_DATA= parameter, as described in step 3.)

If no eligible servers exist, workload management will proceed to step 3 to start a server.

**3. Establish server**

If this step is necessary, the client program will be suspended until the new server address space is created (possibly on another MVS image). For an application environment defined for a routing server, workload management starts at most one server address space per system instance in the sysplex. The first server is started on the system with the most available capacity at the lowest importance level. Subsequent servers are started on other systems in the sysplex when the work running in the existing servers is not meeting its goals. However, if the workload diminishes for a routing manager, workload management does not decrease the number of servers in the sysplex. The number of servers remains at the high water mark.

It is possible to run more than one instance of a subsystem in the sysplex, meaning they have the same subsystem type, and therefore the same application environments, but different subsystem names as specified on IWM4CON. The above rules for starting servers apply separately to each subsystem instance. In other words, for each instance, you can have up to one server per system per application environment.

During server initialization, the server invokes the IWM4CON service with the SERVER\_MANAGER=YES and SERVER\_TYPE=ROUTER parameters, and information in the SERVER\_DATA= parameter to uniquely identify the server. The subsystem type and name specified must match that of the associated routing manager instances which will route clients to the server (as defined in step 1). The WORK\_MANAGER= parameter must remain the default YES so that the server can create independent enclaves in the next step. In addition, the address of an exit routine must be identified on the SRV\_MGR\_EXIT= parameter. Refer to "Using the Routing Server Connect Exit" on page 77. Workload management invokes this exit to initiate the shutdown of the server address space. The exit is expected to initiate the shutdown of the server address space. Input to the exit is mapped by the list form of the IWMSCXIT macro instruction. For more information on the quiesce process for an

application environment, see step 6 below, as well as the “Defining Application Environments” chapter in *MVS Planning: Workload Management*.

#### 4. **Create enclave**

The server address space receives a work request directly from the client and uses the IWM4ECRE service to create an enclave (normally one per request, depending on the definition of the request).

For the duration of the processing of a request, the client task is joined to the enclave via the IWMEJOIN service so that it can be managed to the installation performance objectives. While the task is part of the enclave, it should only be doing work on behalf of the enclave. Upon completion of the request, the task leaves the enclave via the IWMELEAV service.

**Note:** While IWM4STBG and IWM4STEN are similar in many ways to IWMEJOIN and IWMELEAV, they *cannot* be used here, as queueing services are not involved.

#### 5. **Delete enclave**

When work is completed, the server address space informs workload management via the IWM4EDEL service.

#### 6. **Terminate server address space**

Under normal circumstances, it is expected that a server address space terminates only when directed to do so by workload management through the connect exit. Possible reasons for the exit being invoked could be:

- Operator entered a **VARY WLM,QUIESCE** or **VARY WLM,REFRESH** command
- Policy activated from new service definition which no longer includes the application environment associated with the server.
- WLM determines that the server is not needed to meet the goals of the enclave’s service class.

The server is expected to complete any active work requests, clean up, and then disconnect and terminate from workload management through the IWM4DIS service.

If the server encounters a failure that forces it to terminate, it should disconnect from workload management as part of its shutdown. If the server address space terminates without explicitly disconnecting from workload management, workload management detects the termination and performs the IWM4DIS processing at that time.

#### 7. **Terminate routing manager address space**

When the routing manager terminates normally, it is expected to quiesce its activities and disconnect from workload management using the IWM4DIS service. If the routing manager address space terminates without first disconnecting from workload management, workload management detects the termination and performs the IWM4DIS processing at that time. The disconnect and termination of a routing manager does **not** cause the related server address spaces to terminate. It is the responsibility of the subsystem to coordinate termination processing between the routing manager address space and the servers.

---

## Using the Routing Server Connect Exit

A routing server must provide the name of an exit routine on the `SRV_MGR_EXIT@` parameter of `IWM4CON` when it connects to workload management. The exit is invoked when workload management needs to inform the server to terminate during quiesce or refresh processing. When the exit routine is invoked, register 1 contains the address of a parameter list mapped by the list form of the `IWMSCXIT` macro instruction. The parameter list includes an indicator of the action the exit is to take and the connect token associated with the `IWM4CON` macro issued previously by the server.

The execute and standard form of `IWMSCXIT` are intended for use only by the operating system.

### Exit Routine Environment

The server connect exit routine receives control in the following environment:

**Authorization:**

Supervisor state key 0

**Dispatchable unit mode:**

SRB

**Cross memory mode:**

PASN=HASN=SASN with the current home address space the same as when the server issued the `IWM4CON` macro instruction.

**AMODE:**

31-bit

**ASC mode:**

Primary

**Interrupt status:**

Enabled for I/O and external interrupts

**Locks:**

No locks held

**Serialization:**

It is possible for the exit to be called before the caller has received control back from `IWM4CON`. The exit or any program it drives (synchronously or asynchronously) must synchronize with the program issuing `IWM4CON` to ensure that `IWM4CON` has returned a connect token prior to issuing `IWM4DIS` (disconnect) or any other service that requires the connect token.

**Location:**

The server exit must be a resident routine callable from the server address space and must remain available after the server manager disconnects or after the termination of the server task which issued the `IWM4CON`. The exit need not persist after termination of the server address space.

The input parameter list is in pageable storage addressable in the current address space, but should not be changed by the exit.

**Exit Recovery:**

The system may cause the server to become ineligible to be recommended by `IWMRFSV` (find server) if there have been repetitive errors in calling the exit. The exit may optionally establish a functional recovery routine (FRR) or `ESTAEX` for any needed recovery or cleanup of its resources.

## Register Usage

Upon entry to the exit, the register contents are as follows:

- Register 0 = not defined
- Register 1 contains the address of the input parameter list
- Register 2-13 = not defined
- Register 14 = return address
- Register 15 = entry point address

Upon entry to the exit, the access register contents are undefined.

Upon return from the exit, the register contents are expected to be:

- Register 0 = reason code if GR15 return code is non-zero
- Registers 1-14 = not defined (need not be restored to value on entry)
- Register 15 = return code

Upon return from the exit, the access register contents are unchanged.

### Example

Following is an example of invoking the IWMSCXIT macro instruction and the resulting parameter list mapping. This is the parameter passed to the routing server exit routine specified on the SRV\_MGR\_EXIT@ parameter of IWM4CON.

```
IWMSCXIT MF=(L,MYSCXITPL)

+MYSCXITPL DS 0D                ++ IWMSCXIT PARAMETER LIST
+MYSCXITPL_XVERSION DS XL1      ++ INPUT
+MYSCXITPL_XRSV0001 DS CL1      ++ RESERVED
+MYSCXITPL_XPLISTLEN DS XL2     ++ INPUT
+MYSCXITPL_XACTION DS BL.8      ++ ACTION INDICATORS
+MYSCXITPL_XACTION QUIESCE EQU B'10000000'
+                                ++ INDICATES QUIESCE ACTION
+MYSCXITPL_XACTION_RESUME EQU B'01000000'
+                                ++ RESUME (NOT USED IN OS/390 R3)
+MYSCXITPL_XRSV0005 DS CL3      ++ RESERVED
+MYSCXITPL_XCONNTKN DS BL.32    ++ CONNECT TOKEN FROM IWM4CON
+MYSCXITPL_XRSV000C DS CL4      ++ RESERVED
+MYSCXITPL EQU *-MYSCXITPL     ++ LENGTH OF PARAMETER LIST
```



---

## Chapter 7. Using the Scheduling Environment Services

A scheduling environment is a list of resource requirements and their required states. It allows you to manage the scheduling of work in an asymmetric sysplex where the systems differ in installed applications or installed hardware facilities. If an MVS image satisfies all of the requirements in the scheduling environment associated with a given unit of work, then that unit of work can be assigned to that MVS image. If any of the resource requirements are *not* satisfied, then that unit of work *cannot* be assigned to that MVS image. See *z/OS MVS Planning: Workload Management* for information on using the WLM application to define scheduling environments.

---

### Obtaining Scheduling Environment Definitions

Use the **IWMSEQRY** service to obtain scheduling environment and resource definitions and status on each system in the sysplex. This service can be used by a program to produce alternative displays to those provided by the **DISPLAY WLM** command.

In Figure 16 on page 80, IWMSEQRY returns the resource requirements in the DB2LATE scheduling environment (DB2A must be ON, PRIMETIME must be OFF), and the resource state settings on each of the four systems in the sysplex. Because DB2A is set to ON and PRIMETIME is set to OFF on SYS2, that system is identified as the appropriate system for work associated with the DB2LATE scheduling environment.

Scheduling environments,  
according to service definition

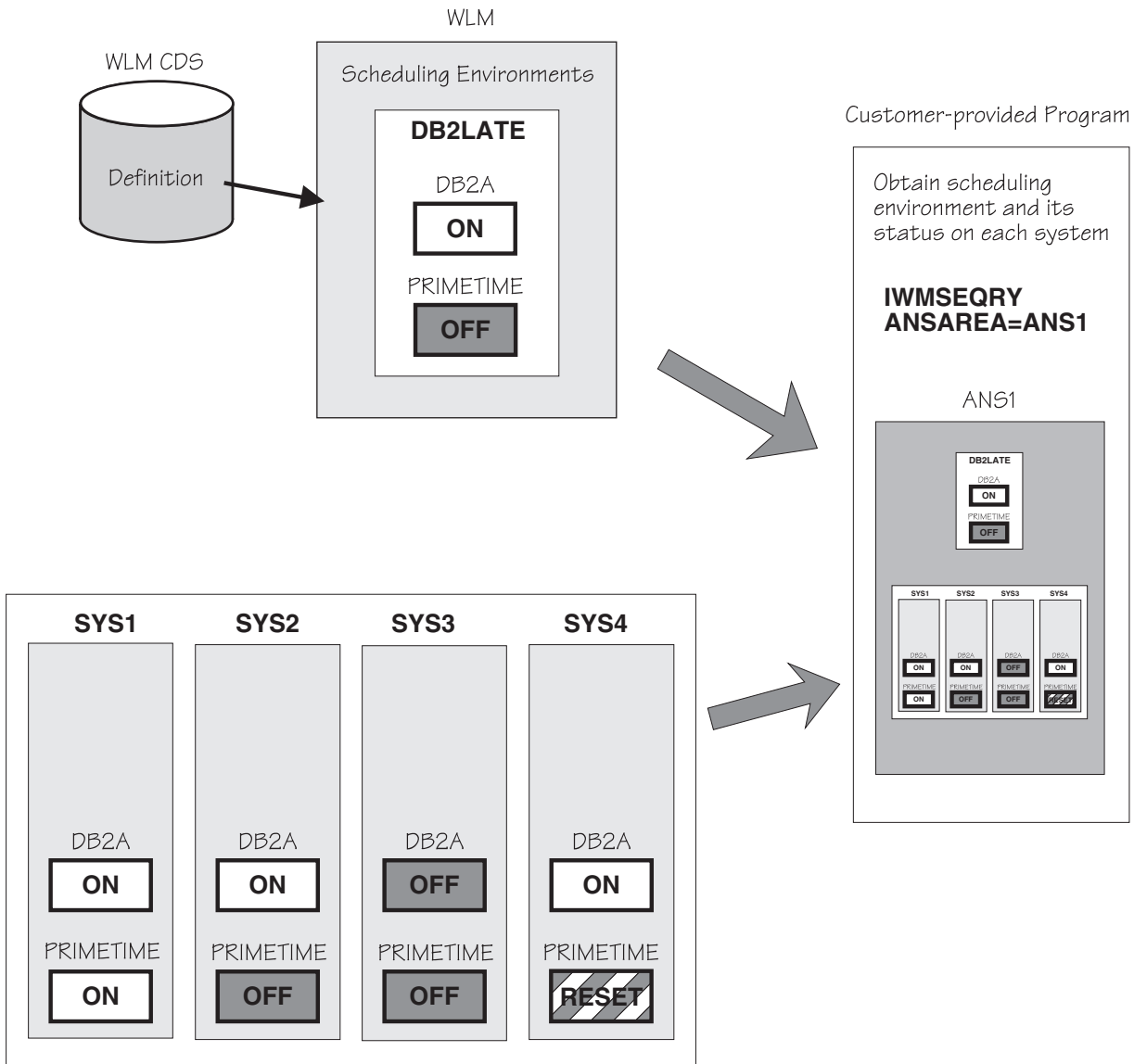


Figure 16. Obtaining Scheduling Environments

## Manipulating Resource State Settings

Use the **IWMSESET** service to change resource state settings. The resource state can be changed only on the system in which the program is executing.

The resource states can be set to:

- **ON**, which will satisfy a resource state requirement of ON.
- **OFF**, which will satisfy a resource state requirement of OFF.
- **RESET**, which will not satisfy *any* resource state requirement.

In Figure 17, a program executing on SYS3 invokes IWMSESET to change the value of DB2A from OFF to ON. SYS3 now satisfies the DB2LATE resource requirements shown in Figure 16 on page 80 (DB2A ON and PRIMETIME OFF).

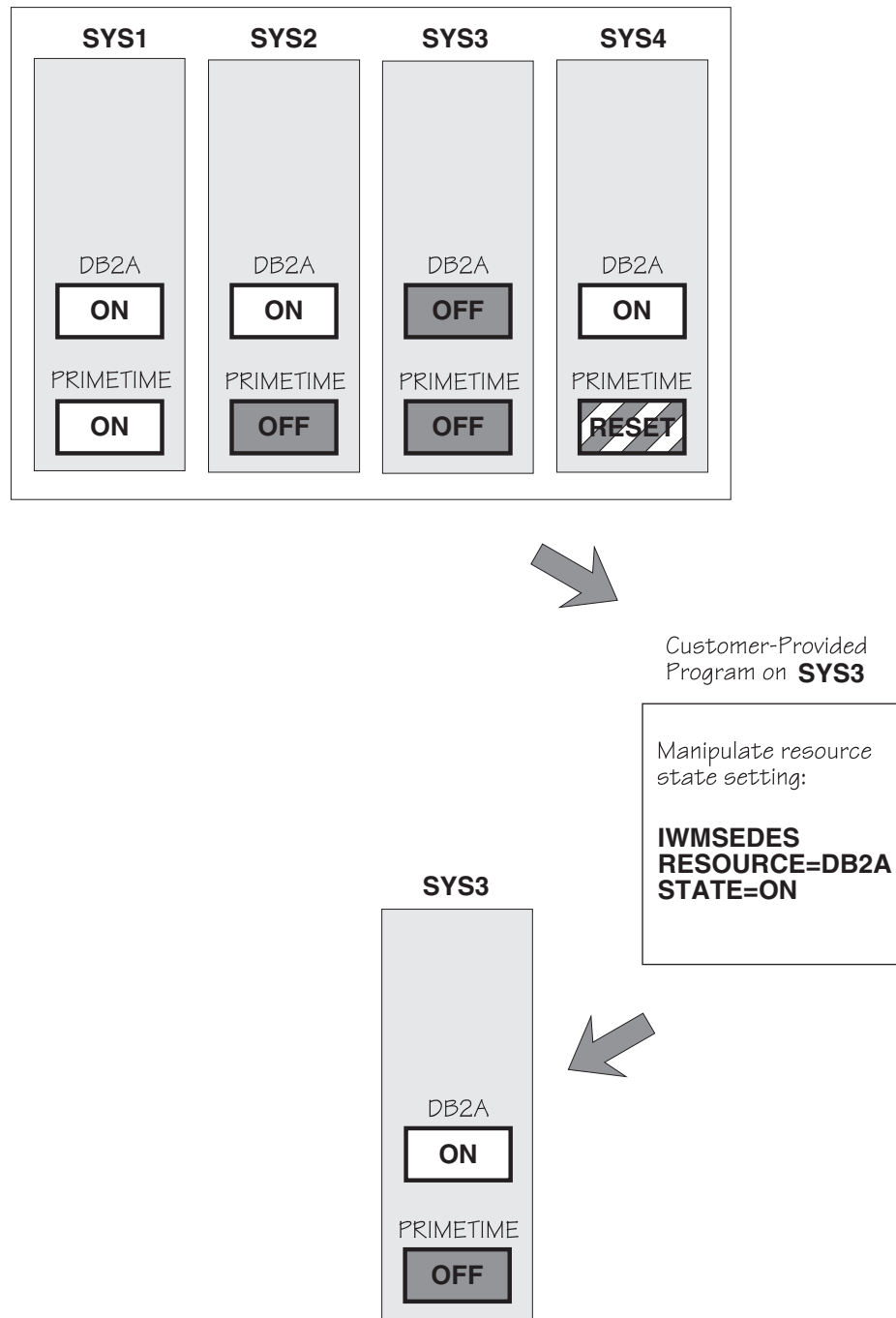


Figure 17. Manipulating Resource State Settings

## A Model Work Flow

Figure 18 shows how a multisystem work scheduler can use the IWMSEVAL and IWMSEDES services to implement support of scheduling environments.

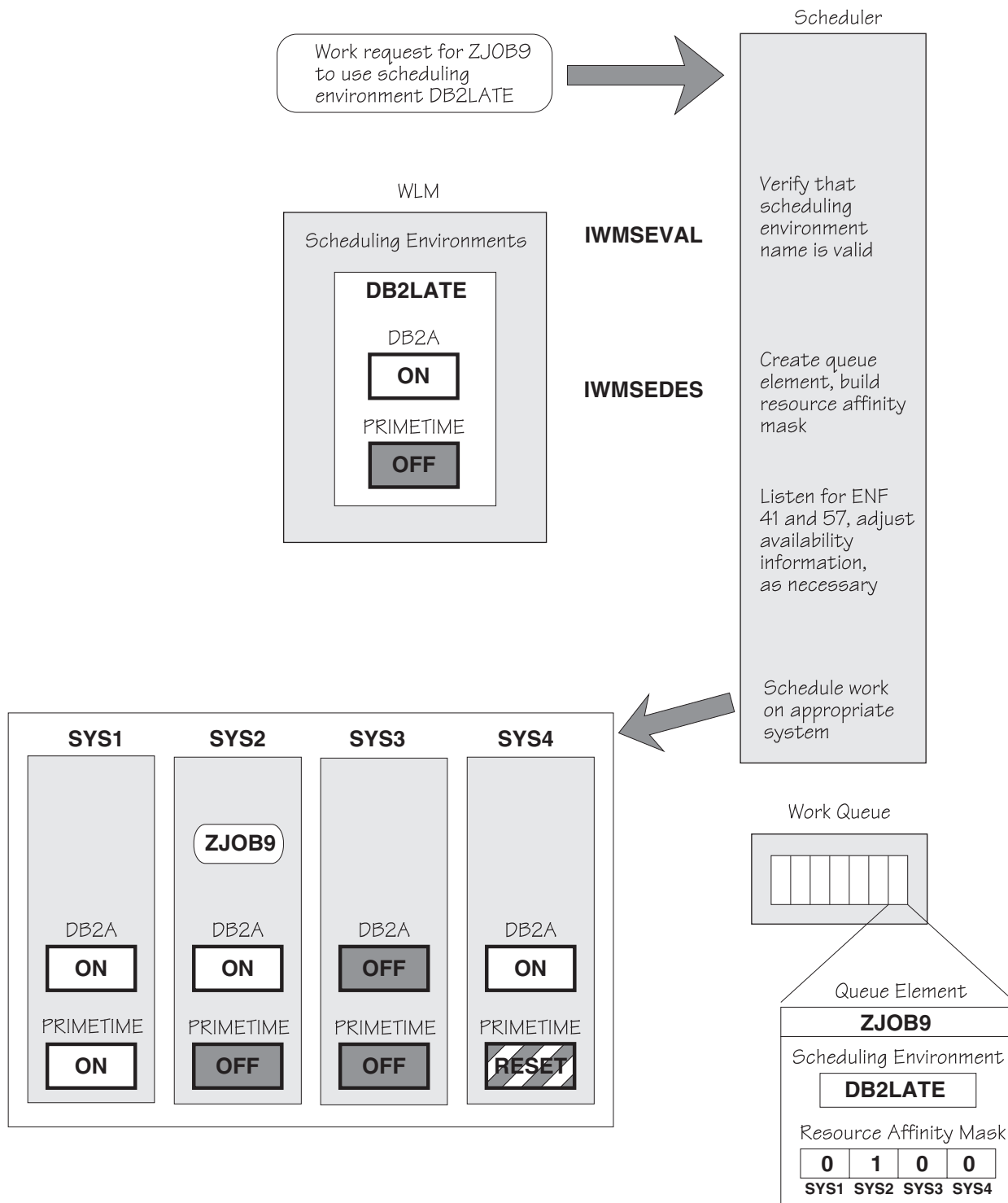


Figure 18. Scheduling Environment Flow

In Figure 18 on page 82, work request ZJOB9 is submitted and associated with the DB2LATE scheduling environment.

The scheduler calls the **IWMSEVAL** service to verify that the scheduling environment name is valid. If there is no such scheduling environment known to workload management, then the scheduler fails the work request. If the scheduling environment name is valid, then the scheduler accepts the work request.

The scheduler creates a queue element for the work request. The scheduling environment name is included in that queue element, as well as a resource affinity mask. For each system in the sysplex, IWMSEDES will indicate whether that system satisfies the scheduling environment in question. The scheduler can then build the resource affinity mask, which is simply a 32-bit string of ones and zeros. A one means that the system satisfies the scheduling environment, and a zero means that the system does *not* satisfy the scheduling environment. The scheduler must keep an ordered list of system names corresponding to the bit positions in the mask.

In Figure 18 on page 82, the resource affinity mask for DB2LATE reads “0100”, because only SYS2 satisfies the DB2LATE scheduling environment.

Scheduling environment definitions and resource state settings can change at any moment. The scheduler must be aware of these changes so that it can adjust its decision-making accordingly. The scheduler listens for two ENF events, 41 and 57, which signal these changes:

- 41 A new service definition has been activated.** When a new service definition is activated, resource names can be added to, or removed from, a scheduling environment. A particular scheduling environment or resource name may even have been deleted from the service definition altogether. (See note below.)
- 57 A scheduling environment has changed state on a system.** It was not available and now is, or vice versa.

For more information about ENF, see *z/OS MVS Programming: Authorized Assembler Services Guide*.

For either event, the scheduler must reevaluate the status of each work request on the queue. It must rebuild the resource affinity masks to reflect the new scheduling environment definitions or the new resource state settings.

If a scheduling environment no longer exists, the scheduler can either delete the associated work requests or place them in a hold state for installation action. The latter choice is appropriate if it is important for the installation to be able to recover the work requests. The installation could recover by installing a new service definition that includes the deleted scheduling environment.

The final step in this work flow is when the work request moves from the queue to the appropriate system for execution. In Figure 18 on page 82, the ZJOB9 work request is scheduled on SYS2, as that is the only system that satisfies the DB2LATE scheduling environment.



---

## Chapter 8. Using the Sysplex Routing Services

The sysplex routing services allow work associated with a server to be distributed across a sysplex. They are intended for use by clients and servers.

A client is any application or product in the network that requests a service. The service could be a request for data, a program to be run, or access to a database or application. In terms of the sysplex routing services, a client is any program routing work to a server. A server is any subsystem address space that provides a service on an MVS image.

The sysplex routing services provide the following functions:

- The **IWMSRSRG** macro lets a caller register as a server.
- The **IWMSRSRS** macro provides a caller with a list of registered servers and the number of requests that should be routed to each server.
- The **IWMSRDRS** macro lets a caller deregister when it is no longer available as a server.
- The **IWMSRDNS** macro provides the caller with list of location names for all registered servers known to the system on which the service is invoked.
- The **IWM4SRSC** macro provides the caller with server-specific information about the performance and capacity of a specific server before work gets routed to it from WLM. This information can be used for balanced routing decisions.

---

### Why Use the Sysplex Routing Services?

The sysplex routing services enable distributed client/server environments to balance work among multiple servers. These services help the distributed programs make the routing decisions, rather than having each installation have to make the decisions.

The sysplex routing services provide information for more intelligent routing. They do not route or distribute work requests. The server must use its existing routing mechanisms.

More than half of the systems running in the sysplex should be running in goal mode. Workload balancing can be more intelligent if all servers are located on systems running in goal mode.

---

### When to Use the Sysplex Routing Services

When a server is available to receive work requests, it issues the **IWMSRSRG** macro to register. You should issue **IWMSRSRG** once per initialization of a server. The system makes this information available asynchronously to all MVS images in the sysplex.

When a client wants to route work to a server, it issues the **IWMSRSRS** macro for a list of registered servers. To help the client decide where to route the request, **IWMSRSRS** returns the relative number of work requests that can be routed to each server in the list.

When a server is no longer available to receive work requests, it issues the **IWMSRDRS** macro to deregister. A server should issue **IWMSRDRS** once per address space termination.

The following section explains each of the services in more detail.

## Registering as an Eligible Server

A server can use IWMSRSRG to register that it is available for receiving work requests. A server should issue IWMSRSRG at address space initialization time. The server identifies itself by a triplet consisting of the following:

- Location name
- Network id
- LU name

Servers with the same location name are considered to be related and to provide equal service. The server must also specify an address space token (STOKEN) identifying the server address space token. Since the propagation of registration to all systems in the sysplex is asynchronous, a newly registered server is not immediately available for selection on other systems.

For complete information about the IWMSRSRG macro, see Chapter 56, “IWMSRSRG – Register a Server for Sysplex Routing,” on page 465.

## Determining Where to Route Work

The IWMSRSRS macro returns a list of eligible servers, and for each server, an associated weight. From this list, a client can determine where to route work. See *z/OS MVS Data Areas, Vol 3 (IVT-RCWK)* for the mapping.

The weights provided allow changes in system conditions and server availability to be factored into the distribution of work. The weights represent the relative number of requests each server should receive. Various capacity considerations are used to calculate the weights.

A client can route the work to the servers as ordered in the list, and can route the number of requests as suggested for each server. A caller should be aware of other clients in the sysplex issuing the IWMSRSRS macro, and route work properly.

For example, if the list consists of the following:

Server	Weight
NETID1.LUNAMEA	3
NETID2.LUNAMEB	8
NETID3.LUNAMEC	2

then the client should route the first 3 requests to NETID1.LUNAMEA, the next 8 to NETID2.LUNAMEB, and so on. When the client has gone through the list, the client can either invoke IWMSRSRS again for a refreshed list, or rotate through the list again.

Acceptable weights are from 1 to 64. If a server is busy, and not available to receive any work requests, it will not appear in the list. If work is routed to the server and it is not able to process the request, the request is not processed.

Clients should issue this macro on a regular basis, so that the list is refreshed every one to three minutes. This way, a client can stay current with changing system loads and server viability.

For complete information about the IWMSRSRS macro, see Chapter 57, “IWMSRSRS – Determine Where to Route Work,” on page 473.



## Deregistering as an Eligible Server

When a server is no longer available for processing work, the server should issue the IWMSRDRS macro to deregister. A server should issue IWMSRDRS at address space termination. This makes the system aware that the server is no longer a candidate for future work requests, and can remove it from the list of eligible servers.

For complete information about the IWMSRSRS macro, see Chapter 54, “IWMSRDRS – Deregister a Server for Sysplex Routing,” on page 451.

---

## Example of Using the Sysplex Routing Services

Figure 19 shows an example of the sysplex routing services.

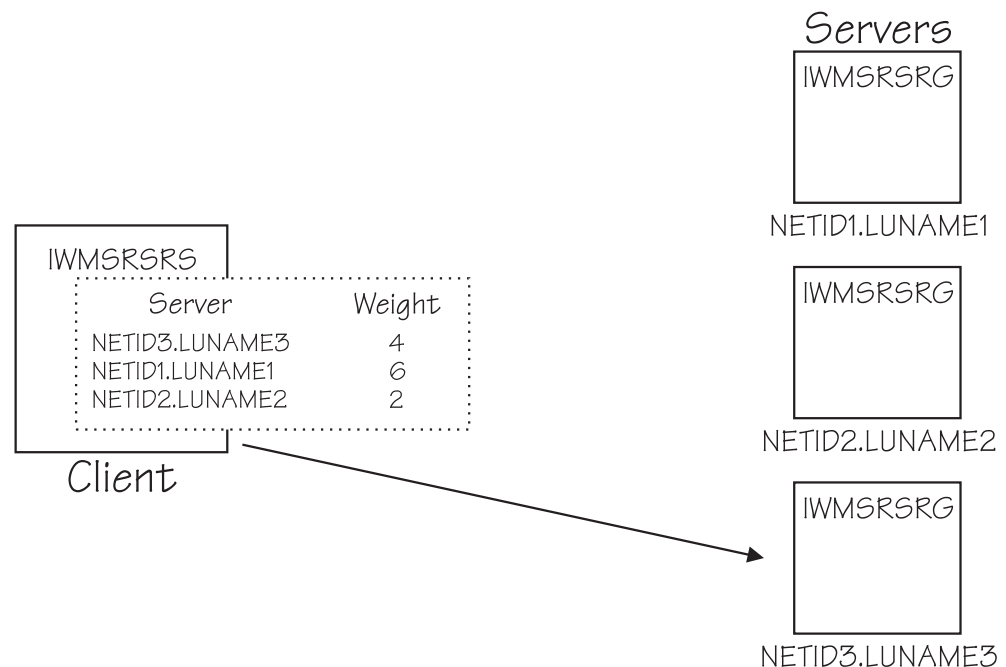


Figure 19. Example of using sysplex routing services

In the figure, there are three servers, and one client program. Each of the servers registers at address space initialization using the IWMSRSRG macro. Suppose the first server registered, with the following defined:

LOCATION1	DS	CL18	location
NETWORKID1	DS	CL8	network id
LUNAME1	DS	CL8	LU name
STOKEN1	DS	CL8	token of server address space

The first server uses the IWMSRSRG macro as follows:

```
IWMSRSRG LOCATION=location1,  
          NETWORK_ID=networkid1,  
          LUNAME=Luname1,  
          STOKEN=stoken1
```

Meanwhile, the client program wants the list of address spaces available for processing work. The client issues IWMSRSRS once for the length of the area required for the list. The length required is in the QUERYLEN output parameter. The client issues:

```
IWMSRSRS SYSINFO_BLOCK=sysinfo_block,  
          ANSLEN=anslen,  
          QUERYLEN=querylen,  
          LOCATION=location
```

**Note:** The QUERYLEN value is a snapshot taken when IWMSRSRS is issued. That value can change between calls due to other servers registering. To compensate, it is recommended that you add a few entries extra length to the size resulting from the first IWMSRSRS call.

The client then obtains the required amount of storage, and issues IWMSRSRS again for the list of eligible servers and their weights:

```
IWMSRSRS SYSINFO_BLOCK=sysinfo_block,  
          ANSLEN=anslen,  
          QUERYLEN=querylen,  
          LOCATION=location
```

The caller receives the list, mapped by IWMWSYSR, as follows:

Server	Weight
-----	-----
NETID3.LUNAME3	4
NETID1.LUNAME1	6
NETID2.LUNAME2	2

The client routes the first 4 work requests it receives to NETID3.LUNAME3, the first in the list. The client routes the next 6 requests to NETID1.LUNAME1, and then the following 2 to NETID2.LUNAME2. The client opted to issue the IWMSRSRS macro every three minutes. Once the client has received 12 requests, it will route the 13th through the 16th requests to NETID3.LUNAME3, and go down the list again until the three minutes are over.

---

## WLM Sysplex Workload Distribution

When WLM is called by DNS or sysplex distributor for advice on where to route work in a sysplex, WLM returns a list of eligible servers with a weight assigned to each server. Work is then routed to the server proportionally to its weight so the higher weight servers receive more work. For example, if four servers A, B, C, and D are returned by WLM with weights of 2, 2, 4, and 8 respectively, DNS will route one-eighth of the requests to each of servers A and B, one-fourth of the requests to server C, and one-half the requests to server D.

The weights are calculated as follows. WLM first assigns a weight to each system, with a higher weight to systems with the most available capacity. The weights are proportional to the amount of available capacity on each system. Then WLM divides the weight for a system equally among the registered servers on that system. In the case of DNS, there can be multiple servers registered with WLM per system image. For sysplex distributor, however, the WLM server registration is done by the TCP/IP stack so there is only one server per system image. This means the server weight always equals the system weight for sysplex distributor. If all systems are running at or near 100% utilization, then higher weights are given to the systems running less important work. The objective is to send work where there is available capacity or, if there is no available capacity, where the least important work will be displaced. Servers on systems that are in a serious storage shortage (SQA, fixed frame, auxiliary storage) are not recommended unless all systems are in a shortage.

## Calculation of server weights

The basis for WLM's weight calculation is the Table 17 for each goal mode system in the sysplex. More than half of the systems in the sysplex need to be in goal mode for WLM to calculate weights. Otherwise, all weights are set equal to one. The table has one row for each level of work in the system:

- Level 0 is used for service consumed by the system
- Levels 1-5 are the externally defined importance levels
- Level 6 is used for Discretionary work
- Level 7 is unused service (equivalent number of service units)

Each row contains the following fields:

- Number of service units (SUs) consumed in 3 minutes at the given importance level and below.
- Percentage of total capacity consumed in 3 minutes at given importance level and below.

For example, Level 4 in the table has the service units consumed by importance 4, importance 5, and discretionary work plus unused capacity.)

1. WLM scans the rows in the table in reverse starting with Level 7 until it finds a level where one or more systems have at least 5% cumulative service units of capacity. WLM then uses this table level for all goal mode systems to assign relative weights.
2. A system weight is calculated for each system in the sysplex using the SU value in the selected row in the table.

$$\text{System weight} = \frac{\text{SUs for this system at selected level} * 64}{\text{total SUs for all systems at selected level}}$$

3. Finally, a server weight is calculated:

$$\text{Server weight} = \frac{\text{system weight}}{\text{\# of servers on system}}$$

## Example

In this example, there are three goal-mode systems in the sysplex, all running at 100% CPU utilization. Each system has two DNS servers running for a particular application. To assign weights to the three systems, WLM scans the CPU consumption table from the bottom up, looking for a level where at least one system has 5% or greater cumulative CPU consumption. In this case, Level 5 row is used because System A exceeds the 5% minimum (and so does System B). Level 5 is then used to calculate the weight for all systems including System C. So in this case the system weights are determined by the amount of importance 5 and discretionary work each system is running. Because there are two servers on each system, the server weight is the system weight divided by 2.

System A weight =  $200 * 64 / 640 = 20$   
 System B weight =  $400 * 64 / 640 = 40$   
 System C weight =  $40 * 64 / 640 = 4$

Server weight =  $20 / 2 = 10$   
 Server weight =  $40 / 2 = 20$   
 Server weight =  $4 / 2 = 2$

Table 17. CPU consumption table

Level	System A		System B		System C	
	SUs	%	SUs	%	SUs	%
0	2000	100	2000	100	2000	100

Table 17. CPU consumption table (continued)

	System A		System B		System C	
1	1800	90	1900	95	1840	92
2	1600	80	1500	75	1600	80
3	1100	55	1500	75	800	40
4	400	20	1200	60	800	40
5	200	10	400	20	40	2
6	80	4	20	1	0	0
7	0	0	0	0	0	0

Since z/OS V1R7 the IWMSRSRS service supports the new function code SPECIFIC, that allows for calculated weights to also take the server-specific information into account. If the IWMSRSRS service is called with FUNCTION=SPECIFIC, then the weight calculated as described above, is additionally multiplied by a *Performance Index (PI)* factor and a *queue time* factor. The value of these two factors is calculated as follows:

**The PI factor**

is 1 if the server has a PI  $\leq 1$  which means that it achieves its goals. The PI factor will become  $< 1$ , if the PI indicates, that the server does not achieve its goal.

**The queue time factor**

becomes  $< 1$  if the server is the owner of enclaves and those enclaves have reported queue times relative to their total elapsed times. This means that their arrival time differs from the enclaves create time, when the enclave was created.

| Since z/OS V1R8 , or z/OS V1R7 with OA14310 installed, a third factor is multiplied  
 | into the weight calculation, the *health factor*. The health factor is set for the server  
 | address space either by the registration service IWMSRSRG or by the health  
 | service IWM4HLTH. For further information about these services, refer to  
 | Chapter 56, "IWMSRSRG – Register a Server for Sysplex Routing," on page 465  
 | and to Chapter 69, "IWM4HLTH – Set Server Health Indicator," on page 581.

---

## Chapter 9. Using the Workload Reporting Services

The workload reporting services are intended for monitoring or reporting products to collect performance data. They replace some of the existing methods of collecting data, and provide a means to collect data for the goal-oriented processing with the service policy.

---

### When to Use the Workload Reporting Services

Because the data collection is cumulative, performance monitors can collect information based on their own reporting intervals. But the collection is stopped and re-started when a significant change occurs in workload management, such as when a new policy is activated, or a system failure occurs. Performance monitors should “bookend” their intervals when such a significant change occurs in workload management. For each of these events, ENF signals are issued. SRM samples the states of address spaces, and an ENF signal is issued when a new set of samples is available. The performance monitor can use the ENF signals to guide its reporting interval. For example, when an ENF code for a new policy activation is issued, the performance monitor can end its last reporting interval, and start the next reporting interval for the newly activated service policy.

To enable the performance monitor to know when a workload reporting change is taking place, such as when a policy is activated, there is a ENF system event code. ENF event code 41 and its related qualifiers indicate when changes are taking place related to service policies. ENF code 41 guides the performance monitor’s reporting intervals, and helps it to issue the services at the appropriate times.

---

### Using ENF Signals to Guide Data Collection

Since all performance data is continually collected until a significant change takes place in workload management, the performance monitor must know when the collection is re-set. Workload management re-sets its collecting when:

- A service policy is activated (VARY WLM,POLICY=)
- A system error occurs.

ENF signals are issued for these events, and for each ENF, a listener is notified synchronously at the start of a change, and at the completion of a change. There is an ENF code qualifier indicating a change started, an ENF code qualifier indicating a change completed, and an ENF code qualifier indicating a change has failed.

When a change is started, workload management captures a last copy of workload data. The performance monitor can listen for the “start” event code, and then invoke IWMRCOLL to get the last available data. The workload data remains the same until either the change completes, or the change fails. Based on the “start” event code, the performance monitor should save this data, end its current interval, and wait for the next code.

**Note:** Since the ENF signal is issued synchronously, the listener should complete processing as quickly as possible. For example, if the ENF is issued broadcasting the start of policy activation, the policy activation is held up until all listeners for that ENF have relinquished control.

If the change is completed, the performance monitor should invoke the proper services, as shown in Table 18 on page 92. A performance monitor should always

complete its last interval with the data collected when it received the “start” event code. If the change failed, the performance monitor can continue to use the data it saved when it received the start event code. Regardless of whether the event has completed or failed, the performance monitor should reinitialize all of its workload activity information. Then the performance monitor should issue:

- IWMPQRY for a copy of the active policy
- IWMRCOLL for workload activity information.

This way, the performance monitor does not need to do two different things for completions and failures.

There is also an asynchronous ENF signal issued when the IWMWRQAA information is available. That way, a performance monitor can synchronize its sampling interval for address space states with workload management’s sampling interval.

Table 18 on page 92 shows the ENF event code and its qualifiers. The table also outlines what a performance monitor could do when the ENF code is heard.

## ENF Event Code 41

Table 18. Using ENF event code 41 to guide data collection for policy changes

Event	Signal and qualifiers	Expected action
VARY WLM, POLICY=	WLMENF11, WLMENF12, WLMENF13	<p><b>WLMENF11</b> Indicates the policy change has begun. The performance monitor should invoke IWMRCOLL to obtain a copy of the last available data, and wait for WLMENF12. The performance monitor should save the copy as the last available for the interval.</p> <p><b>WLMENF12</b> Indicates the policy was successfully activated on this system. To get a copy of the new policy information, the performance monitor should issue the IWMPQRY macro. The performance monitor should then reinitialize its workload activity reporting fields and resume data collection.</p> <p><b>WLMENF13</b> Indicates the new policy was <b>not</b> activated on this system, but may have been activated on some systems in the sysplex. This system is potentially running with a different policy than the rest of the systems in the sysplex. The performance monitor should acquire the current policy information by issuing the IWMPQRY macro. The performance monitor should reinitialize its data collection fields and indicators and resume data collection with subsequent IWMRCOLLs. The performance monitor could alert the customer that this system is not running with the active service policy.</p>
System failure	WLMENF31, WLMENF32, WLMENF33	<p><b>WLMENF31</b> Indicates workload activity reporting failed, recovery has begun. Do not issue IWMRCOLL until recovery is complete.</p> <p><b>WLMENF32</b> Indicates workload activity reporting recovery was successful. Start a new reporting interval.</p> <p><b>WLMENF33</b> Indicates workload activity reporting recovery was unsuccessful. Do not issue IWMRCOLL as no data is returned until the next IPL.</p>

---

## Using the IWMRCOLL Service

The IWMRCOLL service allows a performance monitor to obtain a variety of performance data from a single system. Although IWMRCOLL provides information about a single system, the performance monitor can combine the information to provide a sysplex view.

Table 19 shows how you can use the workload reporting services together with IWMRCOLL to get workload activity information on a single system. The following section explains how to use the information returned by IWMRCOLL.

*Table 19. Using IWMRCOLL with the workload reporting services on a single system*

Action	Reason
Issue REQSRMST SYSEVENT	To get information about the service definition and policy.
Issue IWMPQRY macro	To obtain current active policy length.
Issue GETMAIN	To obtain storage needed to hold the active service policy information
Issue IWMPQRY	To obtain active service policy information
Set up a reporting interval.	To prepare for subsequent IWMRCOLL requests.
Issue IWMRCOLL specifying MINLEN, MAXLEN, and QUERYLEN	To obtain length of storage needed. IWMRCOLL returns ANSTOKN required for subsequent calls to IWMRCOLL.
Issue GETMAIN	To get storage needed to hold information returned by IWMRCOLL.
Issue IWMRCOLL ANSTOKN=token	To get workload activity information mapped by IWMWRCAA.
Set up a reporting interval	Sample workload activity information by issuing IWMRCOLL at your determined interval.
Calculate differences in data	To determine the delta of data returned for the interval in each successive IWMWRCAA.

## Using the Information in IWMWRCAA

By using IWMRCOLL, a performance monitor can get the following kinds of information:

- Resource consumption information
- Response time and distribution information
- General delay information
- Subsystem work manager delay state information

The information provided is mapped by the IWMWRCAA data area. Information is returned on a service class and report class basis. The following section explains how to use the response time data and the subsystem work manager delay state information.

- **Header Data**

The header data present in the RCAA consists of:

- Data specific to the workload management mode in effect, such as which service policy is active.
- General information such as bookkeeping information.

- **Resource Data**

Resource information is available for address spaces and there is a distinction between transactions and server address spaces. A server address space is any address space that does work on behalf of a transaction manager or a resource manager. For example, a server address space could be a CICS AOR, or an IMS control region. Service classes that represent CICS or IMS transactions, as

opposed to address spaces, have no resource data. The resources being consumed by such transactions are reported in the service class of the server address space.

- **Delay Data**

There are two types of delay information returned by IWMRCOLL: general execution delays, and subsystem work manager delays. All data is sampled.

The general execution delays are address space oriented while the subsystem work manager execution delays are service class oriented. They show delays encountered by service classes representing transactions. The subsystem work manager execution delays are in the *subsystem work manager state samples* section of IWMWRCOA. The state samples are available only for subsystem work managers using the execution delay monitoring services.

In order to determine whether a service class has execution delay state sampling information, the header data indicates which type of delay information is available for a service class. You can use address space delay data to calculate execution velocity.

- **Response Time Data**

There is response time information and response time distributions for transactions. Both are reported for service classes and report classes.

You can calculate average response times using the information provided in this section. Total completed transactions (both normal and abnormal) are provided, as well as total transaction completion time. This same information is available for the execution phase of transactions.

## Using the Subsystem Work Manager Delay State Information

The delay information in IWMWRCOA represent delays encountered by subsystem work managers as they process transactions. Workload management can recognize those address spaces that process transactions on behalf of the transaction managers. Such address spaces are called *server address spaces*. Workload management manages the server address spaces to achieve the goals of the transactions the server is processing. If a server address space is managing more than one service class, workload management manages the address space to meet the most stringent service class goal. However, resource consumption and address space delays for server address spaces are reported in the server's service classes.

The delay information shows the different states server address spaces experience while processing transactions. The information is provided on a service and report class basis--in the service or report class representing the transactions. This way, the delay states show for the transactions being processed, not for the address spaces serving the transactions. The states include how many times the service or report class was seen active, ready, and waiting. There are several waiting states. Each of these is reported separately. Some other states include whether the transactions are continued somewhere else in the system, in the sysplex, or in the network.

### Using the Continued State Information

The state information helps a performance monitor show a picture of how well transactions are being processed. Because multiple address spaces can be involved in processing a transaction, a delay could occur in any of several places. IWMWRCOA provides state information to help a performance monitor pin down when transactions experience delays. The states show whether a service class has continued:

- In the local system



- In the sysplex
- In the network

With the cooperation of the participating subsystem work managers, the information reported divides the lifespan of transactions into two phases: a begin-to-end phase, and an execution phase. A begin-to-end phase shows transaction states from the time the subsystem work manager receives a transaction, processes it, and ends it. An execution phase shows transaction states only during the time a subsystem work manager processes the transaction. Delay states may not always appear in each phase. It depends on how a subsystem work manager is reporting the delays it encounters while processing.

For example, for CICS transactions, delay states are recorded from the time the TOR receives the transaction and begins processing, through the time the transaction is processed in the AOR, FOR, or elsewhere, and ended back in the TOR. The phase where the TOR receives the transaction, and ends it is called the begin-to-end phase. The phase where the transaction moves into the AOR, FOR, or elsewhere and is processed is called the execution phase.

The delay states for both the begin-to-end and the execution phase are reported together in the service class of the CICS transactions processed. For example, Figure 20 shows the delay states sampled for both begin-to-end phase and the execution phase for one CICS service class representing CICS transactions.

If the execution phase occurs on the same MVS image as the begin-to-end phase, then (barring some statistical anomalies), the amount of time the service class spends in the *continued - LOCAL* state of the begin-to-end phase should approximately equal the amount of time the service class spends in the execution phase.

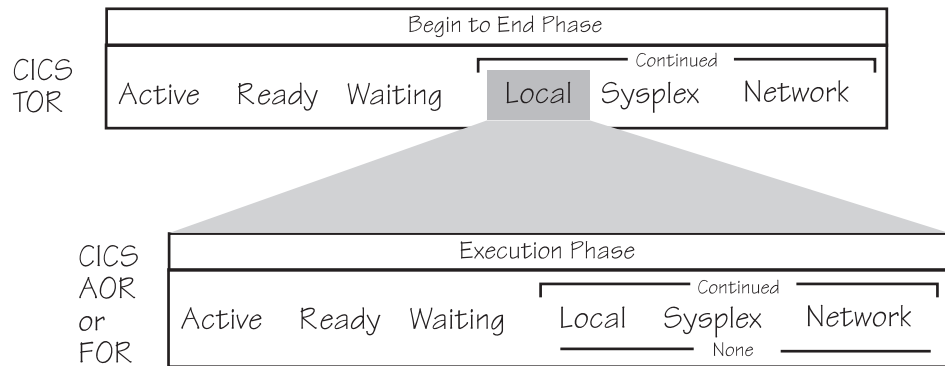


Figure 20. Using states for presenting CICS delay information

The execution phase could be in the same system as the begin-to-end phase, but could also be on another system in the sysplex, or in the network.

By combining the information collected on each system for a given service class, the performance monitor can resolve the states where the transactions in the service class continued elsewhere in the sysplex. For transactions continued elsewhere in the network, workload management cannot know any more information. Only the count of how many were switched out through the network is provided.

Figure 21 shows how the performance monitor could view the state information to provide a sysplex view. In this example, the states are reported for the CICSFAST service class. In this example, the number of samples of transactions continued somewhere in the sysplex should equal the number of transaction states sampled in the sysplex (summed from each system in the sysplex). The performance monitor can combine all the state data from all the systems in the sysplex to provide a sysplex view, correlating the data by service class.

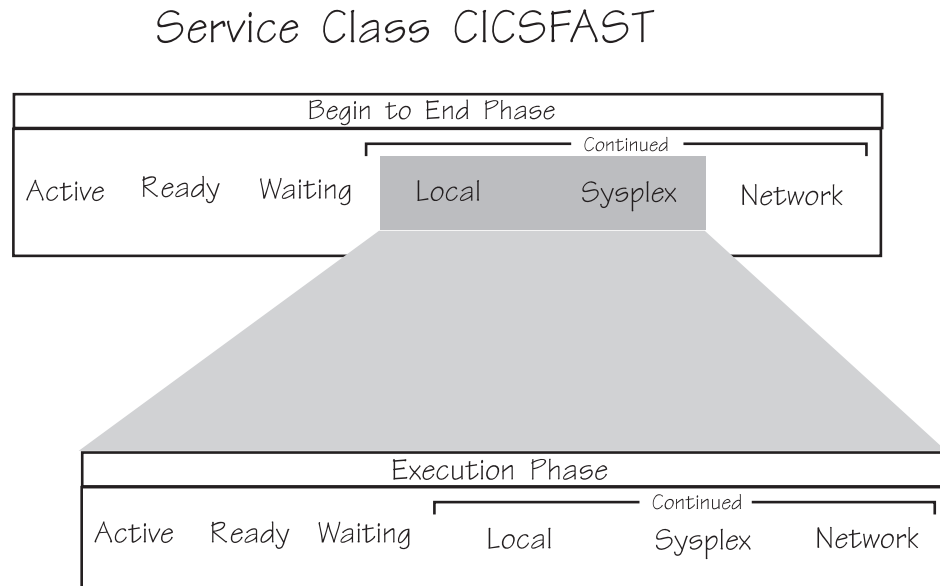


Figure 21. Combining state information for a sysplex view.

## Using Delay States to Report Subsystem Interactions

Not only are multiple address spaces involved in processing transactions, but those address spaces may be part of different subsystems. For example, a CICS TOR may give control to a CICS AOR who in turn may do a query to IMS DLI. Workload management can keep track of subsystems that communicate with each other, and provides the information so that a performance monitor can present the subsystem interactions in processing transactions.

The CICS transaction used in the previous example has a begin-to-end phase and an execution phase. The execution phase could be split among several subsystems, and the delays associated with each distinct subsystem are reported separately. The performance monitor should combine all information by service class by subsystem to provide a sysplex view.

Delay data are presented for as many distinct subsystems as participated in processing each service class. The data are available for both the begin-to-end phase and the execution phase.

A performance monitor could provide a timeline with the various pieces and phases of the work represented differently. In Figure 22 on page 97, the performance monitor sees that the biggest delays encountered are in the **Waiting** state and the **Continued Local** state. The performance monitor could show further information

about the **Continued Local** states by presenting the information found in the execution phase on that system.

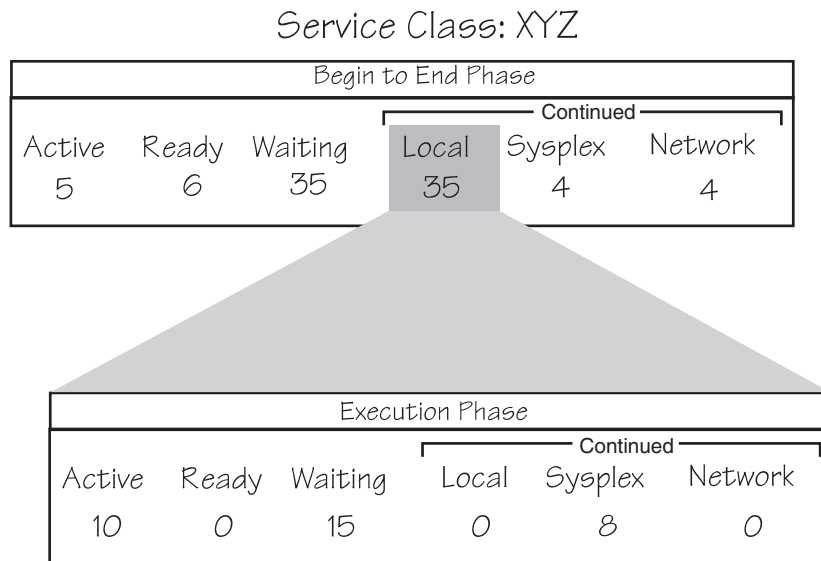


Figure 22. Combining state information for a service class.

In the example, the performance monitor shows that there are execution delays attributed to **Active**, **Waiting**, and **Continued** states. Notice that of the 35 delay states reported in the original display as being **Continued Local**, only 33 of them show up in the execution phase. This is one of the shortcomings of sampling.

In Figure 23 on page 98, there are two subsystems represented in the execution phase. This means that during this interval, both subsystem A and subsystem B performed work on behalf of service class xyz. Again, notice that there is 1 state sample missing due to statistical anomalies. The performance monitor could determine the specific reason for subsystem B **Waiting** states from the delay states.

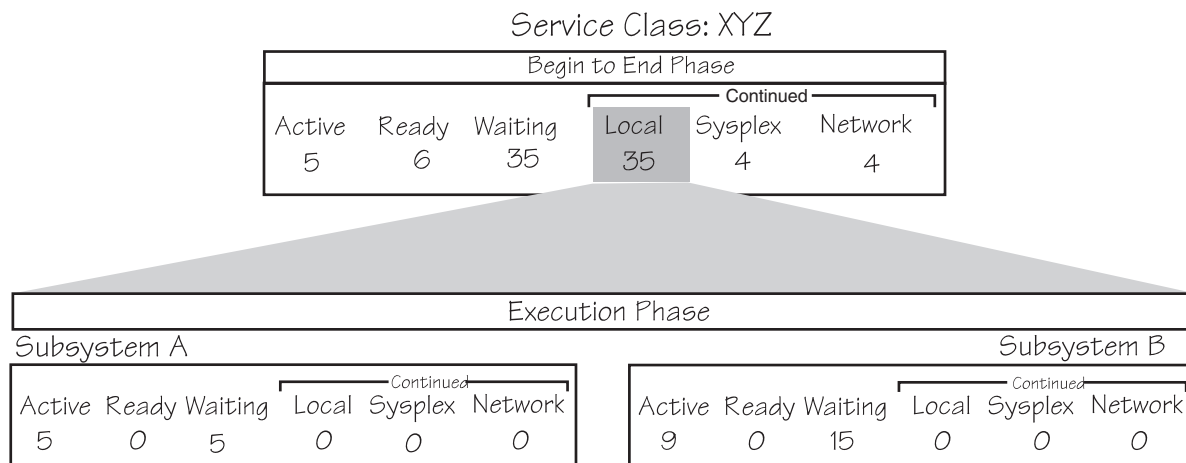


Figure 23. Combining state information across subsystems.

## Using the Response Time Information

To provide a picture of how a performance group was performing, SRM previously reported total transaction time for use in calculating standard deviation. This information is provided for transaction execution time. Response time distributions are provided for both service classes and report classes.

These distributions consist of 14 buckets of information. There is a header explaining the contents of the buckets that is provided once. The header contains the value of the particular bucket, which is a percentage of the specified goal (that is, 50 equates to 50% of the goal; 150 equates to 150%, of the goal) There is always one bucket which exactly maps to the specified goal, with a value of 100%.

In each bucket is the number of transactions that completed in the amount of time represented by that bucket.

In Figure 24 on page 99, each of the fourteen buckets represents a percentage of the specified 1 second goal. For instance, bucket 2 represents all transactions that completed in 500 to 600 milliseconds, while bucket 8 contains the number of transactions that completed in 110% to 120% of the goal or 1.1 to 1.2 seconds. Notice that bucket 6 falls exactly on the goal. This bucket will capture all those transactions that complete in 900-1000 milliseconds.

The two end buckets have special meaning. Bucket 1 contains the total number of transactions that completed in up to 50% of the goal. Bucket 14 contains the number of transactions that completed in greater than 4 times the goal.

## Response Time Distributions

Service Class: XYZ  
Goal: 90% in 1 second

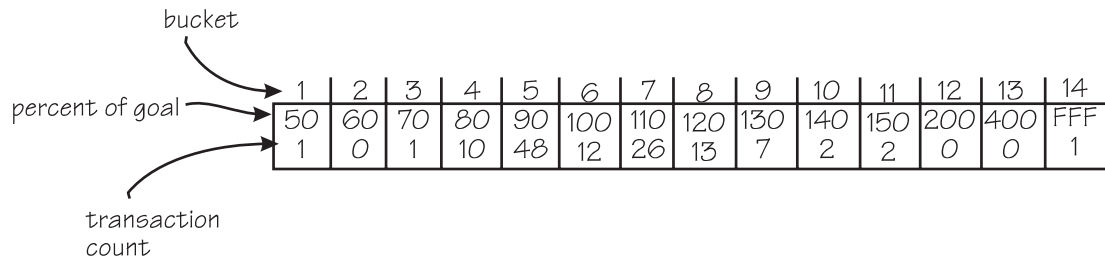


Figure 24. Self-defining response time distributions

## Interpreting Report Class Data

Through classification rules WLM allows you to associate transactions with report classes for reporting purposes only. Report classes can be used to report on a subset of transactions running in a single service class but also combine transactions running in different service classes within one report class. In the first case a report class is called homogeneous, and in the second case it is called heterogeneous.

Before z/OS 1.2, WLM only returned a report class summary. From the data it was not possible to conclude from where it came from, for example the service class, and what service class period contributed to the report class. For that reason, the calculation of response time distribution or velocity, for example, was not possible because there was no guarantee that

- the data in the report class is homogeneous, and
- the data is related to a single-period service class only.

Now, with z/OS 1.2 WLM returns report class periods. A report class period is homogeneous if there is only one service class found contributing to this report class period in a given report interval. To allow a reporting product to determine whether a report class period is homogeneous in its reporting interval, WLM offers two indicators returned by IWMRCOLL:

### mixed-class-indication timestamp

This timestamp indicates when workload data associated with a different service class last contributed data to a report class period that was currently collecting data from another service class (see Figure 25 on page 100).

### service class index

This index indicates the last service class whose data was collected in the report class period.

The following Figure 25 on page 100 illustrates the concept of the mixed-class-indication timestamp in relation to the time interval in which a caller collects workload data.

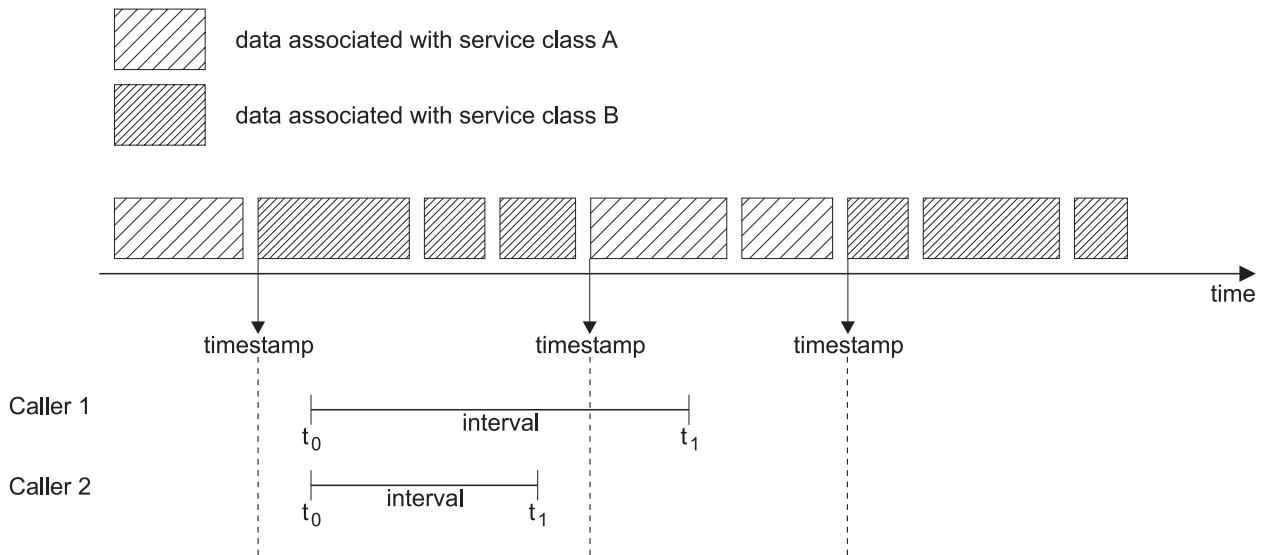


Figure 25. Mixed-class-indication timestamp in relation to the time interval

A caller invokes IWMRCOLL twice in order to get interval data, first at time  $t_0$  to start the interval and second at time  $t_1$  to end collecting data. With the second invocation at  $t_1$ , the caller gets back the mixed-class-indication timestamp. If the returned timestamp is  $\geq t_0$ , as it is for caller 1, it means that transaction data from a different service class started contributing data to the same report class period. The report class is heterogeneous. If the returned timestamp is outside the interval (smaller than  $t_0$ ), as it is for caller 2, it means that the report class remained homogenous during the calling interval.

Being able to see that a period is homogenous allows the reporting product to format response time distribution buckets and work manager delay data for this period while it would not report this data for a heterogeneous period. If the timestamp indicates that the report class is heterogeneous, it is recommended to collapse the periods which means to report the data as if the report class had only one period.

---

## Using the IWMRQRY Service

The IWMRQRY service provides sampled data on address spaces including:

- Address space state samples
- Server information

To see when an address space is serving more than one service class, information is provided for server address spaces in the general execution delay portion of the IWMWRCAA. This information includes:

- The service classes being served by an address space.
- The number of samples of address spaces serving a particular service class.

For example, address space ACSFOR1 is seen serving service class CICSSLOW 60 times, and service class CICSFAST 40 times.

The data returned is mapped by the IWMWRQAA data area. It represents the data collected during one sampling interval, an instantaneous, non-cumulative snapshot of the address space. The data is not tied to a particular job. Products using IWMRQRY must decide whether to acculmulate this latest state data with prior samples.

An asynchronous ENF signal is issued whenever a new copy of IWMWRQAA information is available. The performance monitor can use the ENF signal to determine when to issue IWMRQRY.

Table 20 shows a sample sequence of what to do to get address space information using IWMRQRY.

*Table 20. Using IWMRQRY with the workload reporting services*

<b>Action</b>	<b>Reason</b>
Issue IWMRQRY macro	To obtain answer area length.
Issue GETMAIN	To get storage to hold the address space data.
Issue IWMRQRY	To obtain address space data mapped by IWMWRQAA.
Set up a reporting interval	To collect data from issuing IWMRQRY multiple times.
Issues IWMRQRY INFO=ASID,ASCB=ascb	The performance monitor recognizes an exception for the address space represented by the specified ASCB. To obtain specific data about the ascb address space.





---

## Chapter 10. Using the Administrative Application Services

This chapter explains how to use the administrative application services. The services are intended for programs which provide an interface to define and edit a service definition. They include:

- **IWMDINST**, which allows a program to install a service definition on the WLM couple data set.
- **IWMDEXTR**, which allows a program to extract a service definition from the WLM couple data set.
- **IWMPACT**, which allows a program to activate a service policy.
- **IWMCQRY**, which allows a program to query the classification rules in effect.

For information about service definition concepts, see *z/OS MVS Planning: Workload Management*.

---

### Installing a Service Definition

The IWMDINST macro allows a program to install a service definition onto a WLM couple data set. The service definition is moved onto the WLM couple data set as an area of storage, mapped by the IWMSERVD macro.

### Mapping a Service Definition

The service definition is installed and extracted from the WLM couple data set as a data area mapped by the IWMSERVD mapping macro. The IWMSERVD macro points to the following sections:

#### **IWMSVDEF**

Maps the following service definition information:

- Service policies
- Workloads
- Service classes
- Report classes
- Resource groups
- Service definition coefficients

#### **IWMSVDCR**

Maps the service definition classification rule information.

#### **IWMSVNPA**

Maps the service definition notepad area.

#### **IWMSVAEA**

Maps the service definition application environments.

#### **IWMSVSEA**

Maps the service definition scheduling environments.

### Adding Program-Specific Extensions to a Service Definition

A program can add program-specific information to the service definition. For example, suppose you want to add a kind of reporting extension to a workload. In your program, you allow the service administrator to define this extension when defining workloads.

You can add extensions to each of the following:

- Service definition
- Service policies
- Workloads
- Service classes
- Report classes
- Resource groups
- Service definition coefficients
- Application environments
- Scheduling environments

A program can add extensions to IWMSVDEF, IWMSVDCR, IWMSVAEA, and IWMSVSEA. The following example illustrates extensions to IWMSVDEF. IWMSVDCR, IWMSVAEA, and IWMSVSEA all have roughly similar structures, and are extended in much the same way. For more details on these other structures see *z/OS MVS Data Areas, Vol 3 (IVT-RCWK)* or else see the individual prologs for those macros.

To add the extensions to the service definition structure, the program must update the following fields in IWMSVDEF:

**SVDEF\_EXT\_DATA\_OFF**

The offset to the extension data, from the beginning of IWMSVDEF

**SVDEF\_EXT\_DATA\_LEN**

The total length of the extension data, from the beginning of IWMSVDEF

For each part of the service definition with extensions, the program must update IWMSVDEF with the following:

- The offset to the extension entries from the beginning of IWMSVDEF.
- The number of extension entries.
- The length of the extension entries.

Each entry must have the following fields filled in:

**SVDEFVID**

Identifier of the product adding the extension.

**SVDEFROB**

The name of the related object. For example, if you are adding an extension to a service class, this field should contain the service class name.

**SVDEFEPN**

Related policy name, if the extension is for a service class or resource group. Otherwise, leave this field blank.

**SVDEFEDL**

The length of the extension data.

**SVDEFEDO**

The offset to the extension data.

**Example of Service Definition Extensions**

Figure 26 on page 105 shows the structure of IWMSVDEF for a service definition with extensions to workloads. In the example, there are two workload entries, and therefore two workload extensions. Remember that IWMSERVD points to the IWMSVDEF mapping macro.

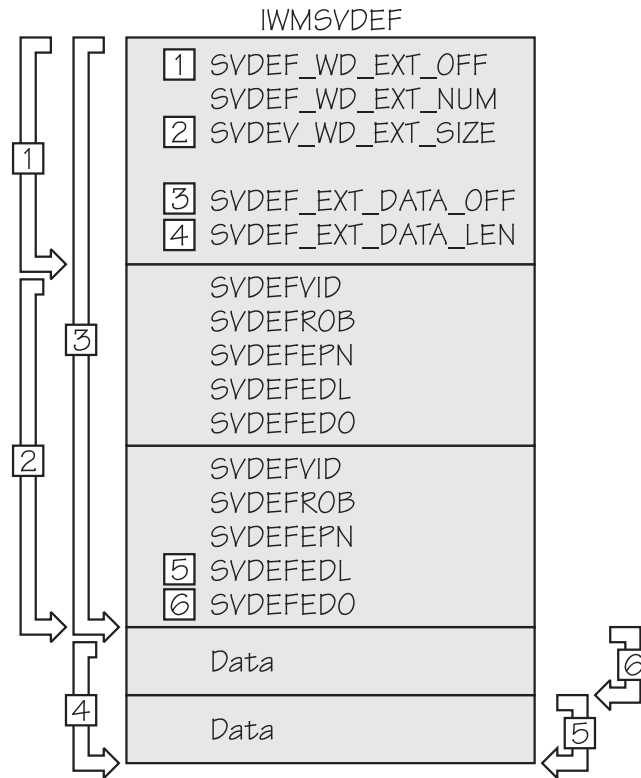


Figure 26. IWMSVDEF mapping for a service definition with workload extensions.

The program adding the workload extensions has updated the following fields in IWMSVDEF:

**SVDEF\_WD\_EXT\_OFF**

The offset to the workload extension data header from the beginning of IWMSVDEF.

**SVDEF\_WD\_EXT\_NUM**

The number of workload extension entries

**SVDEF\_WD\_EXT\_SIZ**

The size of the workload extension entries

**SVDEF\_EXT\_DATA\_OFF**

The offset to the beginning of the extension data, from the beginning of IWMSVDEF.

**SVDEF\_EXT\_DATA\_LEN**

The total length of the extension data, not including the length of the extension entries.

For each extension, the program must create an entry with the following fields:

**SVDEFVID**

Identifier of the your product.

**SVDEFROB**

The name of the workload related to this extension.

**SVDEFEPN**

Leave this field blank. Since this is a workload extension, it applies to all policies.

## SVDEFEDL

The length of the extension data.

## SVDEFEDO

The offset to the extension data from the start of the extension data.

## Maintaining the Service Definition

Make sure your program maintains the service definition data structure. For example, your program has added an extension to service classes, and the service administrator deletes that service class, then your program must ensure that the service class extension is also deleted.

In order for customers to allow for the extra space this information may take up on the WLM couple data set, there are new keywords on the JCL utility to allocate a couple data set. If your program intends to use the service definition extensions, you should make sure customers know how to factor the extensions into the size of the WLM couple data set. For information about how a service administrator allocates a WLM couple data set, see *z/OS MVS Planning: Workload Management*.

## Checking a Service Definition Using IWMDINST

Before installing a service definition on the WLM couple data set, the IWMDINST macro checks to make sure the service definition is valid. The macro checks the service definition to verify the service definition data structure and to detect any storage overlay problems.

All data in a service definition must be valid in order for the system to complete the installation. If the IWMDINST macro finds an error during validity checking, it issues a reason code in the VALCHECK\_RSN parameter. IWMDINST also returns the offset to the place in the IWMSVDEF mapping macro where the error was found in the VALCHECK\_OFFSET parameter. IWMDINST identifies one error per call.

Appendix B, “Application Validation Reason Codes,” on page 751 contains a list of the reason codes, and an explanation of each code. Table 21 shows an example of the information provided for reason code 1B01.

Table 21. SERVD validation reason codes

Section	Reason	Offset	Description
SVDCRSST	1B01	entry	Service class for the subsystem type (SVDCRSCN) not found in the SVDEF

In this example, IWMDINST returned a reason code of 1B01 in the VALCHECK\_RSN parameter. The error is in the SVDCRSST section of IWMSVDCR, and is in a subsystem type entry. IWMDINST found a service class in the classification rules for the subsystem type that was not defined in the service definition. Your program could check the service definition for the undefined service class name, and could issue a message to the service administrator.

## Recommended Validity Checking

In addition to using the validity checking provided by IWMDINST, a program should check for additional errors not covered by IWMDINST.

For example, suppose the service administrator has defined a service class representing IMS transactions, and assigned the service class to a resource group. Because MVS/ESA SP 5.2 does not support resource groups for

transaction-oriented work, you cannot assign a resource group to a service class representing IMS transactions. Rather than allowing the service administrator to make a mistake, your program should issue a warning message. The message could point the service administrator to the IMS service class containing the illegal resource group assignment.

Your program should check the following conditions not checked by IWMDINST:

- No workloads defined.
- No service classes defined.
- The response time goal for a service class period does not exceed the response time goal for the previous period.
- Duplicate service class or resource group names. The names must be unique within a service definition.
- Some rule within a subsystem type refers to a service class, but there is no default service class specified for the subsystem type. If a subsystem type has classification rules defined, then there must be a service class default defined.
- A qualifier type is repeated in a sub-rule for a subsystem type in the classification rules. Only qualifier type longer than 8 characters can be repeated in a sub-rule.
- The classification rules for a subsystem type refer to a service class with multiple periods, but that subsystem type does not support multiple periods. The subsystem types which support multiple periods are JES, ASCH, OMVS, STC, TSO, and DDF.
- The classification rules for a subsystem type refer to a service class with a resource group, but that subsystem type does not support resource groups. The subsystem types which support resource groups are JES, ASCH, OMVS, STC, TSO, and DDF.
- An unreferenced classification group. The service administrator defined a classification group, but did not use it in the classification rules.
- The classification rules for a subsystem type refer to a qualifier type not supported by the subsystem type. For a list of the qualifier types supported by each subsystem, see *z/OS MVS Planning: Workload Management*.
- Subsystem type does not support discretionary or velocity goals. Service class with a discretionary or velocity goal was referred to in the classification rules for that subsystem type. Subsystem types which support discretionary and velocity goals are: JES, ASCH, OMVS, STC, TSO, and DDF.

## Preventing Service Definition Overlays

Suppose a service administrator extracts a service definition from the WLM couple data set, makes some changes, and in the meantime someone else has extracted, changed, and reinstalled the service definition. If the first user installs the service definition, any changes the second user made will be overwritten. To prevent the service administrator from overwriting the service definition, a program can use:

- The COND parameter on IWMDINST
- The ENQ macro

### Using the COND Parameter on IWMDINST

The optional COND parameter lets a caller specify whether or not to install the service definition if the service definition has not been changed since it was extracted from the WLM couple data set. IWMDINST determines whether the service definition has changed by checking a service definition identifier.

The service definition identifier consists of the service definition name, and the time and date the service definition was installed.

Use COND=YES to install the service definition only if the identifier of the currently installed service definition matches the base identifier passed on IN\_BASEID parameter. If the identifiers do not match, your program can issue a message stating that the service definition has changed since the last extract. The message can also ask the service administrator to confirm the installation.

Use COND=NO to specify that the input service definition should be installed unconditionally.

When the service definition has been successfully installed on the WLM couple data set, the system issues ENF signal 41. A program can use the ENF macro to know when a service definition has been installed. For more information about how to use ENF signals, see *z/OS MVS Programming: Authorized Assembler Services Reference EDT-IXG*.

### Using the ENQ Macro

To ensure that two service definitions are not installed simultaneously on the WLM couple data set, a program can use the ENQ macro. Programs can serialize installation of the service definition onto the WLM couple data set by obtaining an exclusive ENQ on:

```
QNAME
    SYSZWLM
RNAME
    WLM_SERVICE_DEFINITION_INSTALL
SCOPE
    SYSTEMS
```

For more information about how to use the ENQ macro, see *z/OS MVS Programming: Authorized Assembler Services Reference EDT-IXG*.

## Example of Using IWMDINST to Install a Service Definition

Suppose you have coded an application to define and edit a service definition. You want to install the service definition onto the WLM couple data set, and do the following:

- Check the service definition data structure and whether there are any storage overlay problems. If there are any problems, have the macro return the location of the error in the service definition.
- Specify that the service definition is to be installed only if the service definition installed on the WLM couple data set has not changed since the last extract.

To install the service definition, you specify:

```
IWMDINST SERVD_AREA=(R4),
        PRODUCT_ID=ProductIdArea,
        QRY_BASEID=(R5),
        VALCHECK_RSN=ValcheckRsn,
        VALCHECK_OFFSET=ValcheckOffset,
        COND=YES,
        RETCODE=Module_Rc,
        RSNCODE=Module_Rsn,
        MF=E
```

With the following defined:

ProductIdArea	DS	CL32	My application product identifier
ValcheckRsn	DS	1F	The validation check reason code
ValcheckOffset	DS	1F	The validation check offset
Module_Rc	DS	1F	The return code
Module_Rsn	DS	1F	The reason code

- To check the service definition data structure, use the VALCHECK\_RSN and VALCHECK\_OFF parameters. If IWMDINST finds any error, it will provide the reason code in the VALCHECK\_RSN parameter, and the offset to the error in the VALCHECK\_OFF parameter. Look up the reason code in Appendix B, “Application Validation Reason Codes,” on page 751 for more information about the error and help in locating it in IWMSERVD.
- To specify that the service definition is to be installed only if the service definition installed on the WLM couple data set has not changed since the last extract, use the COND=YES parameter. If the service definition base identifier has not changed since the service definition was extracted from the WLM couple data set, then IWMDINST continues with the installation.

---

## Extracting a Service Definition

The IWMDXTR macro allows a program to extract a service definition from the WLM couple data set. Once the service definition is extracted, then a service administrator can make changes.

A caller should issue IWMDXTR specifying QUERYLEN to determined the storage required for the service definition. The service definition returned by IWMDXTR is not serialized against future installs, so the length returned could change before a caller can issue IWMDXTR again. The caller should issue IWMDXTR in a loop, checking return and reason codes, and obtaining a larger storage area if necessary.

## Example of Using IWMDXTR to Extract a Service Definition

To extract a service definition from the WLM couple data set, specify:

```
IWMDXTR ANSAREA=(R4),
        ANSLEN=StorSizeForServd,
        QUERYLEN=QueryLenForServd,
        RETCODE=Module_Rc,
        RSNCODE=Module_Rsn,
        MF=E
```

StorSizeForServd	DS	1F	Storage size for service definition
QueryLenForServd	DS	1F	Query length for service definition
Module_Rc	DS	1F	Return code
Module_Rsn	DS	1F	Reason code

---

## Activating a Service Policy

You can also activate a service policy from your application by using the IWMPACT macro. The caller must provide the name of the service policy to be activated in the SERVICE\_POLICY=*service\_policy* parameter. The specified policy must exist in the service definition installed on the WLM couple data set.

A single policy can be activated at any one time, and the policy is activated synchronously.

A caller can optionally request the name of the system where another policy activation is taking place by specifying SYSTEM\_NAME=*system\_name*. Therefore, if a previous IWMPACT request is being processed and a new IWMPACT request is issued, the new request is rejected with an appropriate return and reason code. This occurs regardless of whether the requests were issued on the same system or

different systems in the sysplex. Control is not returned to the caller until the policy has been activated on all systems in the sysplex or for some reason the policy activation could not be completed.

## Example of Activating a Policy using IWMPACT

To activate a service policy, and if another policy activation is in progress, have IWMPACT return the name of the system that activated the policy, specify:

```
IWMPACT POLICY_NAME=policy,  
        SYSTEM_NAME=system,  
        RETCODE=Module_rc,  
        RSNCODE=Module_rsn
```

Where the following are defined:

policy	DS	CL8	The policy name
system	DS	CL8	System name where another policy is activating
Module_rc	DS	1F	Return code
Module_rsn	DS	1F	Reason code

---

## Querying the Active Classification Rules

IWMCQRY lets a caller query the classification rules associated with the active service policy. The classification rules determine how incoming work is assigned a service class and/or report class. The data returned by this macro is mapped by IWMSVDCR. For a description of the IWMSVDCR macro, see *z/OS MVS Data Areas, Vol 3 (IVT-RCWK)*.

Optionally, a caller can request the active service policy identifier by specifying the POLICY\_ID parameter. This is the active policy containing the classification rules returned by this macro. The caller can then compare the service policy ID with the policy identification information returned by the IWMPQRY macro to ensure they are the same.

Some data sections in the IWMSVDCR data area may not be available through IWMCQRY. For example, the time stamps indicating when a classification GROUP was last updated and by whom may not be available. For a complete list of fields that are not available refer to IWMSVDCR as described in *z/OS MVS Data Areas, Vol 3 (IVT-RCWK)*. The complete classification rules associated with a service policy are returned by the IWMDEXTR macro and mapped by IWMSVDCR.

A caller can use the classification rules together with the active service policy to determine the goals associated with incoming work. The service class goals are in the active service policy mapping returned by the IWMPQRY service.

The information returned is not serialized upon return to the caller, and so may be out of date if a service definition was modified, installed, and a new policy activated.



## Example of IWMCQRY

To query the classification information associated with the active policy, specify:

```
IWMCQRY POLICY_ID=(R7),  
        ANSAREA=(R5),  
        ANSLen=anslen,  
        QUERYLEN=cqry_len,  
        RETCODE=RCODE,  
        RSNCODE=RSN
```

Where the following are defined:

anslen	DS	1F	Length of the answer area
cqry_len	DS	1F	Length required for answer area
RCODE	DS	1F	Return code
RSNCODE	DS	1F	Reason code



---

## Chapter 11. Using SMF Record Type 99

SMF record type 99 provides detailed audit information for work run on z/OS. You can use the type 99 records for analyzing the performance characteristics of work. The records contain performance data for each service class period, a trace of SRM actions, the data SRM used to decide which actions to take, and the internal controls SRM uses to manage work. This can help you determine in detail what SRM is doing to meet your work's goals with respect to other work, and the types of delays the work is experiencing.

You should write SMF type 99 records only when you want this detailed information. For general reporting and tuning information for a system, you can use SMF type 72 records.

SRM writes type 99 records for each *policy interval*, or approximately once every 10 seconds.

There are the following type 99 record subtypes:

### **Subtype 1**

Contain system level data, the trace of SRM actions, and data about resource groups. A subtype 1 record is written every policy interval.

### **Subtype 2**

Contain data for service classes. A subtype 2 record is written every policy interval for each service class if any period in the service class had recent activity.

### **Subtype 3**

Contain service class period plot data. A subtype 3 record is written every policy interval for each service class if any period in the service class had recent activity and plot data.

### **Subtype 4**

Contains information about a device cluster. A device cluster is a set of service classes that compete to use the same DASD devices. A subtype 4 record is written every policy interval for each device cluster in the system.

### **Subtype 5**

Contain data about monitored address spaces. A subtype 5 record is written each policy interval for each swapped in monitored address space.

### **Subtype 6**

Contains summary information about each service class period, including the resource control settings for the next policy interval. A subtype 6 record is written each policy interval.

### **Subtype 7**

Contains summary information for the Enterprise Storage Server (ESS) with Parallel Access Volume (PAV) feature. A subtype 7 record is written every third policy interval.

### **Subtype 8**

Contains summary information for LPAR CPU management. A subtype 8 record is written each policy interval, when in LPAR mode.

### **Subtype 9**

Contains summary information for dynamic channel path management. A subtype 9 record is written each policy interval.

Some fields are used as input to adjust the policy. Those fields contain the data collected during the policy interval just prior to policy adjustment. Other fields are altered when the policy is adjusted. Those fields contain the data resulting from the policy adjustment. For example, a service class period dispatch priority field contains the dispatch priority for the next policy interval not the dispatch priorities from the previous policy interval.

---

## When to Start SMF Record Type 99

A type 99 is written every policy interval, which is frequent! The records contain the data, plots, and tables SRM uses to assess the effects of changes. So you should write type 99 records only for a time period in which you want this detailed audit information.

## Starting SMF Record Type 99

You specify the type 99 record in the SMFPRMxx parmlib member under SUBSYS STC.

Because SMF type 99 records are written approximately every 10 seconds, you should write them only for certain time periods. If you use NOTYPE in your SMFPRMxx parmlib member, you should include type 99 in your NOTYPE list. For example:

```
SUBSYS(STC,NOTYPE(99))
```

If you use TYPE in your SMFPRMxx parmlib member, make sure you add TYPE 99 only when you want this level of detail. For example, add:

```
SUBSYS(STC,TYPE(99))
```

You should have an SMFPRMxx parmlib member for general audit information that does not specify type 99, and another for detailed audit information that specifies type 99. This way, you can write the frequently written SMF record types like SMF type 99 only when you need them.

The mapping macro (IRASMF99) for this record is shipped in SYS1.AMODGEN.

---

## Identifying Work in SMF Type 99 Records

This section explains how to identify work in SMF type 99 records. You can identify work throughout SMF type 99 by:

- Service class name
- Service class performance period number
- Resource group name
- Address space name

This information is as defined in the service policy at the time the type 99 record was written.

## Identifying Server Service Classes

Some service classes are *server service classes*, that is they are service classes representing address spaces doing work on behalf of transactions. You can determine whether a service class represented in the subtype 2 period data is a server by first checking the goal type in the service class period section. If the goal type is 0, indicating a system or server service class, you next check whether there

are any server data entries. If there are entries, then the service class period is a server, and the server data describes the service classes being served by this server.

If the goal type is 0, and there are no server entries, then the service class is a system service class.

## Identifying Internal Service Classes

SRM groups some work not defined in the service policy into internal service classes. The internal service class names are:

### **\$SRMDInn - resource group discretionary**

Contains all work in a resource group with a discretionary goal. There is one \$SRMDInn class per resource group with discretionary work.

### **\$SRMDI00 - general discretionary**

Contains all work assigned a discretionary goal, but not assigned a resource group. There is always one \$SRMDI00.

### **\$SRMDISC - SYSOTHER service class**

Contains all work not otherwise assigned to a service class.

### **\$SRMSnnn - server**

Contains all address spaces serving the same set of service classes. That is, the server address space (s) could be serving more than one service class. For example, 2 AORs may be serving the same 3 CICS service classes. Those 2 AORs make up one \$SRMSnn service class.

An address space can belong to one internal server service class, but can move from one class to another. The number of \$SRMSnnn service classes depends on how many external service classes are served and by how many combinations of server address spaces.

### **\$SRMDUMP - SDUMP**

Contains only SDUMP.

### **\$SRMBEST - best**

Contains the special system component address spaces, unless they are explicitly assigned to a service class in the service policy. This includes the SYSTEM service class.

### **\$SRMGOOD - good**

Contains the STC subsystem type default work, if there was no default service class specified in the classification rules for STC in the service policy. This is the SYSSTC service class.

### **\$SRMGOD1-5 - good**

Contains the work defined for service classes SYSSTC1, SYSSTC2, SYSSTC3, SYSSTC4, and SYSSTC5. For further information refer to *z/OS MVS Planning: Workload Management*.

---

## Interpreting Trace Table Entries

Subtype 1 contains a trace table. The trace codes describe which action SRM took or is considering taking to process work. This section describes the SRM concepts reflected in the trace codes. For a list and a brief explanation of the trace codes, see Appendix A, "SMF Type 99 Action Codes," on page 707.

## Policy Adjustment

*Policy adjustment* is how SRM:

- Selects work to help
- Selects performance bottlenecks to address
- Selects which work should donate resource to help other work
- Assesses whether it is worth it to help work.

The purpose of policy adjustment is to meet service class and resource group goals. Policy adjustment is done approximately every 10 seconds.

## Resource Adjustment

*Resource adjustment* is how SRM keeps system resources effectively utilized.

Resource adjustment includes:

- Detecting and addressing under-utilized, over-utilized, and shortage conditions.
- Managing individual working sets via Working Set Management,
- Managing expanded storage migration via expanded storage policies for service class periods and individual address spaces.

SRM handles resource adjustment algorithms within the constraints set by the policy adjustment algorithms.

Resource adjustment is done approximately every 2 seconds.

## Receivers and Donors

A *receiver* is the service class period SRM is considering helping. SRM helps only one receiver during each policy interval, although it may assess multiple receivers before finding one to help. A *donor* is a service class period that donates resources to the receiver. Multiple donors may donate multiple resources to a single receiver during one policy interval.

The resources to help the receiver may also come out of what is referred to as *discretionary* resources. Discretionary resources are those that can be reallocated with little or no effect on the system's ability to meet performance goals.

Service class periods other than receivers and donors can also be affected by changes. These service class periods are referred to as *secondary* receivers and donors. SRM may decide not to help a receiver due to minimal net value for either a primary or secondary donor. "Action Trace Example" on page 127 explains an example in which there are secondary receivers and donors.

If a service class period is being served by one or more address spaces, it is called the *goal* receiver or donor. It is the service class period with the response time goals. To help such service class periods, SRM must donate resources to the server address spaces. The service class period serving a service class, is called a *resource* receiver or donor. SRM adjusts resources for the resource receiver/donor to affect the performance of the goal receiver/donor.

## Performance Index

*Performance index* is a calculation of how well work is meeting its defined goal. For work with response time goals, the performance index is the actual divided by goal. For work with velocity goals, the performance index is goal divided by actual.

A performance index of 1.0 indicates the service class period is exactly meeting its goal. A performance index greater than 1 indicates the service class period is

missing its goal. A performance index less than 1.0 indicates the service class period is beating its goal. Work with a discretionary goal is defined to have a performance index of .81.

Each service class period has a sysplex and a local performance index. The sysplex performance index represents the performance of a service class period across all the systems in the sysplex. The local performance index represents only the performance on the local system.

Within resource groups and importances, receivers are selected in *performance index* order. Donors are selected in the reverse order of receivers. The sysplex performance index is the primary criteria used for selecting receivers and donors and assessing changes.

## Receiver Value

A receiver is helped only if there is projected to be sufficient *receiver value*. Receiver value is a minimum performance index improvement, a minimum group service increase, or other minimum criteria designed to reject very small improvements. The reason to reject actions for too little receiver value is to get on to addressing other problems for other service class periods rather than continuing to make changes that yield only marginal improvements for a few service classes.

## Net Value

A receiver is only helped by a specific donor if SRM projects sufficient *net value* to the resource reallocation. SRM calculates the net value and uses it to determine if using a donor to help a receiver results in more projected harm to the donor than projected improvement to the receiver. If so, the condition is traced, and another donor is selected. The net value assessment considers all external service policy specifications (resource group capacity minimums and maximums, importance, and goals) for both primary and secondary donors.

## Small Processor Consumer

Service class periods that consume little or no processor time are referred to as *small processor consumers*. Under some circumstances these small consumers are assigned a relatively high dispatching priority and not assessed for dispatching priority changes.

## Storage Housekeeping

The purpose of *storage housekeeping* is to decrease storage targets that are out of date or ineffective. Storage housekeeping will only reduce a target if it will not affect the ability of work to meet goals. There are four types of storage housekeeping.

### **time driven**

Reduce target when the projected effect is small and the service class period will still easily meet goals

### **minimal effect**

Reduce target slightly when the projected effect is very small

### **unassessable**

Reduce target slightly when the projected effect cannot be determined but the service class period is meeting goals easily

### **slow mode**

Reduce target slightly when the projected effect cannot be determined but keep the target close to the current allocation

## Reverse Housekeeping

A second type of housekeeping, referred to as *reverse housekeeping* increases targets if these targets are significantly less than the resources the service class period or address space already owns. The targets are increased by reverse housekeeping so that if there is a sudden increase in the demand for resources the service class period or address space will have some protection until the policy adjustment can reevaluate the situation.

## Working Set Management

*Working set management (WSM)* improves system efficiency by adjusting:

- The mix of work in the multiprogramming set
- The central and expanded storage allocated to individual address spaces.

Any adjustments may be made as long as the adjustments do not jeopardize externally specified goals.

WSM monitors address spaces to determine their paging characteristics. Address spaces may be selected for monitoring by policy adjustment or WSM. The point (number of frames) where an address space is not experiencing significant paging overhead is referred to as its *ok1* point. WSM tries to run address spaces at their *ok1* points if possible.

If there is not enough storage to run at *ok1* but some storage is available, WSM puts a restrictive target on the address space. This is referred to as *managing* or *squeezing* the address space. If there is not enough storage for the address space to run at all productively, WSM initiates a swap out. WSM also assesses reallocating central or processor storage frames from one or more address spaces to another address space to determine whether the reallocation would result in a more productive use of storage. These actions are referred to as *a's frames to b* actions.

---

## Interpreting Management Policy Data

This section explains the controls SRM uses to manage work, and where these controls can be found in SMF type 99. SRM algorithms set all the control values internally. All changes to the control values have trace entries in the subtype 1 record.

The SRM controls are:

- Dispatching priority
- MPL targets
- Swap protect time
- Expanded storage policy
- Storage targets
- Cap slices
- I/O priority
- Number of server address spaces

They are shown in the subtype 1, subtype 2, and subtype 5 records.

## Dispatching Priority

SRM defines dispatching priority for service class periods. All address spaces in a service class period have the same base dispatching priority. Multiple service class periods may have the same base dispatching priority. After a dispatching priority change, service class periods may be remapped to different dispatching priorities



such that there is an unoccupied priority between each occupied priority. This process is referred to as priority *unbunching*.

The dispatching priority is recorded in the subtype 2 records.

## MPL Targets

SRM defines an MPL-in and MPL-out target for each service class period. MPL-in target represents the number of address spaces that must be in the swapped-in state for the service class period to meet its goals. MPL-out target is the maximum number of address spaces allowed in the swapped-in state.

The MPL targets are recorded in the subtype 2 records.

## Swap Protect Time

SRM defines swap protect time for service class periods. Swap protect time is the time in milliseconds swapped-out address spaces will remain in processor storage before becoming candidates for swap to auxiliary storage.

The swap protect time is recorded in the subtype 2 records.

## Expanded Storage Policies

An expanded storage policy controls how and which pages are written to expanded or auxiliary storage. There is an expanded storage policy for each of the following types of pages:

- Swap working set pages
- VIO pages
- Hiperspace pages
- Stolen pages and swap trim (demand pages)

To determine how to write the pages, each of the expanded storage policies is set to one of the following:

### **Protected**

Direct pages to expanded storage, where the system protects a specific number of pages on expanded storage.

**LRU** Direct pages to expanded storage, where the system migrates pages in least recently used (LRU) order.

### **Space available**

Direct pages to auxiliary storage unless there is significant space available on expanded storage.

SRM defines the expanded storage policy for swap working set pages on a service class period basis.

SRM defines the expanded storage policy for all other types of pages depending on the service class period goal. For a short response time goal (less than or equal to 20 seconds), SRM defines the expanded storage policy on a service class period basis. For long response time goals (greater than 20 seconds), velocity goals, discretionary goals, and server internal service classes (\$SRMSnnn service class names), SRM may define an individual expanded storage policy for each address space in the service class period.

If there is an individual expanded storage policy for an address space, subtype 2 provides:

- The number of address spaces subject to each policy
- The offset to the address space expanded storage policy entries.
- The length of each address space expanded storage policy entry.
- The number of address space expanded storage policy entries.
- The policies for each address space.

## Storage Targets

SRM defines storage targets to manage expanded and central storage. The storage targets are:

### **protective processor**

The number of frames that are protected in processor (central + expanded) storage. The system will not migrate below this value. There are targets for: address spaces, common area storage, and shared storage.

### **protective central**

The number of frames that are protected in central storage. The system will not steal below this value.

### **restrictive processor**

The number of frames in processor (central + expanded) storage to which an address space is limited. If the target is exceeded, the system will preferentially migrate the excess frames to auxiliary storage.

### **restrictive central**

The number of frames in central storage to which an address space is limited. If the target is exceeded, the system will preferentially steal the excess frames.

The storage target assigned depends on the service class period goal.

For short response time goals, the service class period gets a protective processor storage target. Every address space in the service class period has the same target.

For long response time, velocity, or discretionary goals, or for server internal service classes (\$SRMSnnn), SRM may define individual storage targets for the address spaces in the service class period. Each address spaces may have one or more of each type of storage targets. Storage targets for individual address spaces are recorded in subtype 5, in the monitored address space information section.

## Cap Slices

SRM defines cap slices for resource groups to enforce resource group maximums. Work is not dispatched during its cap slices in order to reduce access to the processor to enforce the resource group maximum. Each cap slice represents 1/64th of total time. All address spaces in all service class periods in a resource group are controlled by the same number of cap slices.

The number of cap slices is recorded in the subtype 1 records.

## I/O Priority

SRM defines an I/O priority for each service class period. All address spaces in a service class period have the same I/O priority. Multiple service class periods may have the same I/O priority. I/O priority is used to prioritize requests on IOS's UCB queues.

The I/O priority is recorded in the subtype 2 records.

## Number of Server Address Spaces

SRM manages the number of server address spaces for users of the queueing manager services (see Chapter 5, “Using the Queueing Manager Services,” on page 61) and manages the number of initiators for WLM-managed JES job classes.

Information on how SRM manages these server address spaces is recorded in the subtype 2 records in the queue server data section and in the remote queue server data section.

---

## Interpreting Plots

SRM creates the following plots to track how well work is being processed:

- System paging delay plot
- Period MPL delay plot
- Period ready user average plot
- Period swap delay plot
- Period paging rate plot
- Period proportional aggregate speed plot
- I/O delay plot
- Queue delay plot
- Address space paging plot
- I/O velocity plot

All plots except the address space paging plot are “one-curved” plots where one variable is plotted against another. The address space paging plot is a “three curved” plot where one variable is plotted against three variables.

## System Paging Delay Plot

SRM uses the system paging delay plot to determine if the paging configuration is close to capacity or if it can support additional work. There is one system paging plot per system. The plot can show the point at which additional page faults will start requiring a disproportionately longer time because the paging subsystem is becoming overloaded.

### System Paging Delay Plot

**x axis**

The system wide page fault rate, in page faults per second.

**y axis**

The system wide auxiliary storage delay sample rate, in samples per minute.

Page faults and delay samples are for all types: private area, common area, and cross memory.

The system paging delay plot is recorded in subtype 1 records.

## Period MPL Delay Plot

SRM uses the period MPL delay plot to assess increasing or decreasing a service class period's MPL targets. The plot shows how response time may be improved by increasing MPL slots or how response time may be degraded by reducing MPL slots.

### Period MPL Delay Plot

**x axis**

The percentage of ready users who have an MPL slot available to them.

**y axis**

The MPL delay per completion in milliseconds.

The period MPL delay plot is recorded in subtype 3 records.

## Period Ready User Average Plot

SRM uses the period ready user average plot to predict the number of ready users when assessing an MPL target change. The plot can show the point at which users will start backing up. The plot shows the approximate MPL target at which users would be disproportionately delayed due to MPL.

### Period Ready User Average Plot

**x axis**

The number of MPL slots, times 16, available to the service class period.

**y axis**

The maximum number of ready users, times 16, averaged over a two second interval.

The period ready user average plot is recorded in subtype 3 records.

## Period Swap Delay Plot

SRM uses the period swap delay plot to assess increasing or decreasing swap protect time for a service class period. The plot shows how response time may be improved or degraded by increasing or decreasing a service class period's swap protect time.

### Period Swap Delay Plot

**x axis**

The average time an address space in the service class period is logically swapped or swapped on expanded, in milliseconds.

**y axis**

The swap delay time per completion, in milliseconds.

The period swap delay plot is recorded in subtype 3 records.

## Period Paging Rate Plot

SRM uses the period paging rate plot to assess increasing or decreasing period wide storage isolation for a service class period.

Period Paging Rate Plot	
<b>x axis</b>	The average address space size in frames.
<b>y axis</b>	The page fault rate in tenths of a page fault per departure from the period.

The period paging rate plot is recorded in subtype 3 records.

## Period Proportional Aggregate Speed Plot

SRM uses the proportional aggregate speed plot to assess the effects of processor access or storage allocation changes for served service classes. Proportional aggregate speed is similar to velocity. Proportional aggregate speed applies to service class periods that are served while velocity applies to service class periods that are not served. The units for proportional aggregate speed are the same units as for velocity:

$$\frac{\text{using samples}}{\text{using samples} + \text{delay samples}} \times 100$$

The samples are an aggregate of the samples of all internal server service classes that serve the service class. The server samples are apportioned to the served classes based on the relative amount of time, also determined by state sampling, that the server is serving the service class.

Proportional Aggregate Speed Plot	
<b>x axis</b>	The <i>proportional aggregate speed</i> of a service class period.
<b>y axis</b>	The performance index at that speed, times 100.

The period proportional aggregate speed plot is recorded in subtype 3 records for served service class periods.

## I/O Delay Plot

SRM uses the I/O Delay plot when assessing increasing or decreasing a service class period's I/O priority.

#### **I/O Delay Plot**

**x axis**

The combined maximum I/O demand of service class periods with I/O priorities above a given priority.

**y axis**

The ratio of I/O delay time to I/O using time at that priority scaled by 16.

Maximum I/O demand is the maximum percentage of time that work units in a service class period would use non-paging DASD devices if they were experiencing no I/O delay. Maximum I/O demand is represented as a percentage scaled by 10.

The I/O delay plots are recorded in the subtype 4 records.

### **Queue Delay Plot**

SRM uses the queue delay plot to assess creating or removing server address spaces. The plot shows how queue delay for a service class period may be reduced by adding server address spaces for work running in the service class period or how queue delay may be increased by removed server address spaces for work running in the service class period.

#### **Queue Delay Plot**

**x axis**

The ratio of work requests that require a task in a server address space to the number of server tasks. This ratio is scaled by 100.

**y axis**

The queue delay per work request in milliseconds.

The queue delay plots are recorded in the subtype 3 records.

### **Address Space Paging Plots**

SRM uses address space paging plots when assessing whether to increase or decrease the central storage or processor storage allocated to an address space. There are two address space paging plots:

#### **Central Storage Plot**

SRM uses the central storage paging plot when assessing increasing or decreasing the central storage allocated to an address space.

### Central Storage Plot

#### x axis

Address space size in frames.

#### y axis

Each one of the following:

- Page-in rate per captured (task and SRB) second from auxiliary and expanded storage.
- Paging cost in milliseconds per elapsed second from auxiliary and expanded storage.
- Captured time in milliseconds per elapsed second.

### Processor Storage Plot

SRM uses the processor storage paging plot when assessing increasing or decreasing the processor storage allocated to an address space.

### Processor Storage Plot

#### x axis

The address space size in frames.

#### y axis

Each one of the following:

- Page-in rate per captured (task+SRB) second from auxiliary storage.
- Paging cost in milliseconds per elapsed second from auxiliary storage.
- Captured time in milliseconds per elapsed second.

The address space paging plots are recorded in subtype 5 records.

## I/O Velocity Plot

SRM uses the I/O velocity plot to keep track of the relationship between the number of channel paths connected to a subsystem (along with the channel path's utilization) and the I/O velocity of that subsystem. There is one I/O velocity plot for each I/O subsystem (control unit).

### I/O Velocity Plot

#### x axis

The contention factor.

#### y axis

The I/O velocity.

If there is a single channel, the contention factor equals the channel utilization. If there are multiple channels, the contention factor is the utilization a single channel would have to have to be equivalent to the channels connected to the control unit, given the number of channels and their average utilization. In this case, "equivalent" means that an I/O operation would have the same probability of experiencing delay.

The I/O velocity plots are recorded in the subtype 9 records.

---

## Interpreting Priority Table Data

Subtype 1 priority table data contains processor resource demand and consumption at each dispatching priority in use. Subtype 2 records contain similar data for each service class period. You can use the data in these two records to determine how much processor capacity is available to each service class period or to explain the actions being taken to increase or decrease access to the processor.

You can use the priority table data in subtype 1 to understand why dispatching priority change actions are rejected. When a dispatching priority change is made, the table shows the before and after demand and consumption data at each priority in use. The before data is actual data. The after data is projected data.

*Maximum demand* is the theoretical maximum percentage of total system processor time the address spaces in a service class period would consume if they were suffering no processor delays. This value is calculated for each service class period and accumulated for each priority.

*Achievable maximum demand* is the percentage of total system processor time the address spaces in a service class period are projected to use, given the maximum demand of all work at higher dispatching priorities. SRM calculates achievable maximum demand to assess dispatching priority changes.

---

## Interpreting Lack of Action

Just as you can determine the actions SRM takes to manage work, you can determine lack of action from SMF type 99. You can use the resource group and service class period information in the subtype 1 and 2 records. If a service class period is having a problem meeting its goals and isn't selected as a receiver, it could be one or more of the following:

- Other work is even worse off.
- The service class period is of lower importance than the receiver selected.
- The receiver selected may be in a resource group that is not meeting its minimum.

The importances and performance indexes for all service class periods are in the subtype 2 records. Resource group information is in the subtype 1 records.

In some cases, potential receivers may be skipped and not assessed as receivers. They are skipped when the service class period hasn't accumulated any state samples that show it was delayed for any resource SRM manages during the last policy interval. You can determine this from the *needs help* indicator in the service class period data.

They are also skipped when a potential receiver's *skip clock* hasn't expired. If a receiver is assessed and all actions to help it are rejected, a skip clock (counter) is set and the service class period will not be selected as a receiver again until the skip clock expires.

In other cases, a service class period is selected as a receiver but not helped. It could be that there was no receiver value projected for a change. The subtype 1 trace entry indicates these cases with the *no receiver value* trace code.



For storage changes such as MPL targets, swap protect time, or storage targets, you can determine the reasons for insufficient receiver value from the service class period's current targets in the subtype 2 records and the plots in the subtype 3 and 5 records.

For rejected dispatching priority actions, you can determine the reasons from the service class period's dispatching priority, service, and maximum demand data in the subtype 2 records and at the priority table data in the subtype 1 records.

For no net value assessments, you can determine the reasons from the service class period data, plot data, and priority table data. From this data, you can also determine which service class periods are using the resources for which another service class period is delayed.

---

## Examples of Interpreting SMF Record Type 99

This section describes the following examples on interpreting the data in SMF type 99:

- Action trace
- MPL policy

The examples show information from SMF type 99 records that were combined and displayed in a report format.

### Action Trace Example

This example shows how to use subtype 1 trace data with subtype 2 service class period data to understand what actions SRM is taking, why those actions are taken, and which work is affected.

The table below shows the subtype 1 trace data output:

Subtype 1 Trace Data Output					
CLASS	P	SPI	LPI	ACTION	
NRBATCH	2	.02	.03	3610	rv_hsk_inc_mpl
TSO	1	.10	.11	2630	tdh_rem_prt
TSO	2	1.84	1.84	270	pa_rec_cand
TSO	1	.10	.11	880	pa_pro_rdon_cand
TSO	2	1.84	1.84	620	pa_pmuo_rec
APPC	1	1.10	1.10	960	pa_pro_unc_sec_don
APPC	3	1.10	1.10	960	pa_pro_unc_sec_don
NRBATCH	1	2.23	3.23	960	pa_pro_unc_sec_don
TSO	1	.10	.11	940	pa_pro_unc_don
TSO	2	1.08	1.08	750	pa_pro_incp_rec

The data has the following headings:

**CLASS**

The service class name.

**P**

The service period number within the service class.

**SPI**

The sysplex performance index for the service class period traced.

**LPI**

The local performance index for the service class period traced.

**ACTION**

The action code and mnemonic.

The table below shows the subtype 2 service class period output.

### Subtype 2 Service Class Period Data

CLASS	P	I	N	SC	GT	SPI	LPI	DP	MPLI	MPLO	SWPT	PSI	EXP	POL
APPC	1	2	N	-99	SRT	1.10	1.10	251	0	999	488	0	L L L L	
APPC	2	2	N	-99	SRT	0	0	251	0	999	0	0	L L L L	
APPC	3	2	N	-72	SRT	1.10	1.10	251	1	999	76206	0	L L L L	
APPC	4	3	N	-99	VEL	0	0	249	1	999	0	n/a	S 0/2/0	2/0 2/0
NRBATCH	1	3	N	-99	LRT	2.16	3.04	249	1	11	0	n/a	S 0/2/0	2/0 2/0
NRBATCH	2	4	Y	-98	LRT	.02	.03	247	2	999	0	n/a	S 0/3/0	3/0 3/0
NRBATCH	3	6	Y	-99	DIS	.81	.81	192	0	14	0	n/a	S 0/7/0	7/0 7/0
OFFBATCH	1	6	Y	-99	DIS	.81	.81	192	0	14	0	n/a	S 0/7/0	7/0 7/0
TSO	1	2	Y	-64	SRT	.10	.11	251	1	11	0	0	L L L L	
TSO	2	2	Y	0	SRT	1.84	1.84	251	1	999	1464	0	L L L L	
TSO	3	2	Y	-2	SRT	.91	.93	247	2	12	2440	0	L L L L	
TSO	4	3	Y	-99	VEL	.26	.26	245	7	999	0	n/a	S 0/6/0	6/0 6/0
\$SRMDISC	1	6	Y	-99	DIS	.81	.81	192	0	14	0	n/a	S 0/7/0	7/0 7/0

The subtype 2 service class policy period output has the following headings:

#### CLASS

The service class name.

**P** The service period number within the service class.

**I** The service class period's importance, with 1 being highest.

**N** The "needs help" indicator. Y means the service class period needs help and N means the service class period doesn't need help.

**SC** The skip clock.

**GT** One of the following goal types:

- SRT - short response time goals
- LRT - long response time goals
- VEL - velocity goals
- DIS - discretionary goals.

**SPI** The sysplex performance index for the service class period traced.

**LPI** The local performance index for the service class period traced.

**DP** Dispatching priority

**MPLI** MPL-in target

**MPLO** MPL-out target

**SWPT** Swap protect time

**PSI** Protective processor storage target

#### EXP POL

The expanded storage access policies, in the following order from left to right:

- Swap working set pages
- Stolen pages and swap trim (demand pages)
- VIO pages
- Hiperspace pages

Where P=protected, L=LRU, and S=space available. If there are policies for individual address spaces, then there is a triplet indicating the number of address spaces with each setting (P,L,or S).

### Interpreting the Trace Data

Based on the information in these subtypes, SRM took the following actions:

- *rv\_hsk\_inc\_mpl*, reverse housekeeping, to increase the MPL targets for service class NRBATCH, period 2.

This action code indicates the service class period was using significantly more MPL slots than guaranteed by its MPL control so its MPL in target was increased.

- *tdh\_rem\_prt*, time driven housekeeping, removing the swap protect time target from first period TSO.  
First period TSO is easily meeting its goals and was assessed to not need swap protect time.
- *pa\_rec\_cand* indicates TSO period 2 was chosen as the receiver.  
This shows the start of a series of actions where SRM is trying to improve the performance of TSO period 2 by increasing its dispatching priority. SRM selected TSO period 2 as the receiver because it had the worst performance index, 1.84, of all the work at the highest importance defined, 2. Service class NRBATCH, period 1, had a worse performance index, 2.16, but it also had a lower importance, 3, so it would be chosen as a receiver after TSO period 2 which had a higher importance and was not meeting goals.
- *pa\_pro\_rdon\_cand* indicates that TSO period 1 has been selected as the processor donor to be assessed.  
SRM selected TSO period 1 as the first donor candidate because it had the best performance index of all service class periods that were running at a dispatching priority higher than or equal to the receiver's dispatching priority.
- *pa\_pmuo\_rec* indicates that the first dispatching priority move to be assessed is to move TSO period 2 up to a higher dispatching priority.
- The next several *pa\_pro\_unc\_sec\_don* trace entries show the other service class periods whose processor access will be affected by the move up of TSO period 2 even though their priority remains unchanged. These service class periods were all secondary donors. TSO period 1 was the primary donor. This is indicated by the *pa\_pro\_unc\_don* action.
- *pa\_pro\_incp\_rec* shows that the primary receiver, TSO period 2, received a dispatching priority increase.

The sysplex and local performance index projections are shown with the final action for each service class period affected.

## MPL Policy Example

This example shows how to use subtype 2 service class period data to analyze a first period TSO problem. The example shows how SRM used the data to resolve the problem. You can use the data in the same way to analyze why SRM isn't solving a problem. From this kind of information, a service administrator can decide whether to change goals or service class importance in the service policy.

This subtype 2 data shows the controls SRM is using for first period TSO, and the resulting performance delays.

Subtype 2 Service Class TSO Period Data												
CLASS: TSO			PERIOD: 1			IMPORTANCE: 2			GOAL TYPE: SHORT RESPONSE TIME			
TIME	SPI	LPI	DP	MPLI	MPLO	SWPT	PSI	EXP	POL	LARGEST DELAYS		
13:15:01	.10	.09	251	3	6	15952	0	S	L	S	S	OTHR/60 ASWP/55 MPLD/2
13:15:11	6.18	6.73	251	6	9	15952	0	S	L	S	S	ASWP/48 CPUD/44 MPLD/42
13:15:22	2.20	2.42	251	8	12	15952	0	S	L	S	S	CPUD/38 MPLD/34 ASWP/2
13:15:32	.06	.06	251	8	12	15952	0	S	L	S	S	OTHR/13 CPUD/3 MPLD/2
13:15:42	.06	.06	251	8	12	15952	0	S	L	S	S	OTHR/22 MPLD/3 CPUD/2
13:15:52	.16	.16	251	8	12	15952	0	S	L	S	S	OTHR/28 MPLD/1
13:16:03	.16	.16	251	8	12	15952	0	S	L	S	S	OTHR/53 ASWP/6 CPUD/2

At 13:15:01 TSO period 1 was meeting its goals easily. This is indicated by a sysplex performance index (SPI) of 0.1 and a local performance index (LPI) of 0.09. The dispatching priority (DP) was 251. The MPL in and out targets (MPLI/MPLO) were 3 and 6. After being swapped out, work in the period was protected in processor storage for 15.952 seconds (SWPT=15952). There was no period wide storage isolation (PSI=0) and the expanded policy was space available for swap working set, VIO, and hiperspace pages, and LRU for demand pages (EXP POL= S L S S). At 13:15:01 TSO period 1 has a swap delay and an MPL delay but was meeting its goals easily.

Conditions change between 13:15:01 and 13:15:11. The sysplex performance index spiked to 6.18 and the local performance index was worse. This period needs help. The delay samples show that the problem could be either swap delay, processor delay, or MPL delay. Since it is the work furthest from meeting its goals, it is selected as a receiver.

The following data, from the subtype 1 record at 13:15:11 shows what happens next. The fields are explained in the first example.

#### Subtype 1 Trace Data Output

CLASS	P	SPI	LPI	ACTION
TSO	1	6.18	6.73	270 pa_rec_cand
TSO	1	6.18	6.73	2540 pa_prt_na_rec_val
TSO	1	6.18	6.73	850 pa_pro_na_no_donor
				290 pa_use_disc_cent
TSO	1	4.69	5.10	3530 pa_inc_mpl

TSO period 1 is selected as the receiver candidate. The trace entry, *pa\_rec\_cand* indicates this.

The largest delay is selected to be worked on first. In this case the largest delay in recent history is swap delay. The swap delay plot is shown below. *ccc* indicates the current plot point. The current plot point shows that only 89 milliseconds of swap delay per transaction could be eliminated even if all swap delay were eliminated. The *pa\_prt\_na\_rec\_val* trace entry indicates that there was insufficient receiver value to be gained by increasing the swap protect time. The swap delay plot data is from the subtype 3 record.

#### Subtype 3: Swap Delay Plot

CLASS: TSO		PERIOD: 1						
SWAP DELAY PLOT		ccc						
SWAP DELAY		286	209	89	76	70	24	0
TIME IN PROC STOR		3572	4021	14407	23673	40803	71248	97323

The next largest delay is processor delay. However TSO period 1 is running alone at the highest dispatching priority in use. Therefore there is no work to donate processor time. This reason for lack of action is indicated by the *pa\_pro\_na\_no\_donor* trace. The dispatch priorities are from the subtype 2 records.

### Subtype 2 Dispatching Priority Data

CLASS	P	DP
NRBATCH	1	247
NRBATCH	2	243
NRBATCH	3	192
OFFBATCH	1	192
TSO	1	251
TSO	2	247
TSO	3	247
TSO	4	243
\$SRMDISC	1	192

The third largest delay is MPL delay. The MPL delay plot below shows that there is value to increasing the MPL. The third entry in the MPL plot, indicated by *ccc* shows that on average, only 48/100ths of the ready users have MPL slots. This results in an MPL delay of 202 milliseconds per completion. This plot is recorded in the subtype 3 record. The fact that MPL targets were increased is indicated by the *pa\_inc\_mpl* trace. The new MPL targets are recorded in the subtype 2 record.

### MPL Delay Plot

CLASS: TSO	PERIOD: 1									
MPL DELAY PLOT				ccc						
MPL DELAY	500	451	202	79	0	0	0	0	0	0
MPL SLOT PERCENTAGE	3	37	48	72	99	109	115	122	129	133

At 13:15:22 the performance index was improving but there was still significant MPL delay and the MPL targets were increased again. At 13:15:32 the work was back to meeting its goals as shown by sysplex and local performance indexes of less than 1.0.



---

## Part 2. Reference: Workload Management Services





---

## Chapter 12. IWMCLSFY – Classify Work

The purpose of this service is to factor in available information about an arriving work request in order to associate a service class and possibly a report class with it.

---

### Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Either problem state or supervisor state. PSW key must either be 0 or match the value supplied on IWM4CON.
<b>Dispatchable unit mode:</b>	Task or SRB
<b>Cross memory mode:</b>	Any PASN, any HASN, any SASN
<b>AMODE:</b>	31-bit
<b>ASC mode:</b>	Primary
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	Unlocked or locked.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

---

### Programming Requirements

1. The macro IWMYCON must be included to use this macro.
2. Caller is responsible for error recovery.
3. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
4. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, IWMRSNCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.
5. All character data, unless otherwise specified, is assumed to be left justified and padded with blanks on the right, as needed, to occupy the specified number of bytes.

---

### Restrictions

1. This macro may only be used on MVS/ESA (version 3 or higher), NOT versions 1 or 2 of MVS.
2. FRRs are allowed
3. This macro may not be used during task/address space termination for the connect owner.
4. If the key specified on IWM4CON was a user key (8-F), then current primary must equal primary at the time that IWM4CON was invoked.
5. Only limited checking is done of the connect token obtained from IWM4CON.
6. SOURCELU is mutually exclusive with NETID/LUNAME.
7. This macro supports multiple versions. Some keywords are only supported by certain versions. Refer to the PLISTVER parameter description for further information.

## Input Register Information

Before issuing the IWMCLSFY macro, the caller must ensure that the following general purpose registers (GPRs) contain the specified information:

Register	Contents
13	The address of a 72-byte standard save area in the primary address space

## Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Reason code if GR15 return code is non-zero
1	Used as work registers by the system
2-13	Unchanged
14	Used as work registers by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

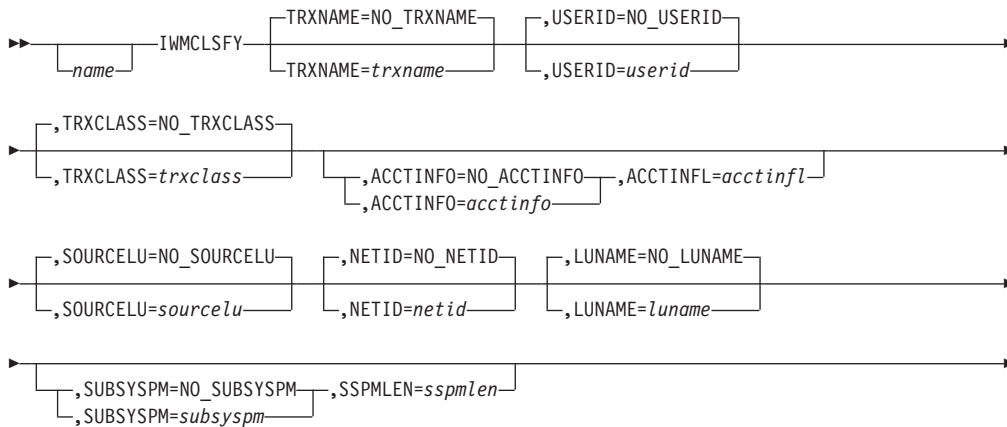
Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

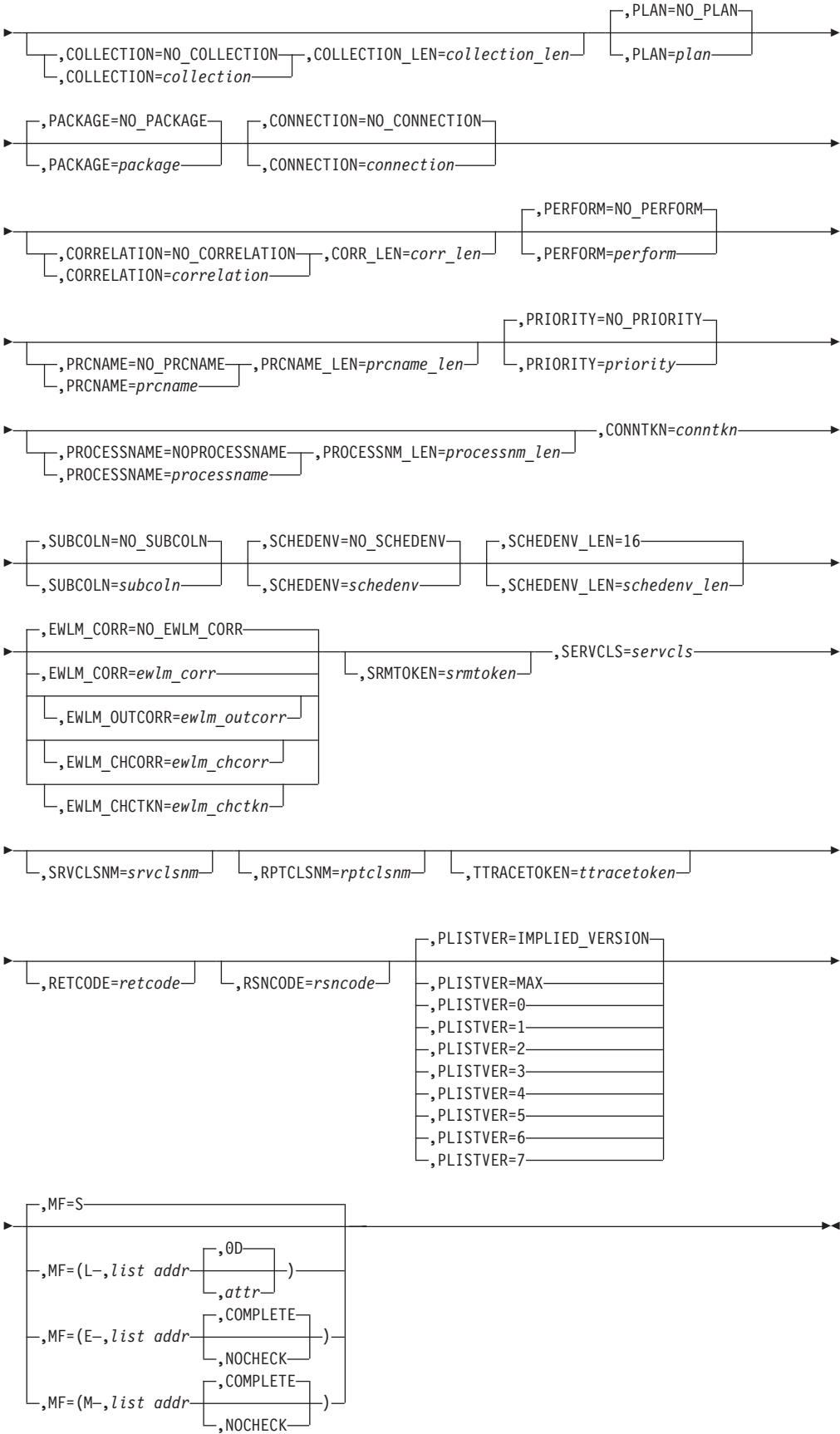
## Performance Implications

None.

## Syntax

### main diagram





---

## Parameters

The parameters are explained as follows:

*name*

An optional symbol, starting in column 1, that is the name on the IWMCLSFY macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

**,ACCTINFL=*acctinfl***

When ACCTINFO=*acctinfo* is specified, a required input parameter, which contains the length of the accounting information field. The maximum value supported is 143.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

**,ACCTINFO=*acctinfo***

**,ACCTINFO=NO\_ACCTINFO**

An optional input parameter, which contains the accounting information. For environments where accounting information is available on some, but not all flows, provision of a data area initialized to all blanks is equivalent to specification of NO\_ACCTINFO. The default is NO\_ACCTINFO, which indicates that no accounting information was passed.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a character field.

**,COLLECTION=*collection***

**,COLLECTION=NO\_COLLECTION**

An optional input parameter, which contains the customer defined name for a group of associated packages. For environments where the collection name may be available on some, but not all flows, provision of a data area initialized to all blanks is equivalent to specification of NO\_COLLECTION. The default is NO\_COLLECTION. indicates that no COLLECTION name is passed.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a character field.

**,COLLECTION\_LEN=*collection\_len***

When COLLECTION=*collection* is specified, a required input parameter, which contains the length of the collection name. There is no restriction on the length of data passed, but all storage between the start and end must be allocated (getmaind).

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

**,CONNECTION=*connection***

**,CONNECTION=NO\_CONNECTION**

An optional input parameter, which contains the name associated with the environment creating the work request, which may reside anywhere within the network. For environments where the connection name may be available on some, but not all flows, provision of a data area initialized to all blanks is equivalent to specification of NO\_CONNECTION. The default is NO\_CONNECTION. indicates that no CONNECTION name is passed.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

**,CONNTKN=conntkn**

A required input parameter, which is returned by IWM4CON for use by the classify routine.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32 bit field.

**,CORR\_LEN=corr\_len**

When CORRELATION=*correlation* is specified, a required input parameter, which contains the length of the correlation identifier. There is no restriction on the length of data passed, but all storage between the start and end must be allocated (getmained).

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

**,CORRELATION=correlation****,CORRELATION=NO\_CORRELATION**

An optional input parameter, which contains the name associated with the user/program creating the work request, which may reside anywhere within the network. For environments where the correlation name may be available on some, but not all flows, provision of a data area initialized to all blanks is equivalent to specifying NO\_CORRELATION. The default is NO\_CORRELATION. This indicates that no CORRELATION name is passed.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a character field.

**,EWLM\_CHCORR=ewlm\_chcorr**

An optional output parameter, which contains the cross-platform Enterprise Workload Management (EWLM) child correlator associated with the instantiated sub work request. Specifying this parameter indicates that a sub work request will be created.

**Notes:**

1. z/OS V1R6 only uses the first 64 bytes of the EWLM\_CHCORR field.
2. The EWLM\_OUTCORR, EWLM\_CHCORR and EWLM\_CHCTKN parameters are mutually exclusive.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 512-character field.

**,EWLM\_CHCTKN=ewlm\_chctkn**

An optional output parameter, which contains the cross-platform Enterprise Workload Management (EWLM) child correlator token associated with the instantiated sub work request. Specifying this parameter indicates that a sub work request will be created. An EWLM child correlator token must not be passed outside of the EWLM management domain.

The EWLM\_OUTCORR, EWLM\_CHCORR and EWLM\_CHCTKN parameters are all mutually exclusive.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 64-character field.

**,EWLM\_CORR=ewlm\_corr****,EWLM\_CORR=NO\_EWLM\_CORR**

An optional input parameter, which contains the cross-platform Enterprise Workload Management (EWLM) correlator associated with the work request. If this parameter is specified and a valid EWLM correlator is passed, the EWLM

transaction class can be used for WLM classification purposes. The default is NO\_EWLM\_CORR which indicates that no EWLM correlator is passed.

The EWLM correlator also serves as the input correlator for the EWLM\_CHCORR, EWLM\_CHTKN, EWLM\_OUTCORR parameters.

**Notes:**

1. The architected length field of an ARM correlator in the first two bytes must contain a value between 4 ('0004'X) and 512 ('0200'X).
2. For environments where the EWLM correlator may be available on some, but not all flows, providing a data area with the first four bytes set to binary zeroes is equivalent to specifying NO\_EWLM\_CORR.
3. The EWLM\_OUTCORR, EWLM\_CHCORR and EWLM\_CHCTKN parameters are mutually exclusive.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a character field.

| **,EWLM\_OUTCORR=ewlm\_outcorr**

| An optional output parameter which receives a validated EWLM correlator on  
| return. The execution form of IWMCLSFY validates the passed correlator in  
| EWLM\_CORR and provides a valid EWLM correlator in EWLM\_OUTCORR as  
| follows:

- If the EWLM\_CORR parameter is specified and the correlator in EWLM\_CORR is a valid ARM correlator in EWLM format, it will be copied to EWLM\_OUTCORR unchanged.
- If the correlator in EWLM\_CORR is not a valid EWLM ARM correlator or the EWLM\_CORR parameter is omitted, a new classify correlator will be returned within the EWLM\_OUTCORR field.

| **Notes:**

1. Specifying EWLM\_OUTCORR (unlike EWLM\_CHCORR or EWLM\_CHCTKN) does not indicate the beginning of a sub work request.
2. The application may specify the same parameter for both EWLM\_CORR and EWLM\_OUTCORR, which means that the EWLM correlator can be validated or replaced in place.
3. The EWLM\_OUTCORR, EWLM\_CHCORR and EWLM\_CHCTKN parameters are mutually exclusive.

| **To code:** Specify the RS-type address, or address in register (2)-(12), of a  
| 512-character field.

| **,LUNAME=luname**

| **,LUNAME=NO\_LUNAME**

| An optional input parameter, which contains the local LU name associated with  
| the requestor. For environments where the local LU name may be available on  
| some, but not all flows, provision of a data area initialized to all blanks is  
| equivalent to specification of NO\_LUNAME.

| SOURCELU is mutually exclusive with LUNAME. The default is NO\_LUNAME.  
| indicates that no local LU name is passed.

| **To code:** Specify the RS-type address, or address in register (2)-(12), of an  
| 8-character field.

| **,MF=S**

| **,MF=(L,list addr)**

| **,MF=(L,list addr,attr)**

**,MF=(L,list addr,0D)**  
**,MF=(E,list addr)**  
**,MF=(E,list addr,COMPLETE)**  
**,MF=(E,list addr,NOCHECK)**  
**,MF=(M,list addr)**  
**,MF=(M,list addr,COMPLETE)**  
**,MF=(M,list addr,NOCHECK)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

Use MF=M together with the list and execute forms of the macro for service routines that need to provide different options according to user-provided input. Use the list form to define a storage area; use the modify form to set the appropriate options; then use the execute form to call the service.

IBM recommends that you use the modify and execute forms of IWMCLSFY in the following order:

- Use IWMCLSFY ...MF=(M,list-addr,COMPLETE) specifying appropriate parameters, including all required ones.
- Use IWMCLSFY ...MF=(M,list-addr,NOCHECK), specifying the parameters that you want to change.
- Use IWMCLSFY ...MF=(E,list-addr,NOCHECK), to execute the macro.

**,list addr**

The name of a storage area to contain the parameters. For MF=S, MF=E, and MF=M, this can be an RS-type address or an address in register (1)-(12).

**,attr**

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

**,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

**,NOCHECK**

Specifies that the system is not to check for required parameters and is not to supply defaults for omitted optional parameters.

**,NETID=netid**

**,NETID=NO\_NETID**

An optional input parameter, which contains the network identifier associated

with the requestor. For environments where the network identifier may be available on some, but not all flows, provision of a data area initialized to all blanks is equivalent to specification of NO\_NETID.

SOURCELU is mutually exclusive with NETID. The default is NO\_NETID. indicates that no network identifier is passed.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

**,PACKAGE=***package*

**,PACKAGE=NO\_PACKAGE**

An optional input parameter, which contains the package name for a set of associated SQL statements. Products using this attribute must chose a specific package name to be associated with the work request, e.g. the first package name used in the unit of work. Individual product documentation will describe how this choice is made to allow the installation to use the WLM administrative application. For environments where the package name may be available on some, but not all flows, provision of a data area initialized to all blanks is equivalent to specification of NO\_PACKAGE. The default is NO\_PACKAGE. indicates that no PACKAGE name is passed.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

**,PERFORM=***perform*

**,PERFORM=NO\_PERFORM**

An optional input parameter, which contains the performance group number (PGN) associated with the work request. If specified, the performance group number value must be within the range of 1-999, represented as character data, left justified and padded with blanks on the right. For environments where the perform value may be available on some, but not all flows, provision of a data area initialized to all blanks is equivalent to specification of NO\_PERFORM. The default is NO\_PERFORM. indicates that no PERFORM value is passed.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

**,PLAN=***plan*

**,PLAN=NO\_PLAN**

An optional input parameter, which contains the access plan name for a set of associated SQL statements. For environments where the plan name may be available on some, but not all flows, provision of a data area initialized to all blanks is equivalent to specification of NO\_PLAN. The default is NO\_PLAN. indicates that no PLAN name is passed.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

**,PLISTVER=IMPLIED\_VERSION**

**,PLISTVER=MAX**

**,PLISTVER=0**

**,PLISTVER=1**

**,PLISTVER=2**

**,PLISTVER=3**

**,PLISTVER=4**

**,PLISTVER=5**

**,PLISTVER=6**

**,PLISTVER=7**

An optional input parameter that specifies the version of the macro. PLISTVER



determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form; in this way, MAX ensures that the parameter list does not overwrite nearby storage.

- **0**, which supports all parameters except those specifically referenced in higher versions.
- **1**, which supports the following parameters along with those from version 0:

PERFORM                      PRCNAME                      PRCNAME\_LEN

- **2**, which supports the following parameters along with those from version 0 and 1:

PRIORITY

- **3**, which supports the following parameters along with those from version 0, 1 and 2:

PROCESSNAME                  PROCESSNM\_LEN

- **4**, which supports the following parameters along with those from version 0, 1, 2 and 3:

TTRACETOKEN

- **5**, which supports both the following parameters and those from version 0, 1, 2, 3, and 4:

SCHEDENV                      SCHEDENV\_LEN                  SRMTOKEN  
SUBCOLN

- **6**, which supports both the following parameters and those from version 0, 1, 2, 3, 4, and 5:

EWLM\_CORR

- **7**, which supports both the following parameters and those from version 0, 1, 2, 3, 4, 5, and 6:

EWLM\_CHCORR                  EWLM\_CHCTKN                  EWLM\_OUTCORR

**To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX
- A decimal value of 0, 1, 2, 3, 4, 5, 6, or 7

**,PRCNAME=*prcname***

**,PRCNAME=NO\_PRCNAME**

An optional input parameter, which contains the DB2 Stored SQL Procedure name associated with the work request. For environments where the SQL procedure name may be available on some, but not all flows, provision of a data area initialized to all blanks is equivalent to specification of NO\_PRCNAME. The default is NO\_PRCNAME, which indicates that no PRCNAME value is passed.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 18-character field.

**,PRCNAME\_LEN=*prcname\_len***

When PRCNAME=*prcname* is specified, a required input parameter, which contains the length of the procedure name. There is no restriction on the length of data passed, but all storage between the start and end must be allocated (getmained).

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

**,PRIORITY=*priority***

**,PRIORITY=NO\_PRIORITY**

An optional input parameter, which contains the priority associated with the work request. For environments where the priority value may be available on some, but not all flows, provision of a data area initialized to hexadecimal 80000000 (the largest negative integer) is equivalent to specification of NO\_PRIORITY. The default is NO\_PRIORITY.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

**,PROCESSNAME=*processname***

**,PROCESSNAME=NOPROCESSNAME**

An optional input parameter, which contains the process name associated with the work request. The default is NOPROCESSNAME. indicates that no PROCESSNAME value is passed.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32-character field.

**,PROCESSNM\_LEN=*processnm\_len***

When PROCESSNAME=*processname* is specified, a required input parameter, which contains the length of the process name. There is no restriction on the length of data passed, but all storage between the start and end must be allocated (getmained).

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

**,RETCODE=*retcode***

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,RPTCLSNM=*rptclsnm***

An optional output parameter, which is to receive the output report class name.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

**,RSNCODE=*rsncode***

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,SCHEDENV=*schedenv*****,SCHEDENV=NO\_SCHEDENV**

An optional input parameter, which contains the scheduling environment value associated with the work request. The default is NO\_SCHEDENV. indicates that no scheduling environment value is passed.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 16-character field.

**,SCHEDENV\_LEN=*schedenv\_len*****,SCHEDENV\_LEN=16**

When SCHEDENV=*schedenv* is specified, an optional input parameter, which contains the length of the scheduling environment. There is no restriction on the length of data passed, but all storage between the start and end must be allocated (getmained). The default is 16.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

**,SERVCLS=*servcls***

A required output parameter, which is to receive the output token which represents the service and report class for the work request.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32 bit field.

**,SOURCELU=*sourcelu*****,SOURCELU=NO\_SOURCELU**

An optional input parameter, which contains the LU name associated with the requestor. This may be the fully qualified NETID.LUNAME, e.g. network name (1-8 bytes), followed by a period, followed by the LU name for the requestor (1-8 bytes). It may also be the 1-8 byte local LU name, with no network qualifier. The SOURCELU field may be from 1-17 characters. In the assembler form, the macro determines the length of the field as follows:

1. If the field is specified by register notation, it will be assumed to be 17 characters (padded with blanks).
2. If the field is specified using an RS form name, then the length will be determined using the L' assembler function.

In the PL/AS form, the rules for the PL/AS compiler will determine the length. The product using IWMCLSFY is responsible for documenting which form is used so that the customer may specify the correct format.

For environments where the LU name may be available on some, but not all flows, provision of a data area initialized to all blanks is equivalent to specification of NO\_SOURCELU.

SOURCELU is mutually exclusive with NETID/LUNAME. The default is NO\_SOURCELU. indicates that no source LU name was passed.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a character field.

**,SRMTOKEN=*srmtoken***

An optional output parameter, token for SRM internal use only.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32 bit field.

**,SRVCLSNM=***srvclsnm*

An optional output parameter, which is to receive the output service class name.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

**,SSPMLEN=***sspmlen*

When SUBSYSPM=*subsyspm* is specified, a required input parameter, which contains the length of the data passed by the work manager. There is no restriction on the length of data passed, but all storage between the start and end must be allocated (getmained).

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

**,SUBCOLN=***subcoln*

**,SUBCOLN=NO\_SUBCOLN**

An optional input parameter, which contains the subsystem collection name associated with the work request. A subsystem collection is a named group of related subsystem address spaces — for instance, the XCF group name of JES2 and JES3 work. The default is NO\_SUBCOLN, indicating that no subsystem collection name is passed.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

**,SUBSYSPM=***subsyspm*

**,SUBSYSPM=NO\_SUBSYSPM**

An optional input parameter, which contains character data related to the work request which is passed by the work manager for use in classification. The nature of the contents of this data must be documented for customer use. For environments where the subsystem parameter is available on some, but not all flows, provision of a data area initialized to all blanks is equivalent to specification of NO\_SUBSYSPM. The default is NO\_SUBSYSPM. indicates that no parameter was passed.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a character field.

**,TRXCLASS=***trxclass*

**,TRXCLASS=NO\_TRXCLASS**

An optional input parameter, which contains a class name within the subsystem. This can be any meaningful value that the installation can recognize and specify to match the value presented by the work manager. For environments where the transaction class is available on some, but not all flows, provision of a data area initialized to all blanks is equivalent to specification of NO\_TRXCLASS. The default is NO\_TRXCLASS. indicates that no transaction class was passed.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

**TRXNAME=***trxname*

**TRXNAME=NO\_TRXNAME**

An optional input parameter, which contains the transaction name for the work request, as known by the work manager. For environments where the transaction name is available on some, but not all flows, provision of a data area initialized to all blanks is equivalent to specification of NO\_TRXNAME. The default is NO\_TRXNAME. indicates that no transaction name is passed.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

**,TTRACETOKEN=*ttracetoken***

An optional output parameter, which is to receive the output transaction trace token associated with the work request.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32-character field.

**,USERID=*userid***

**,USERID=NO\_USERID**

An optional input parameter, which contains the userid associated with the work request. For environments where the user id is available on some, but not all flows, provision of a data area initialized to all blanks is equivalent to specification of NO\_USERID. The default is NO\_USERID. indicates that no userid is passed.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

---

## ABEND Codes

None.

---

## Return and Reason Codes

When the IWMCLSFY macro returns control to your program:

- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, if you coded RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 22. Return and Reason Codes for the IWMCLSFY Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> <code>IwmRetCodeOk</code> <b>Meaning:</b> Successful completion. <b>Action:</b> None required.
4	—	<b>Equate Symbol:</b> <code>IwmRetCodeWarning</code> <b>Meaning:</b> Successful completion, unusual conditions noted.
4	xxxx0409	<b>Equate Symbol:</b> <code>IwmRsnCodeNoConn</code> <b>Meaning:</b> Connect token does not reflect a successful Connect. <b>Action:</b> None required.
8	—	<b>Equate Symbol:</b> <code>IwmRetCodeInvocError</code> <b>Meaning:</b> Invalid invocation environment or parameters.
8	xxxx0821	<b>Equate Symbol:</b> <code>IwmRsnCodeBadConn</code> <b>Meaning:</b> Input connect token does not pass validity checking. <b>Action:</b> Check for possible storage overlay.

## IWMCLSFY Macro

Table 22. Return and Reason Codes for the IWMCLSFY Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0840	<p><b>Equate Symbol:</b> IwmRsnCodeServiceNotEnabled</p> <p><b>Meaning:</b> Caller's space connection is not enabled for this service.</p> <p><b>Action:</b> Avoid requesting this function under the input connection. IWM4CON options must be specified previously to enable this service.</p>
8	xxxx0894	<p><b>Equate Symbol:</b> IwmRsnCodeInvalidEWLMCorr</p> <p><b>Meaning:</b> The classification information contains an EWLM correlator (EWLM_CORR) that does not pass validity checking. The architected ARM correlator length field in the first two Bytes of the EWLM_CORR is either less than 4 ('0004'x) or greater than 512 ('0200'x).</p> <p><b>Action:</b> Check the specification of the EWLM correlator in the classification information.</p>
8	xxxx0895	<p><b>Equate Symbol:</b> IwmRsnCodeEWLMServNotEnabled</p> <p><b>Meaning:</b> Service is not enabled because caller invoked the IWMCONN service with EWLM=NO.</p> <p><b>Action:</b> Specify the parameter EWLM_CORR only when connected with EWLM=YES.</p>
C	—	<p><b>Equate Symbol:</b> IwmRetCodeEnvError</p> <p><b>Meaning:</b> Environmental error.</p>
C	xxxx0C01	<p><b>Equate Symbol:</b> IwmRsnCodeNoStg</p> <p><b>Meaning:</b> Storage is not available for the request.</p> <p><b>Action:</b> There is a storage shortage. The function may work successfully at a later time.</p>
10	—	<p><b>Equate Symbol:</b> IwmRetCodeCompError</p> <p><b>Meaning:</b> Component error.</p> <p><b>Action:</b> No action required. The function may be successful if invoked again.</p>

---

## Examples

Suppose the transactions processed by a subsystem work manager have the following qualifiers:

- Userid
- Transaction name
- Transaction class

To get the service class associated with an incoming work request, specify:

```
IWMCLSFY USERID=AUSERID,TRXCLASS=ATRXCLS,TRXNAME=ATRNXNM,  
          CONNTKN=(R7),SERVCLS=(R9),  
          RETCODE=RETCODE,RSNCODE=RSNCODE
```

Where the following are declared:

```
AUSERID DS CL8  
ATRXCLS DS CL8  
ATRNXNM DS CL8
```





---

## Chapter 13. IWMCNTN — WLM Contention Notification

The IWMCNTN service allows resource managers to notify WLM of changes to the list of resources, work units, or transactions involved with resources that have been in contention (waiters exist) for longer than a resource manager defined interval. The interval should be chosen so that only contention which has lasted long enough to be considered chronic for the issuing resource manager results in calling this service.

The caller can run in task or SRB mode.

---

### Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Supervisor state or program key mask (PKM) allowing keys 0-7.
<b>Dispatchable unit mode:</b>	Task or SRB
<b>Cross memory mode:</b>	Any PASN, any HASN, any SASN
<b>AMODE:</b>	31-bit
<b>ASC mode:</b>	Primary
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks may be held.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

---

### Programming Requirements

1. The macro CVT must be included to use this macro.
2. The macro IWMYCON must be included to use this macro.
3. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
4. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, IWMRSCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.
5. All character data, unless otherwise specified, is assumed to be left justified and padded with blanks on the right, as needed, to occupy the specified number of bytes.
6. Since this service may only be used by system-like code, some validity checking on the parameter list is not performed. These checks would only be needed if the macro were not used to invoke the service routine.

---

### Restrictions

None.

---

### Input Register Information

Before issuing the IWMCNTN macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

## Output Register Information

When control returns to the caller, the GPRs contain:

### Register

#### Contents

- 0** Reason code if GR15 return code is non-zero
- 1** Used as work registers by the system
- 2-13** Unchanged
- 14** Used as work registers by the system
- 15** Return code

When control returns to the caller, the ARs contain:

### Register

#### Contents

- 0-1** Used as work registers by the system
- 2-13** Unchanged
- 14-15** Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

## Performance Implications

None.

## Syntax

### main diagram



## Parameters

The parameters are explained as follows:

*name*

An optional symbol, starting in column 1, that is the name on the IWMCNTN macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

**,INVOCATIONTYPE=UPDATE**

**,INVOCATIONTYPE=REPLACE**

**,INVOCATIONTYPE=ENDOFCONTENTION**

A required parameter, which indicates the type of operation requested

**,INVOCATIONTYPE=UPDATE**

indicates that the operations described in the request list have to be applied to the resource. If contention information for the resource does not already exist, it is created. If after applying the operations there are no holders or waiters (local or remote), tracking of the resource is abandoned locally.

**,INVOCATIONTYPE=REPLACE**

same as UPDATE, except that any existing local resource topology is discarded first.

**,INVOCATIONTYPE=ENDOFCONTENTION**

indicates that all topology information for the resource is discarded.

**,MF=S**

**,MF=(L,list addr)**

**,MF=(L,list addr,attr)**

**,MF=(L,list addr,OD)**

**,MF=(E,list addr)**

**,MF=(E,list addr,COMPLETE)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

*,list addr*

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

*,attr*

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

**,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

**,PLISTVER=IMPLIED\_VERSION**

**,PLISTVER=MAX**

**,PLISTVER=0**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form, when both are assembled with the same level of the system. In this way, MAX ensures that the parameter list does not overwrite nearby storage.

- **0**, if you use the currently available parameters.

**To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX
- A decimal value of 0

**,REQUESTLIST=requestlist**

When INVOCATIONTYPE=UPDATE or REPLACE is specified, a required input parameter, which specifies a list of topology requests. For each request, you must specify:

- Whether you want to add or to delete the contention information
- Whether the entity (address space, enclave, or work unit) is holding the resource or is waiting for it,
- The identification of the entity in terms of STOKEN, TCB address or enclave token.

The macro returns for each request a return and reason code. See IWMCNTRL for the mapping of the request list.

The work unit id varies based on the resource ownership model (transaction or task resource) used by the IWMCNTN exploiter and the work unit type involved, as described in the table below.

Work unit type	Exploiters using the transaction model will pass	Exploiters using the task resource model will pass
Global SRB, local SRB, preemptible SRB (but not a client or enclave SRB)	Home STOKEN, TCB=0, Etoken=0	same as transaction resource ownership model

Work unit type	Exploiters using the transaction model will pass	Exploiters using the task resource model will pass
Client SRB	Client STOKEN TCB=0, Etoken=0	same as transaction resource ownership model
Enclave SRB	STOKEN=0, TCB=0, Etoken=e	same as transaction resource ownership model
Non-enclave task	Home STOKEN, TCB= 0, Etoken=0	Home STOKEN, TCB= t, Etoken=0
Enclave task	STOKEN=0, TCB= 0, Etoken=e	Home STOKEN, TCB= t, Etoken=0

The following return/reason codes may be returned per request:

**Return\_Code** A 2-byte output field set based on whether or not the entity identification information passed validity checks.

- 0** Name: `IwmRetCodeOk`  
Meaning: Successful completion.  
Action: None required.
- 8** Name: `IwmRetCodeInvocError`  
Meaning: Invalid invocation environment or parameters.  
Action: Check reason code

**Reason\_Code**

A 2-byte output field set based on whether or not the entity identification information passed validity checks.

- 0807** Name: `IwmRsnCodeBadSTOKEN`  
Meaning: The specified STOKEN does not pass verification.  
Action: Check for possible storage overlay of the address space token.
- 083A** Name: `IwmRsnCodeBadEnclave`  
Meaning: Enclave token does not pass verification.  
Action: Check for possible storage overlay of the enclave token.
- 0886** Name: `IwmRsnCodeBadRequestCode`  
Meaning: The request code must be either ADD or DELETE  
Action: Correct the request code.
- 0887** Name: `IwmRsnCodeBadEntityType`  
Meaning: The entity type must be either HOLDER or WAITER  
Action: Correct the entity type.
- 088A** Name: `IwmRsnCodeBadEntityId`

Meaning: The specified combination of STOKEN, TCB and/or enclave token does not pass verification.  
Action: Correct the entity id.

**088B** Name: IwmRsnCodeBadTCB  
Meaning: The specified TCB address does not pass verification.  
Action: Correct the TCB address. Task may have terminated since the parameter list was built. TCB may not match STOKEN.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a character field.

**,RESOURCEID=resourceid**

A required input parameter, which identifies the resource uniquely within all resources for a subsystem type and name.

For resources whose type is multisystem, the value must be unique within the subsystem type and name across all systems where the interface might ever be invoked for this resource.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a character field.

**,RESOURCEID\_LEN=resourceid\_len**

A required input parameter, which contains the length of the resource identifier. A resource identifier may not exceed 264 bytes.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

**,RESOURCESCOPE=SINGLESYSTEM**

**,RESOURCESCOPE=MULTISYSTEM**

An optional parameter, which identifies if the resource information is shared with other WLM instances in the cluster. The default is RESOURCESCOPE=SINGLESYSTEM.

**,RESOURCESCOPE=SINGLESYSTEM**

indicates that the resource information is used on the issuing system only.

**,RESOURCESCOPE=MULTISYSTEM**

indicates that the resource information is shared among other systems.

**,RETCODE=retcode**

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,RSNCODE=rsncode**

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**SUBSYS=subsys**

A required input parameter, which contains the generic subsystem type (e.g. IMS, CICS, etc.).

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 4-character field.

**,SUBSYSNM=***subsysnm*

A required input parameter, which identifies the subsystem instance.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

---

## ABEND Codes

None.

---

## Return and Reason Codes

When the IWMCNTN macro returns control to your program:

- GPR 15 (and *retcode*, when you code RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, when you code RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 23. Return and Reason Codes for the IWMCNTN Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> <code>IwmRetCodeOk</code> <b>Meaning:</b> Successful completion. <b>Action:</b> None required.
8	—	<b>Equate Symbol:</b> <code>IwmRetCodeInvocError</code> <b>Meaning:</b> Invalid invocation environment or parameters.
8	xxxx0803	<b>Equate Symbol:</b> <code>IwmRsnCodeDisabled</code> <b>Meaning:</b> Caller is disabled. <b>Action:</b> Avoid requesting this function while disabled.
8	xxxx0804	<b>Equate Symbol:</b> <code>IwmRsnCodeLocked</code> <b>Meaning:</b> Caller is locked. <b>Action:</b> Avoid requesting this function while locked.
8	xxxx080B	<b>Equate Symbol:</b> <code>IwmRsnCodeBadPI</code> <b>Meaning:</b> Error accessing parameter list. <b>Action:</b> Check for possible storage overlay.
8	xxxx0824	<b>Equate Symbol:</b> <code>IwmRsnCodeAmode24</code> <b>Meaning:</b> Caller invoked service but was in 24-bit addressing mode. <b>Action:</b> Request this function only when you are in 31-bit addressing mode.
8	xxxx0825	<b>Equate Symbol:</b> <code>IwmRsnCodeAscModeNotPrimary</code> <b>Meaning:</b> Caller invoked service but was not DAT on primary ASC mode. <b>Action:</b> Avoid requesting this function in this environment.

## IWMCNTN Macro

Table 23. Return and Reason Codes for the IWMCNTN Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0828	<b>Equate Symbol:</b> IwmRsnCodeBadVersion <b>Meaning:</b> Version number in parameter list is not valid. <b>Action:</b> Check for possible storage overlay of the parameter list.
8	xxxx0888	<b>Equate Symbol:</b> IwmRsnCodeBadRequestList <b>Meaning:</b> The request list does not pass verification. <b>Action:</b> Check the return and reason codes in the request list.
8	xxxx0889	<b>Equate Symbol:</b> IwmRsnCodeBadResourceIdLen <b>Meaning:</b> The length of the resource id must not exceed 264 bytes. <b>Action:</b> Specify a correct resource id length.
8	xxxx088C	<b>Equate Symbol:</b> IwmRsnCodeBadRequestListVersion <b>Meaning:</b> The version specified in the request list is not supported. <b>Action:</b> Specify a correct request list version.
8	xxxx088D	<b>Equate Symbol:</b> IwmRsnCodeBadRequestListLength <b>Meaning:</b> The specified request list length is too small to carry the specified number of request entries. <b>Action:</b> Specify a correct request list length or correct the entry count.
10	—	<b>Equate Symbol:</b> IwmRetCodeCompError <b>Meaning:</b> Component error. <b>Action:</b> No action required. The function may be successful if invoked again.

## Example

```

IWMCNTN SUBSYS=SUBSTYPE,SUBSYSNM=SUBSNAME,
        RESOURCEID=RESOURCE,RESOURCEID_LEN=128,
        INVOCATIONTYPE=REPLACE,
        REQUESTLIST=REQUESTS
*
* Storage areas
*
SUBSTYPE DS    CL4           Subsystem type
SUBSNAME DS    CL8           Subsystem name
RESOURCE DS    CL128        Resource id
REQUESTS DS    0D           Request list
EYE      DC    CL8'IWMCNTRL'
VERSION  DC    XL1'01'
RSRV_1  DS    CL3
LENGTH#  DC    F'96'
ENTRY#   DC    F'2'
RSRV_2   DS    CL12
ENTRY_1  DS    0CL32        1st entry
E#1_CODE DC    CL1'A'       code = add
E#1_TYPE DC    CL1'H'       type = holder
E#1_RSRV DS    CL6         reserved
E#1_STKN DC    XL8'0000000000000000' STOKEN not specified

```



E#1_TCB	DC	A'0'	TCB not specified
E#1_ETKN	DS	XL8	enclave token
E#1_RC	DS	H	entity return code
E#1_RSN	DS	H	entity reason code
ENTRY_2	DS	0CL32	2nd entry
E#2_CODE	DC	CL1'A'	code = add
E#2_TYPE	DC	CL1'W'	type = waiter
E#2_RSRV	DS	CL6	reserved
E#2_STKN	DS	XL8	STOKEN
E#2_TCB	DC	A'0'	TCB not specified
E#2_ETKN	DC	XL8'0000000000000000'	enclave token not specified
E#2_RC	DS	H	entity return code
E#2_RSN	DS	H	entity reason code



---

## Chapter 14. IWMCPAFN – WLM CPU Affinity Service

During the program execution, the program can establish processor affinity to certain CPUs. If all these CPUs are not currently online, the program will fail to execute. If the program cannot handle this failure, it should use IWMCPAFN service to enforce the CPU affinity. Prior to establishing CPU affinity, program should invoke the IWMCPAFN service to bring online any CPU that was taken offline by WLM and is required to be online to satisfy the CPU affinity requirement of the program. This service can not bring online any processor which is currently taken offline by operator.

The IWMCPAFN service can also be used to make sure that WLM CPU management processing keeps online at least one of the CPUs from the set of CPUs to which program is establishing CPU affinity.

WLM CPU management processing brings a CPU online and offline to provide consistent speed between the logical and physical CPUs. Use of this service restricts WLM from bringing a CPU offline when it is necessary to provide optimal CPU resources. So, when the program does not require the CPU affinity, the IWMCPAFN service should be invoked again to release the enforcement of the CPU affinity. This enables WLM to move the CPU offline to offer optimal number of online CPUs and to provide consistent speed between the logical and physical CPUs.

The caller can use the service in the following ways:

1. REQUEST = ENFORCE

If all of the CPUs specified on CPUMASK are currently offline, WLM brings online one of the CPUs which was taken offline by WLM previously. WLM keeps this CPU online until IWMCPAFN is invoked to release the CPU affinity enforcement or until the address space which issued this request ends. If all of the CPUs specified on CPUMASK are currently online, WLM makes sure that at least one of the CPUs stays online to satisfy the CPU affinity required by the program.

2. REQUEST = RELEASE

CPU affinity enforcement that was established earlier is now released. WLM CPU management processing is now enabled to bring any CPU offline provided there are no other jobs that require CPU affinity to the CPU.

---

### Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Supervisor state or program key mask (PKM) allowing keys 0-7.
<b>Dispatchable unit mode:</b>	Task
<b>Cross memory mode:</b>	PASN=HASN=SASN
<b>AMODE:</b>	31-bit
<b>ASC mode:</b>	Primary
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks may be held.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

---

## Programming Requirements

1. Make sure no EUT FRRs are established.
2. The macro CVT must be included to use this macro.
3. The macro IWMYCON must be included to use this macro.
4. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
5. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, IWMRSNCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.

---

## Restrictions

1. This macro may not be used during task/address space termination.

---

## Input Register Information

Before issuing the IWMCPAFN macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

---

## Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Reason code if GR15 return code is non-zero
1	Used as work registers by the system
2-13	Unchanged
14	Used as work registers by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

---

## Performance Implications

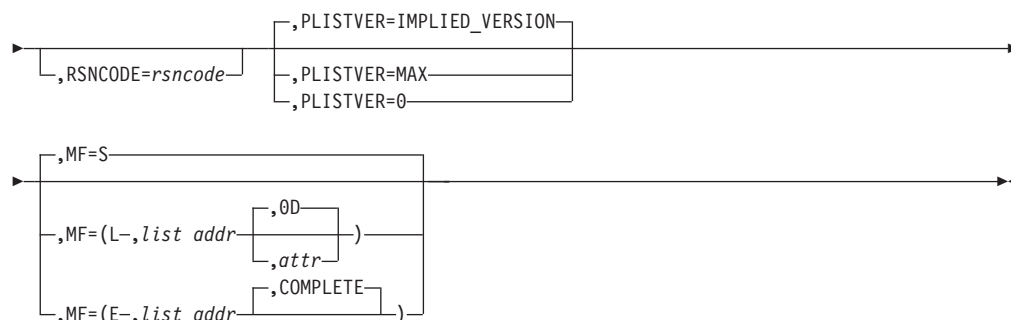
None.

---

## Syntax

### main diagram





## Parameters

The parameters are explained as follows:

### *name*

An optional symbol, starting in column 1, that is the name on the IWMCPAFN macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

### **,CPUMASK=cpumask**

When REQUEST=ENFORCE is specified, a required input parameter, CP mask which specifies the CPU affinity to be enforced. If all of the bits in the mask are zeroes or ones, then no affinity is enforced. Otherwise, the mask contains bits which represent processor affinity to be enforced. Bits that have a value of 1 (on) indicate that the program has affinity to the processor with the same physical address as the bit position. For example, if the 4th bit is on in the mask, the program will have affinity to the processor with physical address 3. To enforce the program's affinity to CP 0, CP 3, and CP 4, the bit mask would be B'1001 1000 0000 0000'. WLM will make sure that at least one of the CPUs is online. If all of the CPUs are offline, WLM brings online one of the CPUs that was previously taken offline by WLM and keeps it online. If all the CPUs are online, WLM makes sure that at least one of the CPUs remains online to satisfy the CPU affinity requirement of the program. WLM keeps the CPU online unless the IWMCPAFN request is invoked with REQUEST=RELEASE or the address space that issued this service terminates.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 16 bit field.

### **,MF=S**

**,MF=(L,list addr)**

**,MF=(L,list addr,attr)**

**,MF=(L,list addr,OD)**

**,MF=(E,list addr)**

**,MF=(E,list addr,COMPLETE)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

## IWMCPAFN Macro

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

*,list addr*

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

*,attr*

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

**,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

**,PLISTVER=IMPLIED\_VERSION**

**,PLISTVER=MAX**

**,PLISTVER=0**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form, when both are assembled with the same level of the system. In this way, MAX ensures that the parameter list does not overwrite nearby storage.

- **0**, if you use the currently available parameters.

**To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX
- A decimal value of 0

**REQUEST=ENFORCE**

**REQUEST=RELEASE**

A required parameter that is a required keyword input which indicates the request type.

**REQUEST=ENFORCE**

indicates that the caller wants to enforce the CPU affinity to the CPUs specified on the CPUMASK keyword.

**REQUEST=RELEASE**

indicates that the caller wants to release the enforcement of the CPU affinity which was established earlier. WLM CPU management processing

will be able to bring any CPU offline when necessary provided there are no other jobs that require CPU affinity to the CPU.

**,RETCODE=retcode**

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,RSNCODE=rsncode**

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

---

## ABEND Codes

None.

---

## Return and Reason Codes

When the IWMCPAFN macro returns control to your program:

- GPR 15 (and *retcode*, when you code RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, when you code RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 24. Return and Reason Codes for the IWMCPAFN Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> <code>IwmRetCodeOk</code> <b>Meaning:</b> Successful completion. <b>Action:</b> None required.
8	—	<b>Equate Symbol:</b> <code>IwmRetCodeInvocError</code> <b>Meaning:</b> Invalid invocation environment or parameters.
8	xxxx0801	<b>Equate Symbol:</b> <code>IwmRsnCodeSrbMode</code> <b>Meaning:</b> Caller is in SRB mode. <b>Action:</b> Avoid requesting this function while in SRB mode.
8	xxxx0803	<b>Equate Symbol:</b> <code>IwmRsnCodeDisabled</code> <b>Meaning:</b> Caller is disabled. <b>Action:</b> Avoid requesting this function while disabled.
8	xxxx0804	<b>Equate Symbol:</b> <code>IwmRsnCodeLocked</code> <b>Meaning:</b> Caller is locked. <b>Action:</b> Avoid requesting this function while locked.
8	xxxx080B	<b>Equate Symbol:</b> <code>IwmRsnCodeBadPI</code> <b>Meaning:</b> Error accessing parameter list. <b>Action:</b> Check for possible storage overlay.

Table 24. Return and Reason Codes for the IWMCPAFN Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0823	<b>Equate Symbol:</b> IwmRsnCodeDatoff <b>Meaning:</b> Caller invoked service while DATOFF <b>Action:</b> Avoid requesting this function in this environment.
8	xxxx0824	<b>Equate Symbol:</b> IwmRsnCodeAmode24 <b>Meaning:</b> Caller invoked service but was in 24 bit addressing mode. <b>Action:</b> Request this function only when you are in 31 bit addressing mode.
8	xxxx0825	<b>Equate Symbol:</b> IwmRsnCodeAscModeNotPrimary <b>Meaning:</b> Caller invoked service but was not in primary ASC mode. <b>Action:</b> Avoid requesting this function in this environment.
8	xxxx0827	<b>Equate Symbol:</b> IwmRsnCodeRsvdNot0 <b>Meaning:</b> Reserved field in parameter list was non-zero <b>Action:</b> Check for use of keywords that are not supported by the z/OS Release on which the program is running.
8	xxxx0828	<b>Equate Symbol:</b> IwmRsnCodeBadVersion <b>Meaning:</b> Version number in parameter list is not valid. <b>Action:</b> Check for possible storage overlay of the parameter list.
8	xxxx0841	<b>Equate Symbol:</b> IwmRsnCodeXmemMode <b>Meaning:</b> Caller is in cross-memory mode. <b>Action:</b> Request this function only when you are not in cross-memory mode.
8	xxxx0876	<b>Equate Symbol:</b> IwmRsnCodeNoCPUOnline <b>Meaning:</b> The system can not enforce the CPU affinity because all of the CPUs specified on CPUMASK keyword are either not configured online yet or currently taken offline by operator. <b>Action:</b> Request operator to bring online the CPUs necessary to satisfy the CPU affinity requirement.
C	—	<b>Equate Symbol:</b> IwmRetCodeEnvError <b>Meaning:</b> Environmental Error
C	xxxx0C01	<b>Equate Symbol:</b> IwmRsnCodeNoStg <b>Meaning:</b> No storage is available for the request. <b>Action:</b> Contact your system programmer.
C	xxxx0C3C	<b>Equate Symbol:</b> IwmRsnCodeConfigFailed <b>Meaning:</b> The system can not enforce the CPU affinity because the CPU required to satisfy the affinity is currently offline and the system failed to configure the CPU online. <b>Action:</b> Contact your system programmer.



Table 24. Return and Reason Codes for the IWMCPAFN Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
10	—	<p><b>Equate Symbol:</b> IwmRetCodeCompError</p> <p><b>Meaning:</b> Component Error</p> <p><b>Action:</b> Contact your system programmer.</p>

---

## Example

To enforce CPU affinity to CP 0, CP 3 and CP4

```

IWMCPAFN REQUEST=ENFORCE,
          CPUMASK='100110000000000'b,
          RETCODE=RC,RSNCODE=RSN
*
* Storage areas
*
RC      DS    F           Return code
RSN     DS    F           Reason code

```

## IWMCPAFN Macro

---

## Chapter 15. IWMCQRY – Query Classification Attributes

The Query Active Classification Rules routine is given control from the IWMCQRY macro. The Query Active Classification Rules macro will complete the parameter list with caller provided data and generate a stacking, space switching, program call to the query service.

The purpose of this routine is to return a representation of the classification rules that are associated with the active policy that is in effect for the sysplex. The data returned by this service describes the installation-defined rules that determine how incoming work is assigned a service class and/or report class by MVS.

The classification rule data returned by this service is mapped by macro IWMSVDCR. This macro is also used to map the classification rules associated with the WLM service definition. As a result, some data sections in this mapping will not be available (filled in) when it is obtained via this service. An example of some of the information that will not be available, are the timestamps indicating when a classification GROUP was last updated and by whom. For a complete list of fields that will not be available refer to the field comments in macro IWMSVDCR.

The classification rules can be used in conjunction with the active service policy to determine what performance goals will be associated with incoming work. The performance goals for a service class are contained within service policy mapping returned by the IWMPQRY service.

The information returned is not serialized upon return to the caller, and so may be out-of-date if a modified service definition was installed and a new policy activated.

The caller can optionally request that the identifier of the active policy that these classification rules are part of, be returned in an area specified by the POLICY\_ID keyword. The caller can then compare the policy identification information returned with the policy data returned by the IWMPQRY macro to ensure they are in synch.

The Query Active Classification Rules macro is provided in list, execute, and standard form. The list form accepts no variable parameters and is used only to reserve space for the query parameter list. The standard form is provided for use with routines which do not require reentrant code. The execute form is provided for use with the list format for reentrant routines.

The parameter list must be in the caller's primary address space or be addressable by the dispatchable unit access list.

---

### Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Supervisor state or program key mask (PKM) allowing keys 0-7.
<b>Dispatchable unit mode:</b>	Task
<b>Cross memory mode:</b>	Any PASN, any HASN, any SASN
<b>AMODE:</b>	31-bit

## IWMCQRY Macro

<b>ASC mode:</b>	Primary or access register (AR) If in Access Register ASC mode, specify SYSSTATE ASCENV=AR before invoking this macro.
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks may be held.
<b>Control parameters:</b>	All parameter areas must reside in current primary or be addressable by the dispatchable unit access list.

---

## Programming Requirements

1. The macro CVT must be included to use this macro.
2. This functions provided by this macro are only supported on MVS/ESA SP5.2.0 or later. If code with this macro expansion is executed on lower MVS/ESA releases (version 3 or later) a return code of 4 and reason code of xxxx041A (constant IwmRsnCodeNoPolMgt) will be returned indicating that the service is not available. This macro should NOT be executed on MVS versions 1 or 2.
3. The macro IWMYCON must be included to use this macro.
4. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
5. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, IWMRNSCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.

---

## Restrictions

The caller cannot have an EUT FRR established.

---

## Input Register Information

Before issuing the IWMCQRY macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

---

## Output Register Information

When control returns to the caller, the GPRs contain:

<b>Register</b>	<b>Contents</b>
<b>0</b>	Reason code if GR15 return code is non-zero
<b>1</b>	Used as work registers by the system
<b>2-13</b>	Unchanged
<b>14</b>	Used as a work register by the system
<b>15</b>	Return code

When control returns to the caller, the ARs contain:

<b>Register</b>	<b>Contents</b>
<b>0-1</b>	Used as work registers by the system
<b>2-13</b>	Unchanged
<b>14-15</b>	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

---

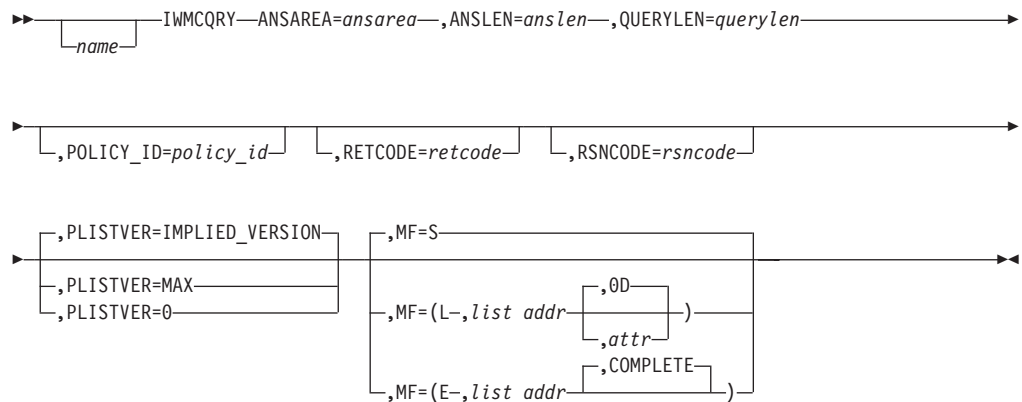
## Performance Implications

None.

---

## Syntax

### main diagram




---

## Parameters

The parameters are explained as follows:

### *name*

An optional symbol, starting in column 1, that is the name on the IWMCQRY macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

### **ANSAREA=ansarea**

A required output parameter, variable specifying an area to contain the data being returned by IWMCQRY. The answer area is defined by the IWMSVDCR macro.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a character field.

### **,ANSLEN=anslen**

A required input parameter, variable which contains the length of the area provided to contain the data being returned by IWMCQRY.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

### **,MF=S**

**,MF=(L,list addr)**

**,MF=(L,list addr,attr)**

**,MF=(L,list addr,OD)**

**,MF=(E,list addr)**

**,MF=(E,list addr,COMPLETE)**

An optional input parameter that specifies the macro form.

## IWMCQRY Macro

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

*,list addr*

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

*,attr*

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

**,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

**,PLISTVER=IMPLIED\_VERSION**

**,PLISTVER=MAX**

**,PLISTVER=0**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form; in this way, MAX ensures that the parameter list does not overwrite nearby storage.

- **0**, if you use the currently available parameters.

**To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX
- A decimal value of 0

**,POLICY\_ID=*policy\_id***

An optional output parameter, variable specifying an area to contain the

identifier of the active policy that these classification rules are a part of. This answer is mapped by the SVIDSSVP DSECT in the IWMSVIDS macro.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a character field.

**,QUERYLEN=***querylen*

A required output parameter, variable which contains the number of bytes needed to contain the classification rule information.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

**,RETCODE=***retcode*

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,RSNCODE=***rsncode*

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

---

## ABEND Codes

**Reason Code (Hex)**  
**Explanation**

**0Axx0005**

An attempt to reference caller's parameters caused an OC4 abend.

---

## Return and Reason Codes

When the IWMCQRY macro returns control to your program:

- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, if you coded RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 25. Return and Reason Codes for the IWMCQRY Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<p><b>Equate Symbol:</b> <code>IwmRetCodeOk</code></p> <p><b>Meaning:</b> Successful completion.</p> <p><b>Action:</b> None required.</p>
4	—	<p><b>Equate Symbol:</b> <code>IwmRetCodeWarning</code></p> <p><b>Meaning:</b> Successful completion, unusual conditions noted.</p>

Table 25. Return and Reason Codes for the IWMQRY Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
4	xxxx040A	<p><b>Equate Symbol:</b> IwmRsnCodeOutputAreaTooSmall</p> <p><b>Meaning:</b> The output area supplied is too small to receive all the available information. The variable specified in the QUERYLEN keyword will contain the size of the storage required to hold the returned data area.</p> <p><b>Action:</b> None required. If necessary, reinvoke the service with an output area of sufficient size to receive all information.</p>
4	xxxx041A	<p><b>Equate Symbol:</b> IwmRsnCodeNoPolMgt</p> <p><b>Meaning:</b> System does not support policy management services. This return code is only set when the MVS release is prior to MVS/ESA SP5.2.0.</p> <p><b>Action:</b> None required.</p>
8	—	<p><b>Equate Symbol:</b> IwmRetCodeInvocError</p> <p><b>Meaning:</b> Invalid invocation environment or parameters.</p>
8	xxxx0801	<p><b>Equate Symbol:</b> IwmRsnCodeSrbMode</p> <p><b>Meaning:</b> Caller is in SRB mode.</p> <p><b>Action:</b> Avoid requesting this function while in SRB mode.</p>
8	xxxx0803	<p><b>Equate Symbol:</b> IwmRsnCodeDisabled</p> <p><b>Meaning:</b> Caller is disabled.</p> <p><b>Action:</b> Avoid requesting this function while disabled.</p>
8	xxxx0804	<p><b>Equate Symbol:</b> IwmRsnCodeLocked</p> <p><b>Meaning:</b> Caller is locked.</p> <p><b>Action:</b> Avoid requesting this function while locked.</p>
8	xxxx0810	<p><b>Equate Symbol:</b> IwmRsnCodeEutFrr</p> <p><b>Meaning:</b> Caller has EUT FRR established.</p> <p><b>Action:</b> Avoid requesting this function with an EUT FRR set.</p>
8	xxxx0823	<p><b>Equate Symbol:</b> IwmRsnCodeDatoff</p> <p><b>Meaning:</b> Caller invoked service while DATOFF</p> <p><b>Action:</b> Avoid requesting this function in this environment.</p>
8	xxxx0824	<p><b>Equate Symbol:</b> IwmRsnCodeAmode24</p> <p><b>Meaning:</b> Caller invoked service but was in 24 bit addressing mode.</p> <p><b>Action:</b> Request this function only when you are in 31 bit addressing mode.</p>
8	xxxx0827	<p><b>Equate Symbol:</b> IwmRsnCodeRsvdNot0</p> <p><b>Meaning:</b> Reserved field in parameter list was non-zero.</p> <p><b>Action:</b> Check for possible storage overlay of the parameter list.</p>



Table 25. Return and Reason Codes for the IWMCQRY Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0828	<p><b>Equate Symbol:</b> IwmRsnCodeBadVersion</p> <p><b>Meaning:</b> Version number in parameter list is not valid or the length specified is incorrect.</p> <p><b>Action:</b> Check for possible storage overlay of the parameter list.</p>
8	xxxx0830	<p><b>Equate Symbol:</b> IwmRsnCodeBadAlet</p> <p><b>Meaning:</b> Caller has passed an invalid ALET.</p> <p><b>Action:</b> Check for possible storage overlay of the parameter list or variable.</p>
10	—	<p><b>Equate Symbol:</b> IwmRetCodeCompError</p> <p><b>Meaning:</b> Component error.</p> <p><b>Action:</b> No action required. The function may be successful if invoked again.</p>



---

## Chapter 16. IWMDEXTR – Extract Service Definition

The Extract Service Definition routine is given control from the IWMDEXTR macro. The Extract Service Definition macro will complete the parameter list with caller provided data and generate a stacking, program call to the extract service.

The purpose of this routine is to return a representation of the Workload Management Service Definition currently installed in the WLM couple dataset for the current sysplex. The service definition returned contains all the policies that are currently eligible to be activated in the sysplex.

The information returned can be used by an application to be presented and manipulated by an end user. The Install Service Definition macro IWMDINST, may be used to install a Service Definition into the WLM couple dataset.

The service definition is a single logical entity described by the service definition descriptor element, defined by IWMSERVD. The service definition descriptor element contains offsets to the 5 distinct areas that comprise the service definition:

- The general service definition data area.  
This data area contains general service definition information like the service definition name and description along with more detailed information like the policy, workload, service class and resource group information. This area is mapped by IWMSVDEF.
- The service definition classification rules data area.  
This data area contains the definitions of the classification rules and classification groups that govern which service and report classes are associated with incoming work when the work enters MVS. This area is mapped by IWMSVDCR.
- The notepad data area.  
This data area contains any comments (or change history) that an installation chooses to associate with the service definition. This area is mapped by IWMSVNPA.
- The service definition application environment data area.  
This data area contains the definitions of the application environments. This area is mapped by IWMSVAEA.
- The service definition scheduling environment data area.  
This data area contains the definitions of the scheduling environments. This area is mapped by IWMSVSEA.

The caller must provide sufficient storage to contain the service definition data requested. If insufficient storage is passed, no data is returned, an appropriate return and reason code is set, and the length required is returned in the variable specified in the QUERYLEN keyword.

Since the data returned is not serialized against future installs, the length returned may still change before the extract is issued again. Therefore, the caller must issue the extract service in a loop, checking return and reason codes, and obtaining a larger storage area as necessary.

The Extract Service Definition macro is provided in list, execute, and standard form. The list form accepts no variable parameters and is used only to reserve space for

## IWMDEXTR Macro

the extract parameter list. The standard form is provided for use with routines which do not require reentrant code. The execute form is provided for use with the list format for reentrant routines.

The parameter list must be in the caller's primary address space or be addressable by the dispatchable unit access list.

---

## Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Problem state. Any PSW key
<b>Dispatchable unit mode:</b>	Task
<b>Cross memory mode:</b>	PASN=HASN=SASN
<b>AMODE:</b>	31-bit
<b>ASC mode:</b>	Primary or access register (AR) If in Access Register ASC mode, specify SYSSTATE ASCENV=AR before invoking this macro.
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks may be held.
<b>Control parameters:</b>	All parameter areas must reside in current primary or be addressable by the dispatchable unit access list.

In addition, all parameters must reside in storage of the same key as the caller is executing in when the macro is invoked unless the the caller is in key 0.

---

## Programming Requirements

1. The macro CVT must be included to use this macro.
2. This functions provided by this macro are only supported on MVS/ESA SP5.2.0 or later. If code with this macro expansion is executed on lower MVS/ESA releases (version 3 or later) a return code of 4 and reason code of xxxx041A (constant IwmRsnCodeNoPolMgt) will be returned indicating that the service is not available. This macro should NOT be executed on MVS versions 1 or 2.
3. The macro IWMYCON must be included to use this macro.
4. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
5. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, IWMSNCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.

---

## Restrictions

The caller cannot have an EUT FRR established.

---

## Input Register Information

Before issuing the IWMDEXTR macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

## Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Reason code if GR15 return code is non-zero
1	Used as work registers by the system
2-13	Unchanged
14	Used as a work register by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

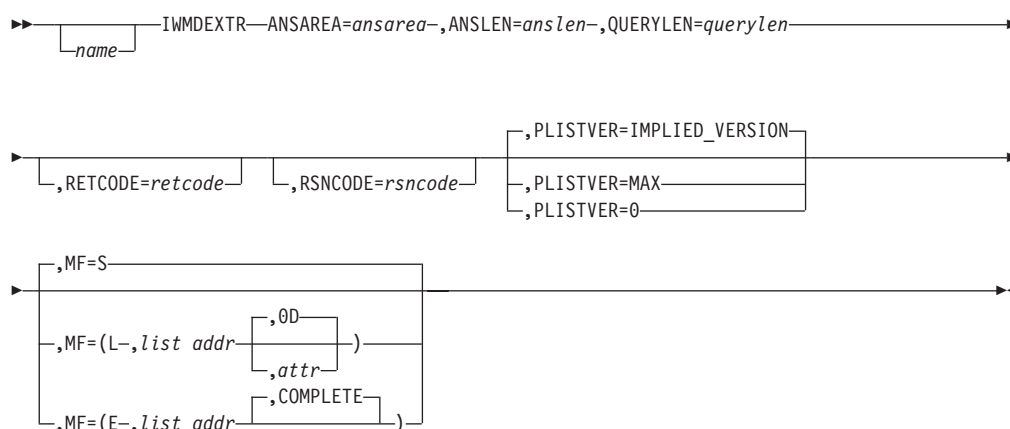
Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

## Performance Implications

None.

## Syntax

### main diagram



## Parameters

The parameters are explained as follows:

### *name*

An optional symbol, starting in column 1, that is the name on the IWMDEXTR macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

## IWMDEXTR Macro

### **ANSAREA=ansarea**

A required output parameter, variable specifying an area to contain the service definition data returned by the extract service. This area is defined by the IWMSERVD macro.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a character field.

### **,ANSLEN=anslen**

A required input parameter, variable to contain the length of the area specified on ANSAREA keyword to contain the service definition data returned by the extract service.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

### **,MF=S**

#### **,MF=(L,list addr)**

#### **,MF=(L,list addr,attr)**

#### **,MF=(L,list addr,0D)**

#### **,MF=(E,list addr)**

#### **,MF=(E,list addr,COMPLETE)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

#### **,list addr**

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

#### **,attr**

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

### **,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

### **,PLISTVER=IMPLIED\_VERSION**

### **,PLISTVER=MAX**

### **,PLISTVER=0**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When

using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form; in this way, MAX ensures that the parameter list does not overwrite nearby storage.

- **0**, if you use the currently available parameters.

**To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX
- A decimal value of 0

**,QUERYLEN=***querylen*

A required output parameter, variable which contains the number of bytes of service definition data returned by the extract service, or the number of bytes of storage required to contain the service definition if insufficient storage was provided.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

**,RETCODE=***retcode*

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,RSNCODE=***rsncode*

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

---

## ABEND Codes

Reason Code (Hex)	Explanation
-------------------	-------------

0Axx0005	
----------	--

	An attempt to reference caller's parameters caused an OC4 abend.
--	--

---

## Return and Reason Codes

When the IWMDEXTR macro returns control to your program:

- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, if you coded RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

## IWMDEXTR Macro

Table 26. Return and Reason Codes for the IWMDEXTR Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<p><b>Equate Symbol:</b> IwmRetCodeOk</p> <p><b>Meaning:</b> Successful completion.</p> <p><b>Action:</b> None required.</p>
4	—	<p><b>Equate Symbol:</b> IwmRetCodeWarning</p> <p><b>Meaning:</b> Successful completion, unusual conditions noted.</p>
4	xxxx040A	<p><b>Equate Symbol:</b> IwmRsnCodeOutputAreaTooSmall</p> <p><b>Meaning:</b> The output area supplied for the service definition area (ANSAREA keyword on macro IWMDEXTR) is too small to receive all the available information. As a result no service definition data is returned. The length required to receive all the service definition data is returned in the variable specified on the QUERYLEN keyword.</p> <p><b>Action:</b> None required. If necessary, reinvoke the service with an output area of sufficient size to receive all information.</p>
4	xxxx0414	<p><b>Equate Symbol:</b> IwmRsnCodeNullCDS</p> <p><b>Meaning:</b> No service definition is currently installed. As a result, no service definition data is returned.</p> <p><b>Action:</b> None required.</p>
4	xxxx0417	<p><b>Equate Symbol:</b> IwmRsnCodeBadServDE</p> <p><b>Meaning:</b> Service Definition retrieved from WLM CDS has failed validation but the structure is still returned to the caller.</p> <p><b>Action:</b> None required. Caution is advised in using the structure.</p>
4	xxxx041A	<p><b>Equate Symbol:</b> IwmRsnCodeNoPolMgt</p> <p><b>Meaning:</b> System does not support policy management services. This return code is only set when the MVS release is prior to MVS/ESA SP5.2.0.</p> <p><b>Action:</b> None required.</p>
8	—	<p><b>Equate Symbol:</b> IwmRetCodeInvocError</p> <p><b>Meaning:</b> Invalid invocation environment or parameters.</p>
8	xxxx0801	<p><b>Equate Symbol:</b> IwmRsnCodeSrbMode</p> <p><b>Meaning:</b> Caller is in SRB mode.</p> <p><b>Action:</b> Avoid requesting this function while in SRB mode.</p>
8	xxxx0803	<p><b>Equate Symbol:</b> IwmRsnCodeDisabled</p> <p><b>Meaning:</b> Caller is disabled.</p> <p><b>Action:</b> Avoid requesting this function while disabled.</p>
8	xxxx0804	<p><b>Equate Symbol:</b> IwmRsnCodeLocked</p> <p><b>Meaning:</b> Caller is locked.</p> <p><b>Action:</b> Avoid requesting this function while locked.</p>



Table 26. Return and Reason Codes for the IWMDEXTR Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0810	<b>Equate Symbol:</b> lwmRsnCodeEutFrr <b>Meaning:</b> Caller has EUT FRR established. <b>Action:</b> Avoid requesting this function with an EUT FRR set.
8	xxxx0823	<b>Equate Symbol:</b> lwmRsnCodeDatoff <b>Meaning:</b> Caller invoked service while DATOFF <b>Action:</b> Avoid requesting this function in this environment.
8	xxxx0824	<b>Equate Symbol:</b> lwmRsnCodeAmode24 <b>Meaning:</b> Caller invoked service but was in 24 bit addressing mode. <b>Action:</b> Request this function only when you are in 31 bit addressing mode.
8	xxxx0827	<b>Equate Symbol:</b> lwmRsnCodeRsvdNot0 <b>Meaning:</b> Reserved field in parameter list was non-zero. <b>Action:</b> Check for possible storage overlay of the parameter list.
8	xxxx0828	<b>Equate Symbol:</b> lwmRsnCodeBadVersion <b>Meaning:</b> Version number in parameter list is not valid or the length specified is incorrect. <b>Action:</b> Check for possible storage overlay of the parameter list.
8	xxxx0830	<b>Equate Symbol:</b> lwmRsnCodeBadAlet <b>Meaning:</b> Caller has passed an invalid ALET. <b>Action:</b> Check for possible storage overlay of the parameter list or variable.
8	xxxx0841	<b>Equate Symbol:</b> lwmRsnCodeXMemMode <b>Meaning:</b> Caller is in cross memory mode. <b>Action:</b> Invoke the function in non-cross memory mode.
C	—	<b>Equate Symbol:</b> lwmRetCodeEnvError <b>Meaning:</b> Environmental error.
C	xxxx0C01	<b>Equate Symbol:</b> lwmRsnCodeNoStg <b>Meaning:</b> Storage is not available for the request. <b>Action:</b> There is a storage shortage. The function may work successfully at a later time.
C	xxxx0C0E	<b>Equate Symbol:</b> lwmRsnCodeInsufAccess <b>Meaning:</b> The RACF facility class is active and a profile has been defined for the MVSADMIN.WLM.POLICY RACF facility class profile to which the caller does not have sufficient (read) access. <b>Action:</b> Invoke the function when the conditions are alleviated.

## IWMDEXTR Macro

Table 26. Return and Reason Codes for the IWMDEXTR Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
C	xxxx0C0F	<b>Equate Symbol:</b> IwmRsnCodeCDSNotAvail  <b>Meaning:</b> A couple dataset for WLM has not been defined or it has been defined but this system does not have connectivity to the dataset.  <b>Action:</b> No action required.
10	—	<b>Equate Symbol:</b> IwmRetCodeCompError  <b>Meaning:</b> Component error.  <b>Action:</b> No action required. The function may be successful if invoked again.

---

## Chapter 17. IWMDINST – Install Service Definition

The Install Service Definition routine is given control from the IWMDINST macro. The Install Service Definition macro will complete the parameter list with caller provided data and generate a stacking, program call to the install service.

The purpose of this routine is to install the WLM service definition supplied into the WLM couple dataset for the current sysplex. After this service definition is installed in the couple dataset all policies contained in it are eligible to be activated in the sysplex.

The service definition is a single logical entity described by the service definition descriptor element, defined by IWMSERVD. The service definition descriptor element contains offsets to the 5 distinct areas that comprise the service definition:

- The general service definition data area.  
This data area contains general service definition information like the service definition name and description along with more detailed information like the policy, workload, service class and resource group information. This area is mapped by the IWMSVDEF.
- The service definition classification rules data area.  
This data area contains the definitions of the classification rules and classification groups that govern which service and report classes are associated with incoming work when the work enters MVS. This area is mapped by IWMSVDCR.
- The notepad data area.  
This data area contains any comments (or change history) that an installation chooses to associate with the service definition. This area is mapped by IWMSVNPA.
- The service definition application environment data area.  
This data area contains the definitions of the application environments. This area is mapped by IWMSVAEA.
- The service definition scheduling environment data area.  
This data area contains the definitions of the scheduling environments. This area is mapped by IWMSVSEA.

The Service definition descriptor element and all four data areas of the service definition must be passed as inputs to the install service definition service. Even if certain data areas are non-applicable, for example no notepad information exists, the data area header information must still be completely filled in and pointed to by the descriptor element.

All input data areas must represent a valid service definition in order for the install to occur. If validity checking for any section of the service definition fails, the entire install process is aborted and a return and reason code indicating that validation of the service definition failed is returned. In addition, a reason code describing the specific error detected is returned in the variable specified on keyword VALCHECK\_RSN and an offset to the specific section of the service definition where the error was detected is returned in the variable specified on the VALCHECK\_OFFSET keyword. Validity check processing occurs until the first error is detected and only a single error is identified on an invocation of this macro.

The caller can also request that the install occurs only if the service definition that was used as a base for the definition being installed is still the currently installed

## IWMDINST Macro

service definition in the WLM couple dataset. This allows the caller the ability to prevent inadvertent over-writes of service definition updates that some other user (caller) made in the window, from when the service definition was initially read, to when the current install with the updated service definition was issued. For more details, refer to the COND keyword below.

The parameter list must be in the caller's primary address space or be addressable by the dispatchable unit access list.

---

## Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Problem state. Any PSW key
<b>Dispatchable unit mode:</b>	Task
<b>Cross memory mode:</b>	PASN=HASN=SASN
<b>AMODE:</b>	31-bit
<b>ASC mode:</b>	Primary or access register (AR) If in Access Register ASC mode, specify SYSSTATE ASCENV=AR before invoking this macro.
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks may be held.
<b>Control parameters:</b>	All parameter areas must reside in current primary or be addressable by the dispatchable unit access list.

In addition, all parameters must reside in storage of the same key as the caller is executing in when the macro is invoked unless the the caller is in key 0.

---

## Programming Requirements

1. The macro CVT must be included to use this macro.
2. This functions provided by this macro are only supported on MVS/ESA SP5.2.0 or later. If code with this macro expansion is executed on lower MVS/ESA releases (version 3 or later) a return code of 4 and reason code of xxxx041A (constant IwmRsnCodeNoPolMgt) will be returned indicating that the service is not available. This macro should NOT be executed on MVS versions 1 or 2.
3. The macro IWMYCON must be included to use this macro.
4. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
5. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, IWMSRNCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.

---

## Restrictions

The caller cannot have an EUT FRR established.

## Input Register Information

Before issuing the IWMDINST macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

## Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Reason code if GR15 return code is non-zero
1	Used as work registers by the system
2-13	Unchanged
14	Used as a work register by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

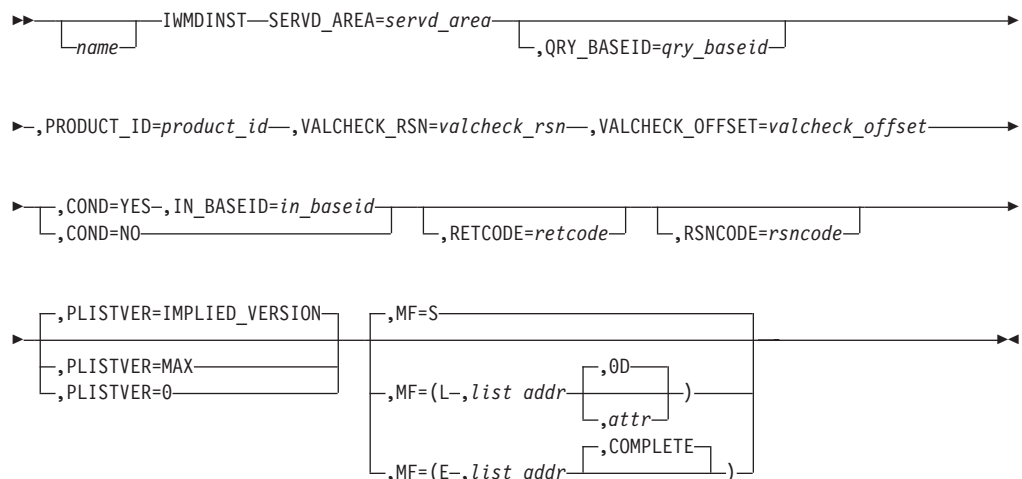
Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

## Performance Implications

None.

## Syntax

### main diagram



## Parameters

The parameters are explained as follows:

*name*

An optional symbol, starting in column 1, that is the name on the IWMDINST macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

**,COND=YES**

**,COND=NO**

A required parameter, which indicates whether checking is performed prior to the install, to determine if the service definition that the input definition is based on is still the currently installed service definition (i.e. another user has not made updates).

**,COND=YES**

indicates that the input service definition should only be installed if the identifier of currently installed service definition matches the base identifier passed on IN\_BASEID keyword. This allows the user to detect changes in the installed service definition, since the last extract was done, and allows the user to confirm whether the install should still occur.

**,COND=NO**

indicates that the input service definition should be installed unconditionally.

**,IN\_BASEID=*in\_baseid***

When COND=YES is specified, a required input parameter, variable specifying an area that contains the identifier of the service definition that was used as a base for the service definition being installed. This area is mapped by the SVIDSSVD DESCT in macro IWMSVIDS.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a character field.

**,MF=S**

**,MF=(L,*list addr*)**

**,MF=(L,*list addr,attr*)**

**,MF=(L,*list addr,OD*)**

**,MF=(E,*list addr*)**

**,MF=(E,*list addr,COMPLETE*)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

**,list addr**

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

**,attr**

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

**,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

**,PLISTVER=IMPLIED\_VERSION****,PLISTVER=MAX****,PLISTVER=0**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form; in this way, MAX ensures that the parameter list does not overwrite nearby storage.

- **0**, if you use the currently available parameters.

## IWMDINST Macro

**To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX
- A decimal value of 0

**,PRODUCT\_ID=***product\_id*

A required input parameter, variable specifying an area that contains an identifier of the product (application) performing the install. The identifier should include information like product name, a unique version/release identifier, and any other information that can help identify your product. This area is mapped by the SVIDSPRD DSECT in the IWMSVIDS macro.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32-character field.

**,QRY\_BASEID=***qry\_baseid*

An optional output parameter, variable specifying an area to contain the identifier of the service definition that is currently installed on the WLM couple dataset. This area is mapped by the SVIDSSVD DSECT in macro IWMSVIDS. When this keyword is specified, the data is returned when the return code indicates successful completion (return code 0) regardless of whether COND(YES) or COND(NO) was specified. In addition, this data is returned on a conditional request (COND(YES)) if the return and reason code indicate that specified IN\_BASEID does not match the baseid of the installed service definition (return code 4, reason code '0413'X).

**To code:** Specify the RS-type address, or address in register (2)-(12), of a character field.

**,RETCODE=***retcode*

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,RSNCODE=***rsncode*

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**SERVD\_AREA=***servd\_area*

A required input parameter, variable specifying an area that contains the service definition data to be installed. This area is defined by the IWMSERVD macro.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a character field.

**,VALCHECK\_OFFSET=***valcheck\_offset*

A required output parameter, variable to contain the offset from the beginning of the service definition (IWMSERVD) to the section of the input service definition where validity check processing found an error described by the reason code returned in VALCHECK\_RSN. This offset is returned under the same conditions as when VALCHECK\_RSN is returned.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

**,VALCHECK\_RSN=***valcheck\_rsn*

A required output parameter, variable that will contain the reason code identifying the specific error in the input service definition found during validity



checking. This reason code is only returned if validation of the input service definition fails and a primary return code of 8 and reason code of xxxx083D is returned.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

## ABEND Codes

**Reason Code (Hex)  
Explanation**

**0Axx0005**

An attempt to reference caller's parameters caused an OC4 abend.

## Return and Reason Codes

When the IWMDINST macro returns control to your program:

- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, if you coded RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 27. Return and Reason Codes for the IWMDINST Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> <code>IwmRetCodeOk</code> <b>Meaning:</b> Successful completion. <b>Action:</b> None required.
4	—	<b>Equate Symbol:</b> <code>IwmRetCodeWarning</code> <b>Meaning:</b> Successful completion, unusual conditions noted.
4	xxxx0413	<b>Equate Symbol:</b> <code>IwmRsnCodeIdsDontMatch</code> <b>Meaning:</b> COND=YES was specified on the IWMDINST macro, yet the service definition identifier specified on the IN_BASEID keyword did not match the identifier of the installed service definition. The identifier of the currently installed service definition is returned in the area specified on the QRY_BASEID keyword. <b>Action:</b> None required.
4	xxxx041A	<b>Equate Symbol:</b> <code>IwmRsnCodeNoPolMgt</code> <b>Meaning:</b> System does not support policy management services. This return code is only set when the MVS release is prior to MVS/ESA SP5.2.0. <b>Action:</b> None required.
8	—	<b>Equate Symbol:</b> <code>IwmRetCodeInvocError</code> <b>Meaning:</b> Invalid invocation environment or parameters.
8	xxxx0801	<b>Equate Symbol:</b> <code>IwmRsnCodeSrbMode</code> <b>Meaning:</b> Caller is in SRB mode. <b>Action:</b> Avoid requesting this function while in SRB mode.

Table 27. Return and Reason Codes for the IWMDINST Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0803	<b>Equate Symbol:</b> lwmRsnCodeDisabled <b>Meaning:</b> Caller is disabled. <b>Action:</b> Avoid requesting this function while disabled.
8	xxxx0804	<b>Equate Symbol:</b> lwmRsnCodeLocked <b>Meaning:</b> Caller is locked. <b>Action:</b> Avoid requesting this function while locked.
8	xxxx0810	<b>Equate Symbol:</b> lwmRsnCodeEutFrr <b>Meaning:</b> Caller has EUT FRR established. <b>Action:</b> Avoid requesting this function with an EUT FRR set.
8	xxxx0823	<b>Equate Symbol:</b> lwmRsnCodeDatoff <b>Meaning:</b> Caller invoked service while DATOFF <b>Action:</b> Avoid requesting this function in this environment.
8	xxxx0824	<b>Equate Symbol:</b> lwmRsnCodeAmode24 <b>Meaning:</b> Caller invoked service but was in 24 bit addressing mode. <b>Action:</b> Request this function only when you are in 31 bit addressing mode.
8	xxxx0827	<b>Equate Symbol:</b> lwmRsnCodeRsvdNot0 <b>Meaning:</b> Reserved field in parameter list was non-zero. <b>Action:</b> Check for possible storage overlay of the parameter list.
8	xxxx0828	<b>Equate Symbol:</b> lwmRsnCodeBadVersion <b>Meaning:</b> Version number in parameter list is not valid or the length specified is incorrect. <b>Action:</b> Check for possible storage overlay of the parameter list.
8	xxxx0829	<b>Equate Symbol:</b> lwmRsnCodeBadOptions <b>Meaning:</b> Parameter list omits required parameters or supplies mutually exclusive parameters or provides data associated with options not selected. <b>Action:</b> Check for possible storage overlay of the parameter list.
8	xxxx0830	<b>Equate Symbol:</b> lwmRsnCodeBadAlet <b>Meaning:</b> Caller has passed an invalid ALET. <b>Action:</b> Check for possible storage overlay of the parameter list or variable.
8	xxxx083D	<b>Equate Symbol:</b> lwmRsnCodeBadServDI <b>Meaning:</b> Caller has passed a Service Definition area that failed validation. <b>Action:</b> See values in VALCHECK_RSN and VALCHECK_OFFSET parameters for more information concerning the specific failure.

Table 27. Return and Reason Codes for the IWMDINST Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0841	<p><b>Equate Symbol:</b> lwmRsnCodeXMemMode</p> <p><b>Meaning:</b> Caller is in cross memory mode.</p> <p><b>Action:</b> Invoke the function in non-cross memory mode.</p>
8	xxxx085B	<p><b>Equate Symbol:</b> lwmRsnCodeNoSERVDArea</p> <p><b>Meaning:</b> Caller invoked service without a required SERVD area or the SERVD area address is 0.</p> <p><b>Action:</b> Check for possible storage overlay of the parameter list or variable.</p>
C	—	<p><b>Equate Symbol:</b> lwmRetCodeEnvError</p> <p><b>Meaning:</b> Environmental error.</p>
C	xxxx0C01	<p><b>Equate Symbol:</b> lwmRsnCodeNoStg</p> <p><b>Meaning:</b> Storage is not available for the request.</p> <p><b>Action:</b> There is a storage shortage. The function may work successfully at a later time.</p>
C	xxxx0C0E	<p><b>Equate Symbol:</b> lwmRsnCodeInsufAccess</p> <p><b>Meaning:</b> The RACF FACILITY class is active and a profile has been defined for the MVSADMIN.WLM.POLICY RACF facility class profile to which the caller does not have sufficient (update) access.</p> <p><b>Action:</b> Invoke the function when the conditions are alleviated.</p>
C	xxxx0C0F	<p><b>Equate Symbol:</b> lwmRsnCodeCDSNotAvail</p> <p><b>Meaning:</b> A couple dataset for WLM has not been defined or it has been defined but this system does not have connectivity to the data set.</p> <p><b>Action:</b> No action required.</p>
C	xxxx0C10	<p><b>Equate Symbol:</b> lwmRsnCodeCDSTooSmall</p> <p><b>Meaning:</b> WLM CDS is too small to process the request.</p> <p><b>Action:</b> No action required.</p>
10	—	<p><b>Equate Symbol:</b> lwmRetCodeCompError</p> <p><b>Meaning:</b> Component error.</p> <p><b>Action:</b> No action required. The function may be successful if invoked again.</p>

## IWMDINST Macro

---

## Chapter 18. IWMEBLK – Work Request Blocked

The IWMEBLK service allows work managers that participate in cross-platform enterprise workload management (EWLM) to indicate that processing of a work request is blocked while waiting for a work request in another application to complete.

The counterpart of this service to indicate that the processing of a work request is no longer blocked is IWMEUBLK.

---

### Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Supervisor state or program key mask (PKM) allowing keys 0-7
<b>Dispatchable unit mode:</b>	Task or SRB
<b>Cross memory mode:</b>	Any PASN, any HASN, any SASN
<b>AMODE:</b>	31- or 64-bit
<b>ASC mode:</b>	Primary or access register (AR)
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks may be held.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

---

### Programming Requirements

1. The macro CVT must be included to use this macro.
2. The macro IWMYCON must be included to use this macro.
3. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
4. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, IWMRSNCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.
5. All character data, unless otherwise specified, is assumed to be left justified and padded with blanks on the right, as needed, to occupy the specified number of bytes.
6. Since this service may only be used by system-like code, some validity checking on the parameter list is not performed. These checks would only be needed if the macro were not used to invoke the service routine.
7. If the parameter EWLMODE=EXPLICIT\_SINGLE is specified, the following restrictions apply:
  - No other task or SRB is allowed to issue other enclave services for the same enclave concurrently.
  - The caller must be in primary ASC mode before invocation.
  - The parameter list, the classification parameters and the savearea pointed to by GPR13 must be addressable in AMODE 31 and primary ASC mode.

## IWMEBLK Macro

- No recovery environment is set up by the service. The caller is responsible to provide an appropriate error recovery environment to handle abnormal terminations.
- The enclave must have been created with option ESTRT=EXPLICIT\_SINGLE on the IWM4ECRE (or IWMECREA) invocation.
- The caller must provide the ETOKEN parameter.

---

## Restrictions

None.

---

## Input Register Information

If the EWLMMODE=EXPLICIT\_SINGLE parameter is specified, the caller must provide a standard 72-byte savearea pointed to by GPR13.

For all other cases the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

---

## Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Reason code if GR15 return code is non-zero
1	Used as work registers by the system
2-13	Unchanged
14	Used as work registers by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

---

## Performance Implications

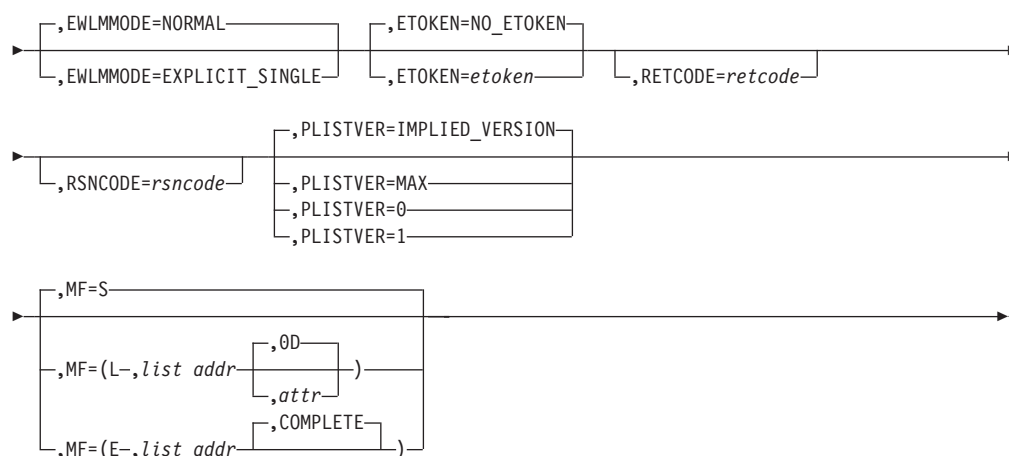
None.

---

## Syntax

**main diagram**

▶ `IWMEBLK—WORKREQ_HDL=workreq_hdl—,BLOCK_HDL=block_hdl—` ▶  
    └───┬───┘  
    *name*



## Parameters

The parameters are explained as follows:

### *name*

An optional symbol, starting in column 1, that is the name on the IWMEBLK macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

### **,BLOCK\_HDL=block\_hdl**

A required output parameter that will receive the handle identifying the blocked work request.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

### **ETOKEN=etoken**

### **ETOKEN=NO\_ETOKEN**

An optional input parameter that contains the enclave token of the enclave under which the work request is processed. The values of ETOKEN are:

- **etoken** is required, if the option EWLMMODE=EXPLICIT\_SINGLE is specified.
- **NO\_ETOKEN** indicates that no ETOKEN is passed. This is the default.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

### **,EWLMMODE=NORMAL**

### **,EWLMMODE=EXPLICIT\_SINGLE**

An optional input keyword that indicates if the parameter EXPLICIT\_SINGLE has been specified on the IWM4ECRE (or IWMECREA) call. Note that EWLMMODE=EXPLICIT\_SINGLE can only be specified if the enclave was created with the ESTRT=EXPLICIT\_SINGLE option. The values of EWLMMODE are:

- **EWLMMODE=NORMAL** indicates that the enclave was created with the ESTRT=EXPLICIT or with the ESTRT=IMPLIED option. EWLMMODE=NORMAL is the default.
- **EWLMMODE=EXPLICIT\_SINGLE** indicates that the enclave was created with the ESTRT=EXPLICIT\_SINGLE option which can only be used for a restricted environment. The caller must provide a standard 72-byte savearea,

## IWMEBLK Macro

which is addressable in AMODE 31 and pointed to by GPR13. The parameter list and the classification parameters must also be addressable in AMODE 31.

**,MF=S**  
**,MF=(L,list addr)**  
**,MF=(L,list addr,attr)**  
**,MF=(L,list addr,0D)**  
**,MF=(E,list addr)**  
**,MF=(E,list addr,COMPLETE)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

*,list addr*

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

*,attr*

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

**,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

**,PLISTVER=IMPLIED\_VERSION**

**,PLISTVER=MAX**

**,PLISTVER=0**

**,PLISTVER=1**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that



the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form, when both are assembled with the same level of the system. In this way, MAX ensures that the parameter list does not overwrite nearby storage.

- 0, if you use the currently available parameters.
- 1, which supports both the following parameters and those from version 0:

ETOKEN  
EWLMMODE

**To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX
- A decimal value of 0, or 1

**,RETCODE=retcode**

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,RSNCODE=rsncode**

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**WORKREQ\_HDL=workreq\_hdl**

A required input parameter that contains the handle which represents the work request. This handle was returned from a previous call to IWMESTRT or IWM4ECRE.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

---

## ABEND Codes

None.

---

## Return and Reason Codes

When the IWMEBLK macro returns control to your program:

- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, if you coded RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 28. Return and Reason Codes for the IWMEBLK Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<p><b>Equate Symbol:</b> <code>IwmRetCodeOk</code></p> <p><b>Meaning:</b> Successful completion.</p> <p><b>Action:</b> None required.</p>
8	—	<p><b>Equate Symbol:</b> <code>IwmRetCodeInvocError</code></p> <p><b>Meaning:</b> Invalid invocation environment or parameters.</p>

## IWMEBLK Macro

Table 28. Return and Reason Codes for the IWMEBLK Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0803	<p><b>Equate Symbol:</b> IwmRsnCodeDisabled</p> <p><b>Meaning:</b> Caller is disabled.</p> <p><b>Action:</b> Avoid requesting this function while disabled.</p>
8	xxxx0804	<p><b>Equate Symbol:</b> IwmRsnCodeLocked</p> <p><b>Meaning:</b> Caller is locked.</p> <p><b>Action:</b> Avoid requesting this function while locked.</p>
8	xxxx080B	<p><b>Equate Symbol:</b> IwmRsnCodeBadPI</p> <p><b>Meaning:</b> Error accessing parameter list.</p> <p><b>Action:</b> Check for possible storage overlay.</p>
8	xxxx0824	<p><b>Equate Symbol:</b> IwmRsnCodeAmode24</p> <p><b>Meaning:</b> Caller invoked service but was in 24-bit addressing mode.</p> <p><b>Action:</b> Request this function only when you are in 31 bit addressing mode.</p>
8	xxxx0828	<p><b>Equate Symbol:</b> IwmRsnCodeBadVersion</p> <p><b>Meaning:</b> Version number in parameter list is not valid.</p> <p><b>Action:</b> Check for possible storage overlay of the parameter list.</p>
8	xxxx083A	<p><b>Equate Symbol:</b> IwmRsnCodeBadEnclave</p> <p><b>Meaning:</b> EWLMODE=EXPLICIT_SINGLE was specified and the required enclave token does not pass verification.</p> <p><b>Action:</b> Check for possible storage overlay of the enclave token, or for asynchronous events which may have deleted the enclave.</p>
8	xxxx0895	<p><b>Equate Symbol:</b> IwmRsnCodeEWLMservNotEnabled</p> <p><b>Meaning:</b> Service is not enabled because caller invoked the IWM4CON (IWMCONN) service with EWLM=NO.</p> <p><b>Action:</b> Ensure that EWLM=YES is specified on the IWM4CON (IWMCONN) request to enable this service.</p>
8	xxxx0896	<p><b>Equate Symbol:</b> IwmRsnCodeBadWorkReqHandle</p> <p><b>Meaning:</b> Work request handle is invalid.</p> <p><b>Action:</b> Check the specification of the WORKREQ_HDL parameter.</p>

Table 28. Return and Reason Codes for the IWMEBLK Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx08A0	<p><b>Equate Symbol:</b> IwmRsnCodeNotExplicitSingle</p> <p><b>Meaning:</b> The service has been invoked with option EWLMMODE=EXPLICIT_SINGLE but the enclave has not been created with the ESTRT=EXPLICIT_SINGLE option or vice versa.</p> <p><b>Action:</b> If ESTRT=EXPLICIT_SINGLE was coded on the IWM4ECRE or IWMECREA call, EWLMMODE=EXPLICIT_SINGLE must also specified on the IWMEBLK call. If ESTRT=EXPLICIT OR ESTRT=IMPLIED was coded on the IWM4ECRE or IWMECREA call, EWLMMODE=NORMAL must be specified on the IWMEBLK call (or the EWLMMODE parameter can be omitted).</p>
10	—	<p><b>Equate Symbol:</b> IwmRetCodeCompError</p> <p><b>Meaning:</b> Component error.</p> <p><b>Action:</b> No action required. The function may be successful if invoked again.</p>

## Examples

```

                IWMEBLK WORKREQ_HDL=WRHANDLE,
                BLOCK_HDL=BKHANDLE
*
* Storage areas
*
WRHANDLE DS    CL8           Work Request Handle
BKHANDLE DS    CL8           Work Request Block Handle
    
```



---

## Chapter 19. IWMECQRY – Query Enclave Classification Attributes

The purpose of this service is to query the classification attributes of an Enclave. The output is mapped by IWMECD.

The Query macro is provided in list, execute, and standard form. The list form accepts no variable parameters and is used only to reserve space for the parameter list. The standard form is provided for use with routines which do not require reentrant code. The execute form is provided for use with the list format for reentrant routines.

The parameter list must be in the caller's primary address space or be addressable by the dispatchable unit access list.

---

### Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Supervisor state or program key mask (PKM) allowing keys 0-7.
<b>Dispatchable unit mode:</b>	Task or SRB
<b>Cross memory mode:</b>	Any PASN, any HASN, any SASN
<b>AMODE:</b>	31-bit
<b>ASC mode:</b>	Primary or access register (AR) Any P,S,H.  If in Access Register ASC mode, specify SYSSTATE ASCENV=AR before invoking this macro.
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks may be held.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

---

### Programming Requirements

1. This macro may only be used on MVS/ESA SP5.2 or later levels.
2. The macro IWMYCON must be included to use this macro.
3. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
4. Reason code and return code constants are defined within IWMYCON.
5. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above.

The constant, IWMRNSNCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.

---

### Restrictions

1. This macro may not be used prior to the completion of WLM address space initialization
2. All parameter areas must reside in current primary or be addressable by the dispatchable unit access list.

## IWMECQRY Macro

3. The caller must provide storage for an answer area mapped by IWMECD. This answer area may reside in the caller's primary address space, or in a dataspace accessible via the current unit of work's dispatchable unit access list (DUal).

---

## Input Register Information

Before issuing the IWMECQRY macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

---

## Output Register Information

When control returns to the caller, the GPRs contain:

<b>Register</b>	<b>Contents</b>
<b>0</b>	Reason code if GR15 return code is non-zero
<b>1</b>	Used as work register by the system
<b>2-13</b>	Unchanged
<b>14</b>	Used as work register by the system
<b>15</b>	Return code

When control returns to the caller, the ARs contain:

<b>Register</b>	<b>Contents</b>
<b>0-1</b>	Used as work registers by the system
<b>2-13</b>	Unchanged
<b>14-15</b>	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

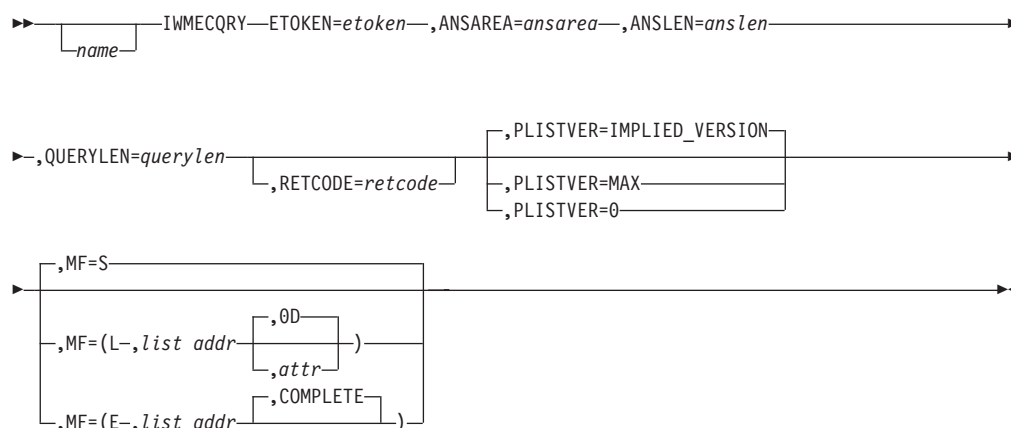
---

## Performance Implications

None.

## Syntax

### main diagram



## Parameters

The parameters are explained as follows:

### *name*

An optional symbol, starting in column 1, that is the name on the IWMECQRY macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

### **,ANSAREA=ansarea**

A required output parameter, which specifies an area to contain the data being returned. The answer area is defined by the IWMECD macro.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a character field.

### **,ANSLEN=anslen**

A required input parameter, variable which contains the length of the area provided to contain the data being returned by IWMECQRY.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

### **ETOKEN=etoken**

A required input parameter, which contains the Enclave token representing the Enclave of interest.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

### **,MF=S**

**,MF=(L,list addr)**

**,MF=(L,list addr,attr)**

**,MF=(L,list addr,OD)**

**,MF=(E,list addr)**

**,MF=(E,list addr,COMPLETE)**

An optional input parameter that specifies the macro form.

## IWMECQRY Macro

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

*,list addr*

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

*,attr*

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

**,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

**,PLISTVER=IMPLIED\_VERSION**

**,PLISTVER=MAX**

**,PLISTVER=0**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form; in this way, MAX ensures that the parameter list does not overwrite nearby storage.

- **0**, if you use the currently available parameters.

**To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX
- A decimal value of 0

**,QUERYLEN=querylen**

A required output parameter, variable which contains the number of bytes needed to contain the classification attributes being returned by IWMECQRY.



The length of the area needed to contain the data is dependent on the MVS release. If the ANSLEN is less than the QUERYLEN, then no data is returned in the output area specified by ANSAREA.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

**,RETCODE=retcode**

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

## ABEND Codes

None.

## Return and Reason Codes

When the IWMECQRY macro returns control to your program, GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 29. Return and Reason Codes for the IWMECQRY Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> IwmRetCodeOk <b>Meaning:</b> Successful completion. <b>Action:</b> None required.
4	—	<b>Equate Symbol:</b> IwmRetCodeWarning <b>Meaning:</b> Successful completion, unusual conditions noted.
4	xxxx040A	<b>Equate Symbol:</b> IwmRsnCodeOutputAreaTooSmall <b>Meaning:</b> The output area supplied is too small to receive all the available information. <b>Action:</b> None required.
4	xxxx043C	<b>Equate Symbol:</b> IwmRsnCodelsReset <b>Meaning:</b> Classification information returned may not reflect how the independent enclave is being managed. The independent enclave was reset to another service class or is reset quiesced. <b>Action:</b> None required.
8	—	<b>Equate Symbol:</b> IwmRetCodeInvocError <b>Meaning:</b> Invalid invocation environment or parameters
8	xxxx0803	<b>Equate Symbol:</b> IwmRsnCodeDisabled <b>Meaning:</b> Caller is disabled. <b>Action:</b> Avoid requesting this function while disabled.

## IWMECQRY Macro

Table 29. Return and Reason Codes for the IWMECQRY Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0804	<p><b>Equate Symbol:</b> lwmRsnCodeLocked</p> <p><b>Meaning:</b> Caller is locked.</p> <p><b>Action:</b> Avoid requesting this function while locked.</p>
8	xxxx080B	<p><b>Equate Symbol:</b> lwmRsnCodeBadPI</p> <p><b>Meaning:</b> Error accessing parameter list.</p> <p><b>Action:</b> Check for possible storage overlay.</p>
8	xxxx0824	<p><b>Equate Symbol:</b> lwmRsnCodeAmode24</p> <p><b>Meaning:</b> Caller invoked service but was in 24 bit addressing mode.</p> <p><b>Action:</b> Request this function only when you are in 31 bit addressing mode.</p>
8	xxxx0830	<p><b>Equate Symbol:</b> lwmRsnCodeBadAlet</p> <p><b>Meaning:</b> Caller has passed an invalid ALET.</p> <p><b>Action:</b> Check for possible storage overlay of the parameter list or variable.</p>
8	xxxx083A	<p><b>Equate Symbol:</b> lwmRsnCodeBadEnclave</p> <p><b>Meaning:</b> Enclave token is invalid.</p> <p><b>Action:</b> Check the specification of the ETOKEN parameter.</p>
10	—	<p><b>Equate Symbol:</b> lwmRetCodeCompError:</p> <p><b>Meaning:</b> Component error</p> <p><b>Action:</b> No action required. The function may be successful if invoked again.</p>

---

## Chapter 20. IW MEDREG — WLM Enclave Deregister Service

The IW MEDREG service allows the caller to deregister an enclave which it previously registered using the IW MEREG service. Deregistration is required as soon as the caller has finished using the enclave so that the enclave can eventually be deleted. If enclave deletion was requested while the enclave was registered, deletion occurs when the last deregistration takes place.

The caller can run in task or SRB mode.

---

### Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Supervisor state or program key mask (PKM) allowing keys 0-7.
<b>Dispatchable unit mode:</b>	Task or SRB
<b>Cross memory mode:</b>	Any PASN, any HASN, any SASN
<b>AMODE:</b>	31-bit
<b>ASC mode:</b>	Primary or access register (AR)
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks may be held.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

---

### Programming Requirements

1. The macro CVT must be included to use this macro.
2. The macro IW MYCON must be included to use this macro.
3. The macro IW MPB must be in the library concatenation, since it is included by IW MYCON.
4. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, IW MRSNCODE\_MASK\_CONST defined in IW MYCON, may be used for this purpose.
5. All character data, unless otherwise specified, is assumed to be left justified and padded with blanks on the right, as needed, to occupy the specified number of bytes.
6. Since this service may only be used by system-like code, some validity checking on the parameter list is not performed. These checks would only be needed if the macro were not used to invoke the service routine.

---

### Restrictions

None

---

### Input Register Information

Before issuing the IW MEDREG macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

## Output Register Information

When control returns to the caller, the GPRs contain:

### Register

#### Contents

- 0** Reason code if GR15 return code is non-zero
- 1** Used as work registers by the system
- 2-13** Unchanged
- 14** Used as work registers by the system
- 15** Return code

When control returns to the caller, the ARs contain:

### Register

#### Contents

- 0-1** Used as work registers by the system
- 2-13** Unchanged
- 14-15** Used as work registers by the system

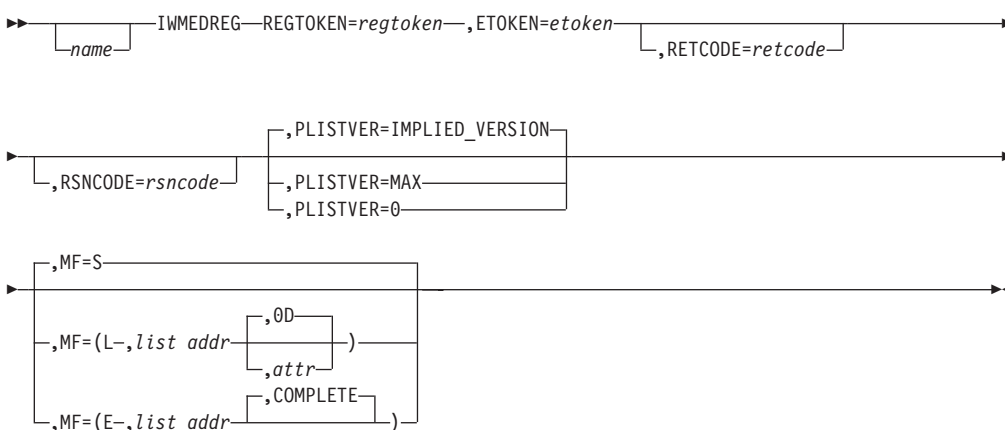
Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

## Performance Implications

None

## Syntax

### main diagram



## Parameters

The parameters are explained as follows:

### *name*

An optional symbol, starting in column 1, that is the name on the IWMEDREG macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

**,ETOKEN=*etoken***

A required input parameter that contains the enclave token.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

**,MF=S**

**,MF=(L,*list addr*)**

**,MF=(L,*list addr,attr*)**

**,MF=(L,*list addr,OD*)**

**,MF=(E,*list addr*)**

**,MF=(E,*list addr,COMPLETE*)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

**,*list addr***

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

**,*attr***

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

**,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

**,PLISTVER=IMPLIED\_VERSION**

**,PLISTVER=MAX**

**,PLISTVER=0**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that

## IWMEDREG Macro

the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form, when both are assembled with the same level of the system. In this way, MAX ensures that the parameter list does not overwrite nearby storage.

- 0, if you use the currently available parameters.

**To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX
- A decimal value of 0

**REGTOKEN=***regtoken*

A required input parameter, which passes the registration token obtained in a previous call to service IWMEREG.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

**,RETCODE=***retcode*

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,RSNCODE=***rsncode*

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

---

## ABEND Codes

None.

---

## Return and Reason Codes

When the IWMEDREG macro returns control to your program:

- GPR 15 (and *retcode*, when you code RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, when you code RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 30. Return and Reason Codes for the IWMEDREG Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> IwmRetCodeOk <b>Meaning:</b> Successful completion. <b>Action:</b> None required.
8	—	<b>Equate Symbol:</b> IwmRetCodeInvocError <b>Meaning:</b> Invalid invocation environment or parameters.
8	xxxx0803	<b>Equate Symbol:</b> IwmRsnCodeDisabled <b>Meaning:</b> Caller is disabled. <b>Action:</b> Avoid requesting this function while disabled.

Table 30. Return and Reason Codes for the IWMEDREG Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0804	<b>Equate Symbol:</b> lwmRsnCodeLocked <b>Meaning:</b> Caller is locked. <b>Action:</b> Avoid requesting this function while locked.
8	xxxx080B	<b>Equate Symbol:</b> lwmRsnCodeBadPI <b>Meaning:</b> Error accessing parameter list. <b>Action:</b> Check for possible storage overlay.
8	xxxx0824	<b>Equate Symbol:</b> lwmRsnCodeAmode24 <b>Meaning:</b> Caller invoked service but was in 24 bit addressing mode. <b>Action:</b> Request this function only when you are in 31 bit addressing mode.
8	xxxx0828	<b>Equate Symbol:</b> lwmRsnCodeBadVersion <b>Meaning:</b> Version number in parameter list is not valid. <b>Action:</b> Check for possible storage overlay of the parameter list.
8	xxxx083A	<b>Equate Symbol:</b> lwmRsnCodeBadEnclave <b>Meaning:</b> Enclave token does not pass verification. <b>Action:</b> Check for possible storage overlay of the enclave token, or asynchronous events which may have deleted the enclave.
8	xxxx0880	<b>Equate Symbol:</b> lwmRsnCodeBadRegToken <b>Meaning:</b> The register token does not pass verification. <b>Action:</b> Check for possible storage overlay of the register token, or asynchronous events which may have deregistered the enclave already.
10	—	<b>Equate Symbol:</b> lwmRetCodeCompError <b>Meaning:</b> Component error. <b>Action:</b> No action required. The function may be successful if invoked again.

## Example

```

IWMEDREG REGTOKEN=REGTKN,ETOKEN=ENCTKN
*
* Storage areas
*
ENCTKN DS CL8 Enclave token
REGTKN DS CL8 Register token

```





---

## Chapter 21. IWMEGCOR – Retrieving a Correlator

The IWMEGCOR service allows work managers that participate in cross-platform enterprise workload management (EWLM) to retrieve the correlator for a given work request handle or the maximum length of a correlator.

---

### Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Supervisor state or program key mask (PKM) allowing keys 0-7
<b>Dispatchable unit mode:</b>	Task or SRB
<b>Cross memory mode:</b>	Any PASN, any HASN, any SASN
<b>AMODE:</b>	31- or 64-bit
<b>ASC mode:</b>	Primary or access register (AR)
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks may be held.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

---

### Programming Requirements

1. The macro CVT must be included to use this macro.
2. The macro IWMYCON must be included to use this macro.
3. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
4. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, IWMRSNCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.
5. All character data, unless otherwise specified, is assumed to be left justified and padded with blanks on the right, as needed, to occupy the specified number of bytes.
6. Since this service may only be used by system-like code, some validity checking on the parameter list is not performed. These checks would only be needed if the macro were not used to invoke the service routine.
7. If the parameter EWLMODE=EXPLICIT\_SINGLE is specified, the following restrictions apply:
  - No other task or SRB is allowed to issue other enclave services for the same enclave concurrently.
  - The caller must be in primary ASC mode before invocation.
  - The parameter list, the classification parameters and the savearea pointed to by GPR13 must be addressable in AMODE 31 and primary ASC mode.
  - No recovery environment is set up by the service. The caller is responsible to provide an appropriate error recovery environment to handle abnormal terminations.
  - The enclave must have been created with option ESTRT=EXPLICIT\_SINGLE on the IWM4ECRE (or IWMECREA) invocation.
  - The caller must provide the ETOKEN parameter.

## Restrictions

None.

## Input Register Information

If the EWLMODE=EXPLICIT\_SINGLE parameter is specified, the caller must provide a standard 72-byte savearea pointed to by GPR13.

For all other cases the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

## Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Reason code if GR15 return code is non-zero
1	Used as work registers by the system
2-13	Unchanged
14	Used as work registers by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

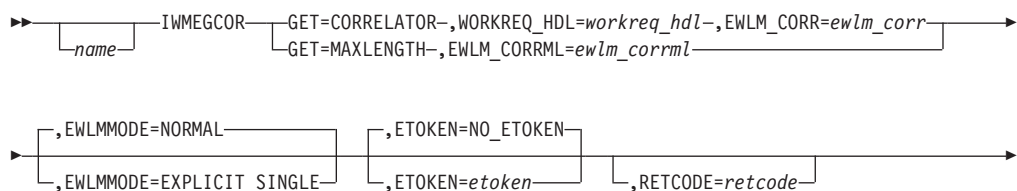
Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

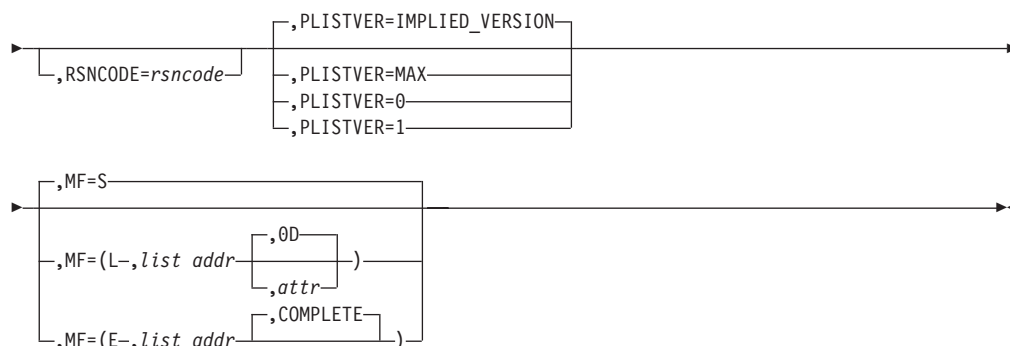
## Performance Implications

None.

## Syntax

### main diagram





## Parameters

The parameters are explained as follows:

### *name*

An optional symbol, starting in column 1, that is the name on the IWMEGCOR macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

### **ETOKEN=etoken**

### **ETOKEN=NO\_ETOKEN**

An optional input parameter that contains the enclave token of the enclave under which the work request is processed. The values of ETOKEN are:

- **etoken** is required, if the option EWLMODE=EXPLICIT\_SINGLE is specified.
- **NO\_ETOKEN** indicates that no ETOKEN is passed. This is the default.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

### **,EWLM\_CORR=ewlm\_corr**

When GET=CORRELATOR is specified, a required input parameter field where the service returns the correlator of the work request.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a character field.

### **,EWLM\_CORRML=ewlm\_corrml**

When GET=MAXLENGTH is specified, a required output parameter, which will receive the maximum length of a correlator.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

### **,EWLMODE=NORMAL**

### **,EWLMODE=EXPLICIT\_SINGLE**

An optional input keyword that indicates if the parameter EXPLICIT\_SINGLE has been specified on the IWM4ECRE (or IWMECREA) call. Note that EWLMODE=EXPLICIT\_SINGLE can only be specified if the enclave was created with the ESTRT=EXPLICIT\_SINGLE option. The values of EWLMODE are:

- **EWLMODE=NORMAL** indicates that the enclave was created with the ESTRT=EXPLICIT or with the ESTRT=IMPLIED option. EWLMODE=NORMAL is the default.
- **EWLMODE=EXPLICIT\_SINGLE** indicates that the enclave was created with the ESTRT=EXPLICIT\_SINGLE option which can only be used for a

restricted environment. The caller must provide a standard 72-byte savearea, which is addressable in AMODE 31 and pointed to by GPR13. The parameter list and the classification parameters must also be addressable in AMODE 31.

### **GET=CORRELATOR**

### **GET=MAXLENGTH**

A required parameter, which describes whether the correlator or the maximum length of a correlator is returned.

### **GET=CORRELATOR**

indicates that the correlator should be returned.

### **GET=MAXLENGTH**

indicates that the maximum length of a correlator should be returned.

### **,MF=S**

**,MF=(L,list addr)**

**,MF=(L,list addr,attr)**

**,MF=(L,list addr,OD)**

**,MF=(E,list addr)**

**,MF=(E,list addr,COMPLETE)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

**,list addr**

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

**,attr**

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

### **,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

### **,PLISTVER=IMPLIED\_VERSION**

**,PLISTVER=MAX**

**,PLISTVER=0**

**,PLISTVER=1**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an

optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form, when both are assembled with the same level of the system. In this way, MAX ensures that the parameter list does not overwrite nearby storage.

- **0**, which supports all parameters except those specifically referenced in higher versions.
- **1**, which supports both the following parameters and those from version 0:

ETOKEN  
EWLMMODE

**To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX
- A decimal value of 0, or 1

**,RETCODE=retcode**

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,RSNCODE=rsncode**

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**WORKREQ\_HDL=workreq\_hdl**

When GET=CORRELATOR is specified, a required input parameter that contains the handle which represents the work request. This handle was returned on a previous invocation of IWMESTRT or IWM4ECRE.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

---

## ABEND Codes

None.

---

## Return and Reason Codes

When the IWMEGCOR macro returns control to your program:

- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, if you coded RSNCODE) contains reason code.

## IWMEGCOR Macro

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 31. Return and Reason Codes for the IWMEGCOR Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> IwmRetCodeOk <b>Meaning:</b> Successful completion. <b>Action:</b> None required.
8	—	<b>Equate Symbol:</b> IwmRetCodeInvocError <b>Meaning:</b> Invalid invocation environment or parameters.
8	xxxx0803	<b>Equate Symbol:</b> IwmRsnCodeDisabled <b>Meaning:</b> Caller is disabled. <b>Action:</b> Avoid requesting this function while disabled.
8	xxxx0804	<b>Equate Symbol:</b> IwmRsnCodeLocked <b>Meaning:</b> Caller is locked. <b>Action:</b> Avoid requesting this function while locked.
8	xxxx080B	<b>Equate Symbol:</b> IwmRsnCodeBadPI <b>Meaning:</b> Error accessing parameter list. <b>Action:</b> Check for possible storage overlay.
8	xxxx0824	<b>Equate Symbol:</b> IwmRsnCodeAmode24 <b>Meaning:</b> Caller invoked service but was in 24 bit addressing mode. <b>Action:</b> Request this function only when you are in 31 bit addressing mode.
8	xxxx0828	<b>Equate Symbol:</b> IwmRsnCodeBadVersion <b>Meaning:</b> Version number in parameter list is not valid. <b>Action:</b> Check for possible storage overlay of the parameter list.
8	xxxx083A	<b>Equate Symbol:</b> IwmRsnCodeBadEnclave <b>Meaning:</b> EWLMODE=EXPLICIT_SINGLE was specified and the required enclave token does not pass verification. <b>Action:</b> Check for possible storage overlay of the enclave token, or for asynchronous events which may have deleted the enclave.
8	xxxx0895	<b>Equate Symbol:</b> IwmRsnCodeEWLMSEvNotEnabled <b>Meaning:</b> Service is not enabled because caller invoked the IWM4CON (IWMCONN) service with EWLM=NO. <b>Action:</b> Ensure that EWLM=YES is specified on the IWM4CON request to enable this service.
8	xxxx0896	<b>Equate Symbol:</b> IwmRsnCodeBadWorkReqHandle <b>Meaning:</b> Work request handle in parameter list is not valid. <b>Action:</b> Check the specification of the WORKREQ_HDL parameter.

Table 31. Return and Reason Codes for the IWMEGCOR Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx08A0	<p><b>Equate Symbol:</b> IwmRsnCodeNotExplicitSingle</p> <p><b>Meaning:</b> The service has been invoked with option EWLMODE=EXPLICIT_SINGLE but the enclave has not been created with the ESTRT=EXPLICIT_SINGLE option or vice versa.</p> <p><b>Action:</b> If ESTRT=EXPLICIT_SINGLE was coded on the IWM4ECRE or IWMECREA call, EWLMODE=EXPLICIT_SINGLE must also specified on the IWMEBLK call. If ESTRT=EXPLICIT OR ESTRT=IMPLIED was coded on the IWM4ECRE or IWMECREA call, EWLMODE=NORMAL must be specified on the IWMEBLK call (or the EWLMODE parameter can be omitted).</p>
10	—	<p><b>Equate Symbol:</b> IwmRetCodeCompError</p> <p><b>Meaning:</b> Component error.</p> <p><b>Action:</b> No action required. The function may be successful if invoked again.</p>

## Examples

```

        IWMEGCOR GET=MAXLENGTH,EWLM_CORRML=CMLEN
*
* Storage areas
*
CMLEN  DS    F                Maximum correlator length
        IWMEGCOR GET=CORRELATOR,
        WORKREQ_HDL=WRHANDLE,EWLM_CORR=CORR
*
* Storage areas
*
WRHANDLE DS    CL8            Work Request Handle
CORR     DS    CL127         Output field of size
                                maximum correlator length
    
```





---

## Chapter 22. IWMEJOIN – Join an Enclave

The purpose of this service is to allow the task invoking this service to join an Enclave for the purpose of performance management. The scope of this service affects only a single task at the time the service is invoked. However any tasks which are attached by the current task subsequently will also become part of the Enclave environment. This inheritance of the Enclave attribute will apply to any further level of newly attached subtasks as well. However subtasks which exist at the time this service is invoked will not become part of the Enclave environment nor will any subtasks which are created subsequently by these non-Enclave tasks become part of the Enclave environment, unless they explicitly join.

Note that a task may only join an Enclave if it is not already part of an Enclave. In particular, a subtask which inherited the Enclave attribute from its mother task (which may happen either as a result of the mother task issuing IWMEJOIN or IWM4STBG) is not allowed to use IWMEJOIN to explicitly join an Enclave. This restriction is independent of whether the Enclave specified is the same Enclave as it is in, or a different Enclave from the one it is in. Such a subtask which inherited the Enclave attribute is also not allowed to use IWMELEAV to explicitly leave the Enclave. The subtask would only leave the Enclave upon its own (task) termination or when the Enclave is deleted (IWM4EDEL). Also, a task which successfully establishes a Begin environment (IWM4STBG) may not invoke Enclave Join, nor is the task allowed to use Enclave Leave while this Begin environment exists.

Upon successful completion of this service, the CPU time for the task (and any subsequently attached subtasks) will be attributed to the Enclave for the purpose of service unit calculations and performance period switches, rather than being attributed to the address space owning the task. Management and reporting for the Enclave will include activity for the task until the task either leaves the Enclave or the Enclave is deleted.

---

### Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Supervisor state or program key mask (PKM) allowing keys 0-7.
<b>Dispatchable unit mode:</b>	Task
<b>Cross memory mode:</b>	Any PASN, any HASN, any SASN
<b>AMODE:</b>	31-bit
<b>ASC mode:</b>	Primary
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks may be held.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

---

### Programming Requirements

1. The macro CVT must be included to use this macro.
2. The macro IWMYCON must be included to use this macro.
3. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.

## IWMEJOIN Macro

- Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, IWMSRNCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.
- All character data, unless otherwise specified, is assumed to be left justified and padded with blanks on the right, as needed, to occupy the specified number of bytes.

---

## Restrictions

- The caller cannot have an EUT FRR established.

---

## Input Register Information

Before issuing the IWMEJOIN macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

---

## Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Reason code if GR15 return code is non-zero
1	Used as work registers by the system
2-13	Unchanged
14	Used as work registers by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

---

## Performance Implications

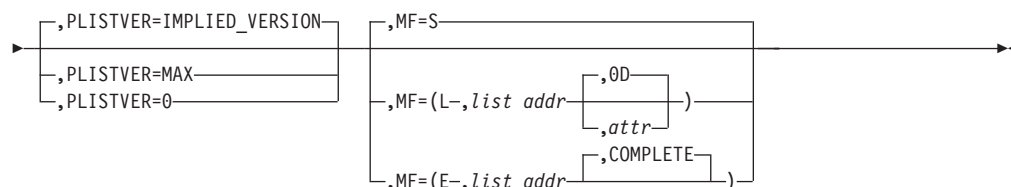
None.

---

## Syntax

### main diagram

→ `name` IWMEJOIN—ETOKEN=*etoken* [, RETCODE=*retcode*] [, RSNCODE=*rsncode*] →



## Parameters

The parameters are explained as follows:

### *name*

An optional symbol, starting in column 1, that is the name on the IWMEJOIN macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

### **ETOKEN=etoken**

A required input parameter, which contains the Enclave token to be associated with the task as returned by IWM4ECRE.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

### **,MF=S**

### **,MF=(L, list addr)**

### **,MF=(L, list addr, attr)**

### **,MF=(L, list addr, OD)**

### **,MF=(E, list addr)**

### **,MF=(E, list addr, COMPLETE)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

### *,list addr*

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

### *,attr*

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

### **,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

## IWMEJOIN Macro

### **,PLISTVER=IMPLIED\_VERSION**

### **,PLISTVER=MAX**

### **,PLISTVER=0**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form; in this way, MAX ensures that the parameter list does not overwrite nearby storage.

- **0**, if you use the currently available parameters.

**To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX
- A decimal value of 0

### **,RETCODE=*retcode***

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

### **,RSNCODE=*rsncode***

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

---

## ABEND Codes

None.

---

## Return and Reason Codes

When the IWMEJOIN macro returns control to your program:

- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, if you coded RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 32. Return and Reason Codes for the IWMEJOIN Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> lwmRetCodeOk <b>Meaning:</b> Successful completion. <b>Action:</b> None required.
4	—	<b>Equate Symbol:</b> lwmRetCodeWarning <b>Meaning:</b> Successful completion, unusual conditions noted.
4	xxxx041F	<b>Equate Symbol:</b> lwmRsnCodeExecEnvChanged <b>Meaning:</b> The execution environment has changed while the requested function is in progress. <b>Action:</b> None required.
8	—	<b>Equate Symbol:</b> lwmRetCodeInvocError <b>Meaning:</b> Invalid invocation environment or parameters.
8	xxxx0801	<b>Equate Symbol:</b> lwmRsnCodeSrbMode <b>Meaning:</b> Caller is in SRB mode. <b>Action:</b> Avoid requesting this function while in SRB mode.
8	xxxx0803	<b>Equate Symbol:</b> lwmRsnCodeDisabled <b>Meaning:</b> Caller is disabled. <b>Action:</b> Avoid requesting this function while disabled.
8	xxxx0804	<b>Equate Symbol:</b> lwmRsnCodeLocked <b>Meaning:</b> Caller is locked. <b>Action:</b> Avoid requesting this function while locked.
8	xxxx080B	<b>Equate Symbol:</b> lwmRsnCodeBadPI <b>Meaning:</b> Error accessing parameter list. <b>Action:</b> Check for possible storage overlay.
8	xxxx0810	<b>Equate Symbol:</b> lwmRsnCodeEutFrr <b>Meaning:</b> Caller has EUT FRR established. <b>Action:</b> Avoid requesting this function with an EUT FRR set.
8	xxxx0823	<b>Equate Symbol:</b> lwmRsnCodeDatoff <b>Meaning:</b> Caller invoked service while DATOFF <b>Action:</b> Avoid requesting this function in this environment.
8	xxxx0824	<b>Equate Symbol:</b> lwmRsnCodeAmode24 <b>Meaning:</b> Caller invoked service but was in 24 bit addressing mode. <b>Action:</b> Request this function only when you are in 31 bit addressing mode.
8	xxxx0825	<b>Equate Symbol:</b> lwmRsnCodeAscModeNotPrimary <b>Meaning:</b> Caller invoked service but was not DAT on Primary ASC mode. <b>Action:</b> Avoid requesting this function in this environment.

## IWMEJOIN Macro

Table 32. Return and Reason Codes for the IWMEJOIN Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0826	<p><b>Equate Symbol:</b> IwmRsnCodeTaskTerm</p> <p><b>Meaning:</b> Caller invoked service while task termination is in progress for the current task.</p> <p><b>Action:</b> Avoid requesting this function while task termination is in progress.</p>
8	xxxx0828	<p><b>Equate Symbol:</b> IwmRsnCodeBadVersion</p> <p><b>Meaning:</b> Version number in parameter list is not valid or version length field is not valid.</p> <p><b>Action:</b> Check for possible storage overlay of the parameter list.</p>
8	xxxx083A	<p><b>Equate Symbol:</b> IwmRsnCodeBadEnclave</p> <p><b>Meaning:</b> Enclave token does not pass verification.</p> <p><b>Action:</b> Check for possible storage overlay of the Enclave token, or asynchronous events which may have deleted the Enclave.</p>
8	xxxx0850	<p><b>Equate Symbol:</b> IwmRsnCodeBeginEnvOutstanding</p> <p><b>Meaning:</b> Caller is already operating under an outstanding Begin environment which has implicitly joined an Enclave.</p> <p><b>Action:</b> Avoid requesting this function in this environment.</p>
8	xxxx0857	<p><b>Equate Symbol:</b> IwmRsnCodeAlreadyInEnclave</p> <p><b>Meaning:</b> Current dispatchable workunit is already in an Enclave.</p> <p><b>Action:</b> Avoid requesting this function while the caller is already in an Enclave.</p>
10	—	<p><b>Equate Symbol:</b> IwmRetCodeCompError</p> <p><b>Meaning:</b> Component error.</p> <p><b>Action:</b> Contact your system programmer.</p>

## Example

To allow the current task to join an Enclave:

```

        IWMEJOIN ETOKEN=ENCTOKEN,RETCODE=RC,RSNCODE=RSN
*
* Storage areas
*
ENCTOKEN DS    CL8           Contains the Enclave token
*                               associated with the work
*                               request as returned by IWM4ECRE
RC        DS    F            Return code
RSN       DS    F            Reason code

```

---

## Chapter 23. IWMELEAV – Leave an Enclave

The purpose of this service is to allow the task invoking this service to leave an Enclave. For the purpose of performance management, the task will become associated with its home address space. The scope of this service affects the current task at the time the service is invoked.

Note that a task may only leave an Enclave if it explicitly joined the Enclave. A subtask which inherited the Enclave attribute from its mother task is not allowed to use IWMELEAV to explicitly leave the Enclave. The subtask would only leave the Enclave upon its own (task) termination or when the Enclave is deleted (IWM4EDEL). Also, a task which successfully establishes a Begin environment (IWM4STBG) may not invoke Enclave Join, nor is the task allowed to use Enclave Leave while this Begin environment exists.

Upon successful completion of this service, the CPU time for the task (and any subsequently attached subtasks) will be attributed to the home address space for the purpose of service unit calculations and performance period switches, rather than being attributed to the Enclave.

---

### Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Supervisor state or program key mask (PKM) allowing keys 0-7.
<b>Dispatchable unit mode:</b>	Task
<b>Cross memory mode:</b>	Any PASN, any HASN, any SASN
<b>AMODE:</b>	31-bit
<b>ASC mode:</b>	Primary
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks may be held.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

---

### Programming Requirements

1. The macro CVT must be included to use this macro.
2. The macro IWMYCON must be included to use this macro.
3. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
4. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, IWMRSCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.
5. All character data, unless otherwise specified, is assumed to be left justified and padded with blanks on the right, as needed, to occupy the specified number of bytes.

## Restrictions

The caller cannot have an EUT FRR established.

## Input Register Information

Before issuing the IWMELEAV macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

## Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Reason code if GR15 return code is non-zero
1	Used as work registers by the system
2-13	Unchanged
14	Used as work registers by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

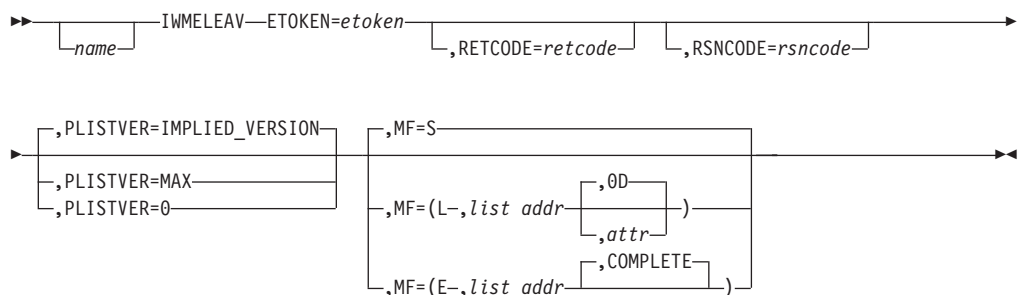
Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

## Performance Implications

None.

## Syntax

### main diagram



## Parameters

The parameters are explained as follows:



*name*

An optional symbol, starting in column 1, that is the name on the IWMELEAV macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

**ETOKEN=etoken**

A required input parameter, which contains the Enclave token associated with the work request as returned by IWM4ECRE.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

**,MF=S****,MF=(L,list addr)****,MF=(L,list addr,attr)****,MF=(L,list addr,0D)****,MF=(E,list addr)****,MF=(E,list addr,COMPLETE)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

*,list addr*

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

*,attr*

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

**,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

**,PLISTVER=IMPLIED\_VERSION****,PLISTVER=MAX****,PLISTVER=0**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.

## IWMELEAV Macro

- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify `PLISTVER=MAX` on the list form of the macro. Specifying `MAX` ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form; in this way, `MAX` ensures that the parameter list does not overwrite nearby storage.

- **0**, if you use the currently available parameters.

**To code:** Specify one of the following:

- `IMPLIED_VERSION`
- `MAX`
- A decimal value of 0

### **,RETCODE=retcode**

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

### **,RSNCODE=rsncode**

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

---

## ABEND Codes

None.

---

## Return and Reason Codes

When the `IWMELEAV` macro returns control to your program:

- GPR 15 (and *retcode*, if you coded `RETCODE`) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, if you coded `RSNCODE`) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 33. Return and Reason Codes for the `IWMELEAV` Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> <code>IwmRetCodeOk</code> <b>Meaning:</b> Successful completion. <b>Action:</b> None required.
4	—	<b>Equate Symbol:</b> <code>IwmRetCodeWarning</code> <b>Meaning:</b> Successful completion, unusual conditions noted.
4	xxxx041C	<b>Equate Symbol:</b> <code>IwmRsnCodeNotEnclave</code> <b>Meaning:</b> The current dispatchable workunit is not associated with an Enclave. <b>Action:</b> Check for possible asynchronous events which may have deleted the Enclave.

Table 33. Return and Reason Codes for the IWMELEAV Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	—	<b>Equate Symbol:</b> lwmRetCodeInvocError <b>Meaning:</b> Invalid invocation environment or parameters.
8	xxxx0801	<b>Equate Symbol:</b> lwmRsnCodeSrbMode <b>Meaning:</b> Caller is in SRB mode. <b>Action:</b> Avoid requesting this function while in SRB mode.
8	xxxx0803	<b>Equate Symbol:</b> lwmRsnCodeDisabled <b>Meaning:</b> Caller is disabled. <b>Action:</b> Avoid requesting this function while disabled.
8	xxxx0804	<b>Equate Symbol:</b> lwmRsnCodeLocked <b>Meaning:</b> Caller is locked. <b>Action:</b> Avoid requesting this function while locked.
8	xxxx080B	<b>Equate Symbol:</b> lwmRsnCodeBadPI <b>Meaning:</b> Error accessing parameter list. <b>Action:</b> Check for possible storage overlay.
8	xxxx0810	<b>Equate Symbol:</b> lwmRsnCodeEutFrr <b>Meaning:</b> Caller has EUT FRR established. <b>Action:</b> Avoid requesting this function with an EUT FRR set.
8	xxxx0823	<b>Equate Symbol:</b> lwmRsnCodeDatoff <b>Meaning:</b> Caller invoked service while DATOFF <b>Action:</b> Avoid requesting this function in this environment.
8	xxxx0824	<b>Equate Symbol:</b> lwmRsnCodeAmode24 <b>Meaning:</b> Caller invoked service but was in 24 bit addressing mode. <b>Action:</b> Request this function only when you are in 31 bit addressing mode.
8	xxxx0825	<b>Equate Symbol:</b> lwmRsnCodeAscModeNotPrimary <b>Meaning:</b> Caller invoked service but was not DAT on Primary ASC mode. <b>Action:</b> Avoid requesting this function in this environment.
8	xxxx0826	<b>Equate Symbol:</b> lwmRsnCodeTaskTerm <b>Meaning:</b> Caller invoked service while task termination is in progress for the current task. <b>Action:</b> Avoid requesting this function while task termination is in progress.
8	xxxx0828	<b>Equate Symbol:</b> lwmRsnCodeBadVersion <b>Meaning:</b> Version number in parameter list is not valid or version length field is not valid. <b>Action:</b> Check for possible storage overlay of the parameter list.

## IWMELEAV Macro

Table 33. Return and Reason Codes for the IWMELEAV Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx083A	<p><b>Equate Symbol:</b> IwmRsnCodeBadEnclave</p> <p><b>Meaning:</b> Enclave token does not pass verification.</p> <p><b>Action:</b> Check for possible storage overlay of the Enclave token.</p>
8	xxxx0845	<p><b>Equate Symbol:</b> IwmRsnCodeWrongEnclave</p> <p><b>Meaning:</b> The current dispatchable workunit is not associated with the input Enclave.</p> <p><b>Action:</b> Check for possible storage overlay of the Enclave token.</p>
8	xxxx0850	<p><b>Equate Symbol:</b> IwmRsnCodeBeginEnvOutstanding</p> <p><b>Meaning:</b> Current dispatchable workunit is operating under an outstanding Begin environment, Enclave leave is not allowed. IWM4STEN is the required operation.</p> <p><b>Action:</b> Avoid requesting this function in this environment.</p>
8	xxxx0858	<p><b>Equate Symbol:</b> IwmRsnCodeNotEjoinedTcb</p> <p><b>Meaning:</b> The current dispatchable workunit did not issue Enclave Join, but only inherited Enclave attribute from mother task.</p> <p><b>Action:</b> Avoid requesting this function in this environment.</p>
8	xxxx0859	<p><b>Equate Symbol:</b> IwmRsnCodeEnclaveSubTaskExists</p> <p><b>Meaning:</b> The current dispatchable workunit has residual subtasks propagated to the Enclave which are still associated with the Enclave.</p> <p><b>Action:</b> Avoid requesting this function in this environment.</p>
10	—	<p><b>Equate Symbol:</b> IwmRetCodeCompError</p> <p><b>Meaning:</b> Component error.</p> <p><b>Action:</b> Contact your system programmer.</p>

## Example

To cause the current task to leave its Enclave environment:

```

        IWMELEAV ENCTOKEN=ENCTOKEN,RETCODE=RC,RSNCODE=RSN
*
* Storage areas
*
ENCTOKEN DS    CL8           Contains the Enclave token
*                               associated with the work
*                               request as returned by IWM4ECRE
RC        DS    F            Return code
RSN       DS    F            Reason code

```

---

## Chapter 24. IWMEQTME – Enclave CPU Time Query

The purpose of this service is to return the enclave processor times if the current dispatchable work unit is associated with an enclave.

---

### Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Supervisor state. Any PSW key.
<b>Dispatchable unit mode:</b>	Task or SRB
<b>Cross memory mode:</b>	Non-XMEM or XMEM. Any PASN, HASN, SASN.
<b>AMODE:</b>	31-bit
<b>ASC mode:</b>	Primary
<b>Interrupt status:</b>	Enabled or disabled for I/O and external interrupts
<b>Locks:</b>	No locks are required.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

---

### Programming Requirements

1. The macro IWMYCON must be included to use this macro.
2. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
3. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, IWMRSNCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.
4. Caller is responsible for error recovery.

---

### Restrictions

This macro supports multiple versions. Some keywords are only supported by certain versions. Refer to the PLISTVER parameter description for further information.

---

### Input Register Information

Before issuing the IWMEQTME macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

---

### Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Reason code if GR15 return code is non-zero
1	Used as work registers by the system
2-13	Unchanged
14	Used as work registers by the system

## IWMEQTME Macro

15 Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

---

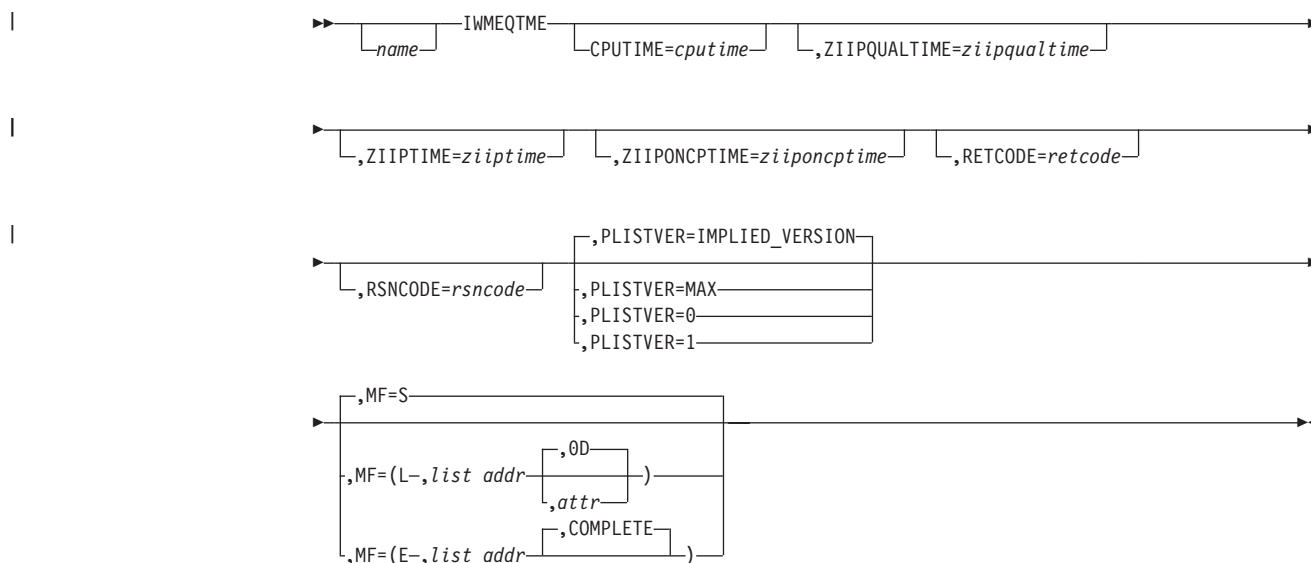
## Performance Implications

None.

---

## Syntax

### main diagram



---

## Parameters

The parameters are explained as follows:

### *name*

An optional symbol, starting in column 1, that is the name on the IWMEQTME macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

### **CPUTIME=cputime**

An optional output parameter, which will contain the total accumulated CPU time for an enclave, if the current dispatchable work unit is associated with an enclave. The CPU time will be in TOD clock format.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

**,MF=S**  
**,MF=(L,list addr)**  
**,MF=(L,list addr,attr)**  
**,MF=(L,list addr,OD)**  
**,MF=(E,list addr)**  
**,MF=(E,list addr,COMPLETE)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

*,list addr*

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

*,attr*

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

**,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

**,PLISTVER=IMPLIED\_VERSION**

**,PLISTVER=MAX**

**,PLISTVER=0**

**,PLISTVER=1**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form, when both are assembled with the same level of the system. In this way, MAX ensures that the parameter list does not overwrite nearby storage.

## IWMEQTME Macro

- **0**, which supports all parameters except those specifically referenced in higher versions.
- **1**, which supports both the following parameters and those from version 0:

ZIIPONCPTIME                      ZIIPQUALTIME                      ZIIPTIME

**To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX
- A decimal value of 0, or 1

**,RETCODE=*retcode***

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,RSNCODE=*rsncode***

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,ZIIPONCPTIME=*ziiponcptime***

An optional output parameter, which will contain the total accumulated time spent on a standard processor for zIIP-eligible work for the enclave that is associated with the current dispatchable work unit. The time will be in TOD clock format, normalized to standard processor speed.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

**,ZIIPQUALTIME=*ziipqualtime***

An optional output parameter, which will contain the total time the enclave that is associated with the current dispatchable work unit was qualified to run on an integrated information processor (zIIP). The time returned will be in TOD clock format, normalized to standard processor speed.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

**,ZIIPTIME=*ziiptime***

An optional output parameter, which will contain the total accumulated time spent on an integrated information processor (zIIP) for the enclave that is associated with the current dispatchable work unit. The zIIP time will be in TOD clock format, normalized to standard processor speed.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

---

## ABEND Codes

None.

---

## Return and Reason Codes

When the IWMEQTME macro returns control to your program:

- GPR 15 (and *retcode*, when you code RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, when you code RSNCODE) contains reason code.



The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 34. Return and Reason Codes for the IWMEQTME Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> lwmRetCodeOk <b>Meaning:</b> Successful completion. <b>Action:</b> None required.
4	—	<b>Equate Symbol:</b> lwmRetCodeWarning <b>Meaning:</b> Successful completion, unusual conditions noted.
4	xxxx041C	<b>Equate Symbol:</b> lwmRsnCodeNotEnclave <b>Meaning:</b> The current dispatchable workunit is not associated with an Enclave. <b>Action:</b> None required.
8	—	<b>Equate Symbol:</b> lwmRetCodeInvocError <b>Meaning:</b> Invalid invocation environment or parameters.
8	xxxx0828	<b>Equate Symbol:</b> lwmRsnCodeBadVersion <b>Meaning:</b> Version number in parameter list is not valid. <b>Action:</b> Check for possible storage overlay of the parameter list.



---

## Chapter 25. IWMEREG — WLM Enclave Register Service

The IWMEREG service allows the caller to register an enclave in order to prevent it from being deleted. This is useful if the caller wants to schedule SRBs or join tasks to an enclave that is owned by another subsystem. Registration guarantees that the enclave will continue to exist until the corresponding deregistration is done, even if the other subsystem deletes the enclave. The system defers the enclave's deletion until after the last deregistration.

The address space identified as the home address space at the time of registration is held responsible for deregistration in case of abnormal termination of the job step, the job, or the address space itself.

The caller can run in task or SRB mode.

---

### Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Supervisor state or program key mask (PKM) allowing keys 0-7.
<b>Dispatchable unit mode:</b>	Task or SRB
<b>Cross memory mode:</b>	Any PASN, any HASN, any SASN
<b>AMODE:</b>	31-bit
<b>ASC mode:</b>	Primary or access register (AR)
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks may be held.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

---

### Programming Requirements

1. The macro CVT must be included to use this macro.
2. The macro IWMYCON must be included to use this macro.
3. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
4. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, IWMSRNCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.
5. All character data, unless otherwise specified, is assumed to be left justified and padded with blanks on the right, as needed, to occupy the specified number of bytes.
6. Since this service may only be used by system-like code, some validity checking on the parameter list is not performed. These checks would only be needed if the macro were not used to invoke the service routine.

---

### Restrictions

None.

## Input Register Information

Before issuing the IWMEREG macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

## Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Reason code if GR15 return code is non-zero
1	Used as work registers by the system
2-13	Unchanged
14	Used as work registers by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

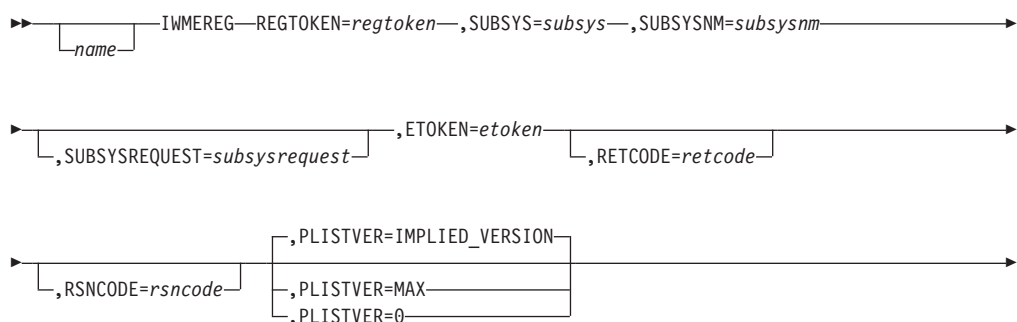
Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

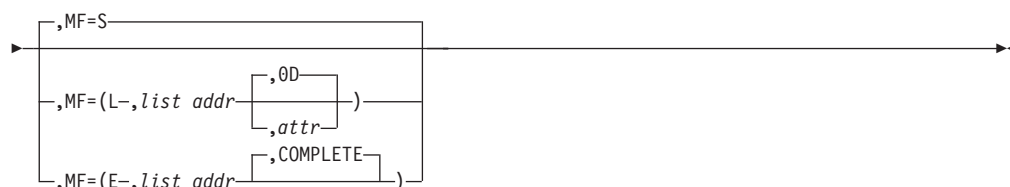
## Performance Implications

None

## Syntax

### main diagram





## Parameters

The parameters are explained as follows:

### *name*

An optional symbol, starting in column 1, that is the name on the IWMEREG macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

### **,ETOKEN=etoken**

A required input parameter that contains the enclave token.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

### **,MF=S**

### **,MF=(L,list addr)**

### **,MF=(L,list addr,attr)**

### **,MF=(L,list addr,0D)**

### **,MF=(E,list addr)**

### **,MF=(E,list addr,COMPLETE)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

### *,list addr*

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

### *,attr*

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

### **,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

### **,PLISTVER=IMPLIED\_VERSION**

### **,PLISTVER=MAX**

### **,PLISTVER=0**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form, when both are assembled with the same level of the system. In this way, MAX ensures that the parameter list does not overwrite nearby storage.

- **0**, if you use the currently available parameters.

**To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX
- A decimal value of 0

### **REGTOKEN=*regtoken***

A required output parameter that will receive the registration token

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

### **,RETCODE=*retcode***

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

### **,RSNCODE=*rsncode***

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

### **,SUBSYS=*subsys***

A required input parameter, which contains the generic subsystem type (e.g. IMS, CICS, etc.).

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 4-character field.

### **,SUBSYSNM=*subsysnm***

A required input parameter, which identifies the subsystem instance.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

### **,SUBSYSREQUEST=*subsysrequest***

An optional input parameter that allows the caller to pass additional information in order to distinguish between different invocations by the same subsystem.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32-character field.

---

## ABEND Codes

None.

---

## Return and Reason Codes

When the IWMEREG macro returns control to your program:

- GPR 15 (and *retcode*, when you code RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, when you code RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 35. Return and Reason Codes for the IWMEREG Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> <code>IwmRetCodeOk</code> <b>Meaning:</b> Successful completion. <b>Action:</b> None required.
8	—	<b>Equate Symbol:</b> <code>IwmRetCodeInvocError</code> <b>Meaning:</b> Invalid invocation environment or parameters.
8	xxxx0803	<b>Equate Symbol:</b> <code>IwmRsnCodeDisabled</code> <b>Meaning:</b> Caller is disabled. <b>Action:</b> Avoid requesting this function while disabled.
8	xxxx0804	<b>Equate Symbol:</b> <code>IwmRsnCodeLocked</code> <b>Meaning:</b> Caller is locked. <b>Action:</b> Avoid requesting this function while locked.
8	xxxx080B	<b>Equate Symbol:</b> <code>IwmRsnCodeBadPI</code> <b>Meaning:</b> Error accessing parameter list. <b>Action:</b> Check for possible storage overlay.
8	xxxx0824	<b>Equate Symbol:</b> <code>IwmRsnCodeAmode24</code> <b>Meaning:</b> Caller invoked service but was in 24 bit addressing mode. <b>Action:</b> Request this function only when you are in 31 bit addressing mode.
8	xxxx0828	<b>Equate Symbol:</b> <code>IwmRsnCodeBadVersion</code> <b>Meaning:</b> Version number in parameter list is not valid. <b>Action:</b> Check for possible storage overlay of the parameter list.

## IWMEREG Macro

Table 35. Return and Reason Codes for the IWMEREG Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx083A	<b>Equate Symbol:</b> IwmRsnCodeBadEnclave <b>Meaning:</b> Enclave token does not pass verification. <b>Action:</b> Check for possible storage overlay of the enclave token, or asynchronous events which may have deleted the enclave.
8	xxxx0882	<b>Equate Symbol:</b> IwmRsnCodeTooManyRegistrations <b>Meaning:</b> There are too many concurrent registrations requested. <b>Action:</b> There is a resource shortage. The function may work successfully at a later time.
10	—	<b>Equate Symbol:</b> IwmRetCodeCompError <b>Meaning:</b> Component error. <b>Action:</b> No action required. The function may be successful if invoked again.

---

## Example

```
IWMEREG ETOKEN=ENCTKN,REGTOKEN=REGTKN,
        SUBSYS=SUBSTYPE,SUBSYSNM=SUBSNAME
*
* Storage areas
*
ENCTKN  DS    CL8           Enclave token
SUBSNAME DS    CL8           Subsystem name
SUBSTYPE DS    CL4           Subsystem type
REGTKN  DS    CL8           Register token
```



---

## Chapter 26. IWMERES — Change an Enclave

The IWMERES macro allows the caller to change the performance controls for work associated with an independent enclave. The caller can:

- Change the service class of work currently in execution, with the SRVCLASS keyword. Resetting to a new service class also resumes quiesced work.
- Quiesce work currently in execution, with the QUIESCE keyword.
- Reclassify work currently in execution according to the service policy in effect, with the RESUME keyword. The RESUME keyword also resumes quiesced work.

The system does not allow foreign enclaves or dependent enclaves to be reset. Refer to “Restrictions” for further details.

---

### Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Supervisor state or program key mask (PKM) allowing keys 0-7.
<b>Dispatchable unit mode:</b>	Task or SRB
<b>Cross memory mode:</b>	PASN=HASN=SASN
<b>AMODE:</b>	31-bit
<b>ASC mode:</b>	Primary
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks may be held.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

---

### Programming Requirements

1. The macro CVT must be included to use this macro.
2. The macro IWMYCON must be included to use this macro.
3. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
4. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, IWMSRNCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.

---

### Restrictions

1. This macro may not be used prior to the completion of WLM address space initialization.
2. This macro may only be used on z/OS V1R3 or higher.
3. The caller cannot reset a dependent enclave. A dependent enclave can only be reset by resetting the address space that owns the enclave.
4. The caller cannot reset a foreign enclave. A foreign enclave can only be reset by resetting the original enclave on the originating system (foreign independent enclave) or by resetting the remote owner address space (foreign dependent enclave).
5. The caller cannot reset an enclave that is implicitly quiesced because one or more address space currently serving the enclave is quiesced. An address space is serving an enclave when any of its tasks is joined to the enclave or

## IWMERES Macro

when an SRB is scheduled to the enclave and the SRB used the ENCASSOC-sysevent to establish an association with the address space.

---

## Input Register Information

Before issuing the IWMERES macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

---

## Output Register Information

When control returns to the caller, the GPRs contain:

### Register

#### Contents

0	Reason code if GR15 return code is non-zero
1	Used as work registers by the system
2-13	Unchanged
14	Used as work registers by the system
15	Return code

When control returns to the caller, the ARs contain:

### Register

#### Contents

0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

---

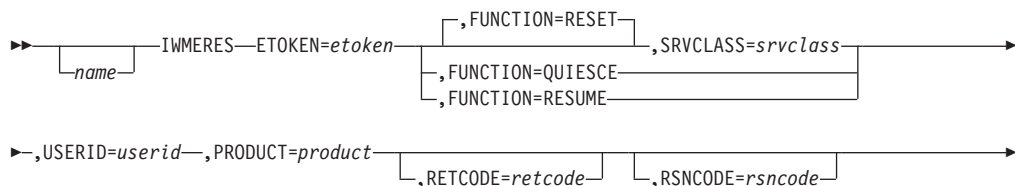
## Performance Implications

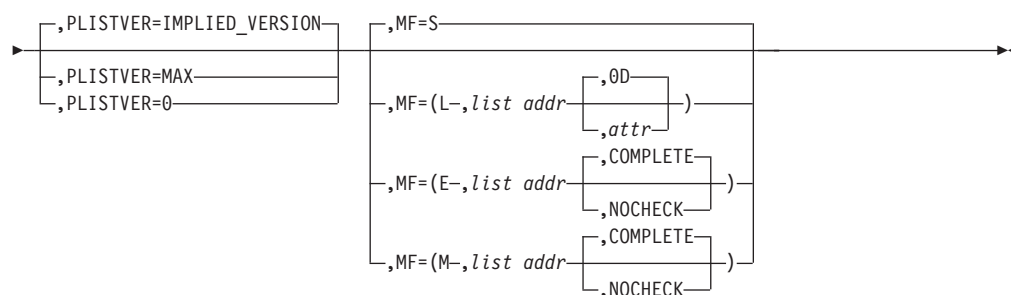
None.

---

## Syntax

### main diagram





## Parameters

The parameters are explained as follows:

### *name*

An optional symbol, starting in column 1, that is the name on the IWMERES macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

### **ETOKEN=etoken**

A required input parameter that contains the enclave token. The enclave token must represent an original independent enclave.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

### **,FUNCTION=RESET**

### **,FUNCTION=QUIESCE**

### **,FUNCTION=RESUME**

An optional parameter, which indicates the function to perform against the enclave. The default is FUNCTION=RESET.

#### **,FUNCTION=RESET**

Requests that the enclave's service class be changed.

#### **,FUNCTION=QUIESCE**

Requests that the enclave be quiesced.

#### **,FUNCTION=RESUME**

Requests that the enclave be reclassified according to the service policy in effect. This undoes a prior request to reset the enclave to a particular service class, or to quiesce the enclave.

### **,MF=S**

**,MF=(L,list addr)**

**,MF=(L,list addr,attr)**

**,MF=(L,list addr,OD)**

**,MF=(E,list addr)**

**,MF=(E,list addr,COMPLETE)**

**,MF=(E,list addr,NOCHECK)**

**,MF=(M,list addr)**

**,MF=(M,list addr,COMPLETE)**

**,MF=(M,list addr,NOCHECK)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

## IWMERES Macro

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

Use MF=M together with the list and execute forms of the macro for service routines that need to provide different options according to user-provided input. Use the list form to define a storage area; use the modify form to set the appropriate options; then use the execute form to call the service.

IBM recommends that you use the modify and execute forms of IWMERES in the following order:

- Use IWMERES ...MF=(M,list-addr,COMPLETE) specifying appropriate parameters, including all required ones.
- Use IWMERES ...MF=(M,list-addr,NOCHECK), specifying the parameters that you want to change.
- Use IWMERES ...MF=(E,list-addr,NOCHECK), to execute the macro.

*,list addr*

The name of a storage area to contain the parameters. For MF=S, MF=E, and MF=M, this can be an RS-type address or an address in register (1)-(12).

*,attr*

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

**,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

**,NOCHECK**

Specifies that the system is not to check for required parameters and is not to supply defaults for omitted optional parameters.

**,PLISTVER=IMPLIED\_VERSION**

**,PLISTVER=MAX**

**,PLISTVER=0**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify `PLISTVER=MAX` on the list form of the macro. Specifying `MAX` ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form, when both are assembled with the same level of the system. In this way, `MAX` ensures that the parameter list does not overwrite nearby storage.

- `0`, if you use the currently available parameters.

**To code:** Specify one of the following:

- `IMPLIED_VERSION`
- `MAX`
- A decimal value of `0`

**,PRODUCT=product**

A required input parameter, which contains the product name that is requesting the enclave be changed. The product name is included in the SMF 90 subtype 30 record created by IWMERES.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

**,RETCODE=retcode**

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,RSNCODE=rsncode**

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,SRVCLASS=svclass**

When `FUNCTION=RESET` is specified, a required input parameter, which is the service class to be assigned to the enclave. Resetting to a new service class also resumes quiesced work.

**To code:** Specify the RS-type address of an 8-character field.

**,USERID=userid**

A required input parameter, which contains the id of the user who is requesting the enclave be changed. The userid is included in the SMF 90 subtype 30 record created by IWMERES. If there is no userid available, the caller should pass blanks.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

## ABEND Codes

None.

## Return and Reason Codes

When the IWMERES macro returns control to your program:

- GPR 15 (and *retcode*, when you code `RETCODE`) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, when you code `RSNCODE`) contains reason code.

## IWMERES Macro

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value

Table 36. Return and Reason Codes for the IWMERES Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> IwmRetCodeOk <b>Meaning:</b> Successful completion. <b>Action:</b> None required.
4	—	<b>Equate Symbol:</b> IwmRetCodeWarning <b>Meaning:</b> Successful completion, unusual conditions noted.
4	xxxx043B	<b>Equate Symbol:</b> IwmRsnCodelsQuiesced <b>Meaning:</b> The enclave cannot be reset because an address space currently serving this enclave is quiesced. An address space is known to serve an enclave if any of its tasks is joined to the enclave or if an SRB is scheduled to the enclave and the SRB established an association with the address space by using the ENCASSOC-sysevent. <b>Action:</b> Retry the request when no address spaces serving the enclave is quiesced.
8	—	<b>Equate Symbol:</b> IwmRetCodeInvocError <b>Meaning:</b> Invalid invocation environment or parameters.
8	xxxx0803	<b>Equate Symbol:</b> IwmRsnCodeDisabled <b>Meaning:</b> Caller is disabled. <b>Action:</b> Avoid requesting this function while disabled.
8	xxxx0804	<b>Equate Symbol:</b> IwmRsnCodeLocked <b>Meaning:</b> Caller is locked. <b>Action:</b> Avoid requesting this function while locked.
8	xxxx080B	<b>Equate Symbol:</b> IwmRsnCodeBadPI <b>Meaning:</b> Error accessing parameter list. <b>Action:</b> Check for possible storage overlay.
8	xxxx0824	<b>Equate Symbol:</b> IwmRsnCodeAmode24 <b>Meaning:</b> Caller invoked service but was in 24 bit addressing mode. <b>Action:</b> Request this function only when you are in 31 bit addressing mode.
8	xxxx0825	<b>Equate Symbol:</b> IwmRsnCodeAscModeNotPrimary <b>Meaning:</b> Caller invoked service but was not in primary ASC mode. <b>Action:</b> Avoid requesting this function in this environment.
8	xxxx0827	<b>Equate Symbol:</b> IwmRsnCodeRsvdNot0 <b>Meaning:</b> Reserved field in parameter list was non-zero. <b>Action:</b> Check for use of keywords that are not supported by the MVS release on which the program is running.

Table 36. Return and Reason Codes for the IWMERES Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0828	<p><b>Equate Symbol:</b> lwmRsnCodeBadVersion</p> <p><b>Meaning:</b> Version number in parameter list is not valid.</p> <p><b>Action:</b> Check for possible storage overlay of the parameter list.</p>
8	xxxx0829	<p><b>Equate Symbol:</b> lwmRsnCodeBadOptions</p> <p><b>Meaning:</b> Parameter list omits required parameters.</p> <p><b>Action:</b> Check for possible storage overlay of the parameter list.</p>
8	xxxx083A	<p><b>Equate Symbol:</b> lwmRsnCodeBadEnclave</p> <p><b>Meaning:</b> Enclave token does not pass verification.</p> <p><b>Action:</b> Check for possible storage overlay of the enclave token, or asynchronous events which may have deleted the enclave.</p>
8	xxxx0841	<p><b>Equate Symbol:</b> lwmRsnCodeXMemMode</p> <p><b>Meaning:</b> Caller is in cross memory mode.</p> <p><b>Action:</b> Invoke the function in non-cross memory mode.</p>
8	xxxx0872	<p><b>Equate Symbol:</b> lwmRsnCodeForeignEnclave</p> <p><b>Meaning:</b> The requested service is not supported for a foreign enclave. This reason code is returned for independent foreign enclaves only.</p> <p><b>Action:</b> All participants of a multisystem enclave can only be reset together by resetting the original enclave on the originating system.</p>
8	xxxx0885	<p><b>Equate Symbol:</b> lwmRsnCodeDependentEnclave</p> <p><b>Meaning:</b> The requested service is not supported for a dependent enclave. This reason code is returned for both, an original and a foreign dependent enclave.</p> <p><b>Action:</b> A dependent enclave cannot be reset directly. It can only be reset by resetting its owning address space.</p>
C	—	<p><b>Equate Symbol:</b> lwmRetCodeEnvError</p> <p><b>Meaning:</b> Environmental error.</p>
C	xxxx0C28	<p><b>Equate Symbol:</b> lwmRsnCodeBadServiceClass</p> <p><b>Meaning:</b> The input service class name is not defined in the active workload manager policy.</p> <p><b>Action:</b> Record or report the error if appropriate.</p>
C	xxxx0C2E	<p><b>Equate Symbol:</b> lwmRsnCodeWrongMode</p> <p><b>Meaning:</b> Reserved</p>
C	xxxx0C32	<p><b>Equate Symbol:</b> lwmRsnCodeNotEligibleForSrvClass</p> <p><b>Meaning:</b> The active job in the specified address space or the specified enclave is not eligible for reset into the specified system service class. Only address spaces created with the ASCRE HIPRI attribute are eligible for reset into the SYSTEM service class.</p> <p><b>Action:</b> Record or report the error if appropriate.</p>

## IWMERES Macro

Table 36. Return and Reason Codes for the IWMERES Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
10	—	<b>Equate Symbol:</b> IwmRetCodeCompError <b>Meaning:</b> Component error. <b>Action:</b> Contact your system programmer.

---

## Example

To change the service class of the enclave identified by the token TOKEN:

```
IWMERES ETOKEN=TOKEN,SRVCLASS=SCNAME,USERID=USR,
        PRODUCT=PROD
*
* Storage areas
*
TOKEN   DS   CL8           Contains the enclave token
SCNAME  DS   CL8           Contains the service class name
*                               to assign to the job
USR     DS   CL8           Contains the id of the user who
*                               is requesting the change
PROD    DS   CL8           Contains the product name of
*                               the code invoking IWMERES
```



---

## Chapter 27. IWMESTOP – Stopping a Work Request

The IWMESTOP service allows work managers that participate in cross-platform enterprise workload management (EWLM) to explicitly indicate the stop of an EWLM work request. Using this service the application work request model is decoupled from WLM's enclave model.

The counterpart of this service to indicate the start of an EWLM work request is IWMESTRT.

---

### Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Supervisor state or program key mask (PKM) allowing keys 0-7
<b>Dispatchable unit mode:</b>	Task or SRB
<b>Cross memory mode:</b>	Any PASN, any HASN, any SASN
<b>AMODE:</b>	31- and 64 bit
<b>ASC mode:</b>	Primary or access register (AR)
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks may be held.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

---

### Programming Requirements

1. The macro CVT must be included to use this macro.
2. The macro IWMYCON must be included to use this macro.
3. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
4. Note that the high-order halfword of 31-bit register 0, and the reason code variable when specified, may be non-zero and represent diagnostic data which is not part of the external interface. The high-order halfword should thus be excluded from comparison with the reason code values described above. The constant, IWMRSNCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.
5. All character data, unless otherwise specified, is assumed to be left justified and padded with blanks on the right, as needed, to occupy the specified number of bytes.
6. Since this service may only be used by system-like code, some validity checking on the parameter list is not performed. These checks would only be needed if the macro were not used to invoke the service routine.
7. If the parameter EWLMODE=EXPLICIT\_SINGLE is specified, the following restrictions apply:
  - No other task or SRB is allowed to issue other enclave services for the same enclave concurrently.
  - The caller must be in primary ASC mode before invocation.
  - The parameter list, the classification parameters and the savearea pointed to by GPR13 must be addressable in AMODE 31 and primary ASC mode.

## IWMESTOP Macro

- No recovery environment is set up by the service. The caller is responsible to provide an appropriate error recovery environment to handle abnormal terminations.
- The enclave must have been created with option ESTRT=EXPLICIT\_SINGLE on the IWM4ECRE (or IWMECREA) invocation.
- The caller must provide the ETOKEN parameter.

---

## Restrictions

None.

---

## Input Register Information

If the EWLMMODE=EXPLICIT\_SINGLE parameter is specified, the caller must provide a standard 72-byte savearea pointed to by GPR13.

For all other cases the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

---

## Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Reason code if GR15 return code is non-zero
1	Used as work registers by the system
2-13	Unchanged
14	Used as work registers by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

---

## Performance Implications

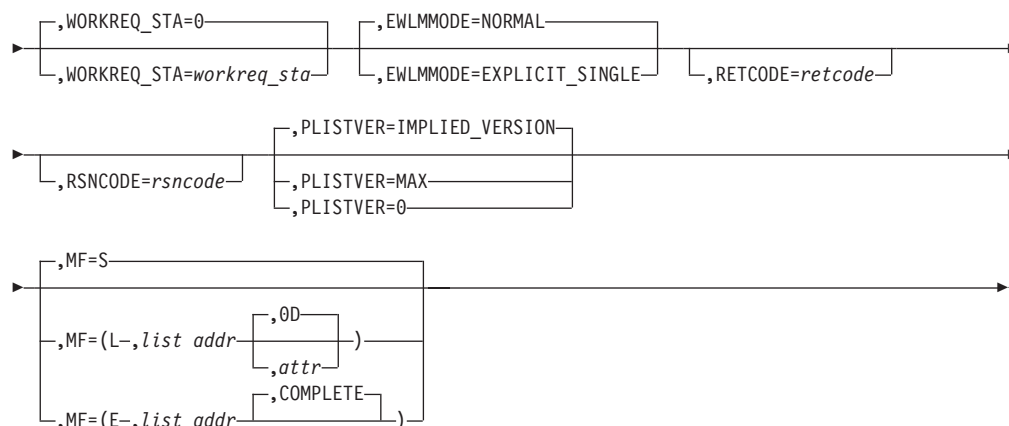
None.

---

## Syntax

### main diagram

▶ `name` IWMESTOP—ETOKEN=*etoken*—WORKREQ\_HDL=*workreq\_hdl* ▶



## Parameters

The parameters are explained as follows:

### *name*

An optional symbol, starting in column 1, that is the name on the IWMESTOP macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

### **ETOKEN=etoken**

A required input parameter that contains the enclave token of the enclave under which the work request is processed.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

### **,EWLMMODE=NORMAL**

### **,EWLMMODE=EXPLICIT\_SINGLE**

An optional input keyword that indicates if the parameter EXPLICIT\_SINGLE has been specified on the IWM4ECRE (or IWMECREA) call. Note that EWLMMODE=EXPLICIT\_SINGLE can only be specified if the enclave was created with the ESTRT=EXPLICIT\_SINGLE option. The values of EWLMMODE are:

- **EWLMMODE=NORMAL** indicates that the enclave was created with the ESTRT=EXPLICIT or with the ESTRT=IMPLIED option. EWLMMODE=NORMAL is the default.
- **EWLMMODE=EXPLICIT\_SINGLE** indicates that the enclave was created with the ESTRT=EXPLICIT\_SINGLE option which can only be used for a restricted environment. The caller must provide a standard 72-byte savearea, which is addressable in AMODE 31 and pointed to by GPR13. The parameter list and the classification parameters must also be addressable in AMODE 31.

### **,MF=S**

**,MF=(L,list addr)**

**,MF=(L,list addr,attr)**

**,MF=(L,list addr,OD)**

**,MF=(E,list addr)**

**,MF=(E,list addr,COMPLETE)**

An optional input parameter that specifies the macro form.

## IWMESTOP Macro

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

*,list addr*

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

*,attr*

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

**,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

**,PLISTVER=IMPLIED\_VERSION**

**,PLISTVER=MAX**

**,PLISTVER=0**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form, when both are assembled with the same level of the system. In this way, MAX ensures that the parameter list does not overwrite nearby storage.

- **0**, if you use the currently available parameters.

**To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX
- A decimal value of 0

**,RETCODE=retcode**

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,RSNCODE=rsncode**

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**WORKREQ\_HDL=workreq\_hdl**

A required input parameter that contains the handle which represents the work request. This handle was returned from a previous call to IWMESTRT or IWM4ECRE.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

**,WORKREQ\_STA=workreq\_sta****,WORKREQ\_STA=0**

An optional input parameter, which contains the completion status code of the work request. Available completion status codes are described in the EWLM ARM interface specification. The use of symbolic constants `IwmEwlmArmStatusGood`, `IwmEwlmArmStatusAborted`, `IwmEwlmArmStatusFailed`, `IwmEwlmArmStatusUnknown` (defined in macro `IWMYCON`) is recommended.

The default is 0. This indicates successful completion.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

---

## ABEND Codes

None.

---

## Return and Reason Codes

When the IWMESTOP macro returns control to your program:

- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, if you coded RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 37. Return and Reason Codes for the IWMESTOP Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> <code>IwmRetCodeOk</code> <b>Meaning:</b> Successful completion. <b>Action:</b> None required.
8	—	<b>Equate Symbol:</b> <code>IwmRetCodeInvocError</code> <b>Meaning:</b> Invalid invocation environment or parameters.

Table 37. Return and Reason Codes for the IWMESTOP Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0803	<b>Equate Symbol:</b> IwmRsnCodeDisabled <b>Meaning:</b> Caller is disabled. <b>Action:</b> Avoid requesting this function while disabled.
8	xxxx0804	<b>Equate Symbol:</b> IwmRsnCodeLocked <b>Meaning:</b> Caller is locked. <b>Action:</b> Avoid requesting this function while locked.
8	xxxx080B	<b>Equate Symbol:</b> IwmRsnCodeBadPI <b>Meaning:</b> Error accessing parameter list. <b>Action:</b> Check for possible storage overlay.
8	xxxx0824	<b>Equate Symbol:</b> IwmRsnCodeAmode24 <b>Meaning:</b> Caller invoked service but was in 24 bit addressing mode. <b>Action:</b> Request this function only when you are in 31 bit addressing mode.
8	xxxx0828	<b>Equate Symbol:</b> IwmRsnCodeBadVersion <b>Meaning:</b> Version number in parameter list is not valid. <b>Action:</b> Check for possible storage overlay of the parameter list.
8	xxxx083A	<b>Equate Symbol:</b> IwmRsnCodeBadEnclave <b>Meaning:</b> Enclave token does not pass verification. <b>Action:</b> Check for possible storage overlay of the enclave token, or for asynchronous events which may have deleted the enclave.
8	xxxx0885	<b>Equate Symbol:</b> IwmRsnCodeDependentEnclave <b>Meaning:</b> Service is not available for dependent enclaves. <b>Action:</b> Avoid requesting this function for dependent enclaves.
8	xxxx0895	<b>Equate Symbol:</b> IwmRsnCodeEWLMServNotEnabled <b>Meaning:</b> Service is not enabled because caller invoked the IWM4CON (IWMCONN) service with EWLM=NO. <b>Action:</b> Ensure that EWLM=YES is specified on the IWM4CON request to enable this service.
8	xxxx0896	<b>Equate Symbol:</b> IwmRsnCodeBadWorkReqHandle <b>Meaning:</b> Work request handle is not associated with passed enclave <b>Action:</b> Check the specification of the WORKREQ_HDL parameter.
8	xxxx0897	<b>Equate Symbol:</b> IwmRsnCodeTranStatusInvalid <b>Meaning:</b> Passed work request completion status is not valid. <b>Action:</b> Check the EWLM ARM interface specification for valid completion status values.

Table 37. Return and Reason Codes for the IWMESTOP Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx08A0	<p><b>Equate Symbol:</b> IwmRsnCodeNotExplicitSingle</p> <p><b>Meaning:</b> The service has been invoked with option EWLMMODE=EXPLICIT_SINGLE but the enclave has not been created with the ESTRT=EXPLICIT_SINGLE option or vice versa.</p> <p><b>Action:</b> If ESTRT=EXPLICIT_SINGLE was coded on the IWM4ECRE or IWMECREA call, EWLMMODE=EXPLICIT_SINGLE must also be specified on the IWMEBLK call. If ESTRT=EXPLICIT OR ESTRT=IMPLIED was coded on the IWM4ECRE or IWMECREA call, EWLMMODE=NORMAL must be specified on the IWMEBLK call (or the EWLMMODE parameter can be omitted).</p>
10	—	<p><b>Equate Symbol:</b> IwmRetCodeCompError</p> <p><b>Meaning:</b> Component error.</p> <p><b>Action:</b> No action required. The function may be successful if invoked again.</p>

## Example

```

IWMESTOP ETOKEN=ENCTKN,WORKREQ_HDL=WRHANDLE,
          WORKREQ_STA=WRSTATUS
*
* Storage areas
*
ENCTKN   DS   CL8           Enclave token
WRHANDLE DS   CL8           Work Request Handle
WRSTATUS DS   CL4           Work Request Completion Status

```

## IWMESTOP Macro



---

## Chapter 28. IWMESTRT – Starting a Work Request

The IWMESTRT service allows work managers that participate in cross-platform enterprise workload management (EWLM) to explicitly indicate the start of an EWLM work request. Using this service the application work request model is decoupled from WLM's enclave model.

The counterpart of this service to indicate the end of an EWLM work request is IWMESTOP.

---

### Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Supervisor state or program key mask (PKM) allowing keys 0-7
<b>Dispatchable unit mode:</b>	Task or SRB
<b>Cross memory mode:</b>	Any PASN, any HASN, any SASN
<b>AMODE:</b>	31- and 64-bit
<b>ASC mode:</b>	Primary or access register (AR)
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks may be held.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

---

### Programming Requirements

1. The macro CVT must be included to use this macro.
2. The macro IWMYCON must be included to use this macro.
3. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
4. Note that the high order halfword of 31-bit register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, IWMRSNCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.
5. All character data, unless otherwise specified, is assumed to be left justified and padded with blanks on the right, as needed, to occupy the specified number of bytes.
6. Since this service may only be used by system-like code, some validity checking on the parameter list is not performed. These checks would only be needed if the macro were not used to invoke the service routine.
7. If the parameter EWLMODE=EXPLICIT\_SINGLE is specified, the following restrictions apply:
  - No other task or SRB is allowed to issue other enclave services for the same enclave concurrently.
  - The caller must be in primary ASC mode before invocation.
  - The parameter list, and the savearea pointed to by GPR13 must be addressable in AMODE 31 and primary ASC mode.

## IWMESTRT Macro

- No recovery environment is set up by the service. The caller is responsible to provide an appropriate error recovery environment to handle abnormal terminations.
- The enclave must have been created with option ESTRT=EXPLICIT\_SINGLE on the IWM4ECRE (or IWMECREA) invocation.
- All data in the optional classify parameter list (CLSFY=xxx) are ignored, including EWLM\_CORR. If the application wants to specify an EWLM parent correlator or classification attributes for an EWLM hop0 work request, these data have to be passed in the classification parameter list on the IWM4ECRE (IWMECREA) call and thus will be the same for all work requests on the enclave.

---

## Restrictions

None.

---

## Input Register Information

If the EWLMODE=EXPLICIT\_SINGLE parameter is specified, the caller must provide a standard 72-byte savearea pointed to by GPR13.

For all other cases the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

---

## Output Register Information

When control returns to the caller, the GPRs contain:

<b>Register</b>	<b>Contents</b>
<b>0</b>	Reason code if GR15 return code is non-zero
<b>1</b>	Used as work registers by the system
<b>2-13</b>	Unchanged
<b>14</b>	Used as work registers by the system
<b>15</b>	Return code

When control returns to the caller, the ARs contain:

<b>Register</b>	<b>Contents</b>
<b>0-1</b>	Used as work registers by the system
<b>2-13</b>	Unchanged
<b>14-15</b>	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

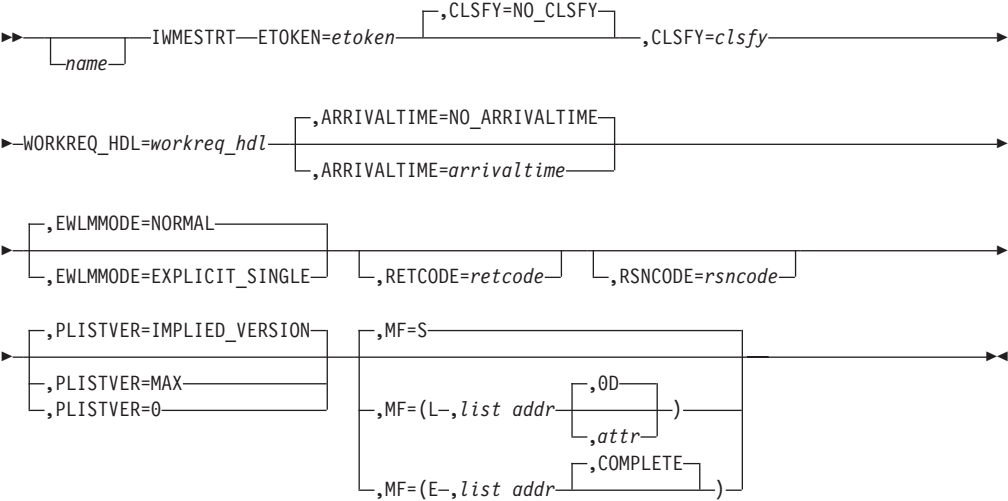
---

## Performance Implications

None.

Syntax

main diagram



Parameters

The parameters are explained as follows:

*name*

An optional symbol, starting in column 1, that is the name on the IWMESTRT macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

**,ARRIVALTIME=arrivaltime**  
**,ARRIVALTIME=NO\_ARRIVALTIME**

An optional input parameter, which contains the time when the work request actually started. This time is used to calculate the response time of the work request. The format of the field is STCK. The default is NO\_ARRIVALTIME. This indicates that no arrival time is passed.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 64-bit field.

**,CLSFY=clsfy**  
**,CLSFY=NO\_CLSFY**

An optional input parameter, which contains the classification information in the format of the parameter list for IWMCLSFY. This name is the data area name, not its pointer. IWMCLSFY MF(M) should be used to initialize the area prior to invocation of IWMESTRT. If the EWLM\_CORR field within that classification information is non-zero, a sub-work request is started. The classify parameter list must be from z/OS V1R5 or higher.

The variable length fields associated with the classify parameter list given by the CLSFY keyword have the following limitations in addition to those documented in IWMCLSFY:

- SUBSYSPM is limited to 255 bytes
- COLLECTION is limited to 18 bytes
- CORRELATION is limited to 12 bytes

## IWMESTRT Macro

**To code:** Specify the RS-type address, or address in register (2)-(12), of a character field.

**Note:** If EWLMODE=EXPLICIT\_SINGLE is specified, all data in the optional classify parameter list (CLSFY=xxx) is ignored, including the EWLM\_CORR. If the application wants to specify an EWLM parent correlator or classification attributes for an EWLM hop0 work request, these data have to be passed in the classification parameter list on the IWM4ECRE (IWMECREA) call and thus will be the same for all work requests on the enclave.

CLSFY=NO\_CLSFY indicates that no classify parameter list is passed. This is the default.

**ETOKEN=etoken**

A required input parameter that contains the enclave token of the enclave under which the work request is processed.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

**,EWLMODE=NORMAL**

**,EWLMODE=EXPLICIT\_SINGLE**

An optional input keyword that indicates whether the parameter EXPLICIT\_SINGLE has been specified on the IWM4ECRE (or IWMECREA) call. Note that EWLMODE=EXPLICIT\_SINGLE can only be specified if the enclave was created with the ESTRT=EXPLICIT\_SINGLE option. The values of EWLMODE are:

- **EWLMODE=NORMAL** indicates that the enclave was created with the ESTRT=EXPLICIT or with the ESTRT=IMPLIED option. EWLMODE=NORMAL is the default.
- **EWLMODE=EXPLICIT\_SINGLE** indicates that the enclave was created with the ESTRT=EXPLICIT\_SINGLE option which can only be used for a restricted environment. The caller must provide a standard 72-byte savearea, which is addressable in AMODE 31 and pointed to by GPR13. The parameter list and the classification parameters must also be addressable in AMODE 31.

**,MF=S**

**,MF=(L,list addr)**

**,MF=(L,list addr,attr)**

**,MF=(L,list addr,OD)**

**,MF=(E,list addr)**

**,MF=(E,list addr,COMPLETE)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant

code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

*,list addr*

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

*,attr*

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

**,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

**,PLISTVER=IMPLIED\_VERSION**

**,PLISTVER=MAX**

**,PLISTVER=0**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form, when both are assembled with the same level of the system. In this way, MAX ensures that the parameter list does not overwrite nearby storage.

- **0**, if you use the currently available parameters.

**To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX
- A decimal value of 0

**,RETCODE=retcode**

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,RSNCODE=rsncode**

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**WORKREQ\_HDL=workreq\_hdl**

A required output parameter that will receive the handle which represents the

## IWMESTRT Macro

work request. The application must pass this handle to the other work request services IWMESTOP, IWMEBLK, IWMEUBLK, and IWMEGCOR.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

---

## ABEND Codes

None.

---

## Return and Reason Codes

When the IWMESTRT macro returns control to your program:

- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, if you coded RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 38. Return and Reason Codes for the IWMESTRT Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> <code>IwmRetCodeOk</code> <b>Meaning:</b> Successful completion. <b>Action:</b> None required.
8	—	<b>Equate Symbol:</b> <code>IwmRetCodeInvocError</code> <b>Meaning:</b> Invalid invocation environment or parameters.
8	xxxx0803	<b>Equate Symbol:</b> <code>IwmRsnCodeDisabled</code> <b>Meaning:</b> Caller is disabled. <b>Action:</b> Avoid requesting this function while disabled.
8	xxxx0804	<b>Equate Symbol:</b> <code>IwmRsnCodeLocked</code> <b>Meaning:</b> Caller is locked. <b>Action:</b> Avoid requesting this function while locked.
8	xxxx080B	<b>Equate Symbol:</b> <code>IwmRsnCodeBadPI</code> <b>Meaning:</b> Error accessing parameter list. <b>Action:</b> Check for possible storage overlay.
8	xxxx0824	<b>Equate Symbol:</b> <code>IwmRsnCodeAmode24</code> <b>Meaning:</b> Caller invoked service but was in 24 bit addressing mode. <b>Action:</b> Request this function only when you are in 31 bit addressing mode.
8	xxxx0828	<b>Equate Symbol:</b> <code>IwmRsnCodeBadVersion</code> <b>Meaning:</b> Version number in parameter list is not valid. <b>Action:</b> Check for possible storage overlay of the parameter list.

Table 38. Return and Reason Codes for the IWMESTRT Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0838	<p><b>Equate Symbol:</b> IwmRsnCodeCisfyAreaTooBig</p> <p><b>Meaning:</b> Input area associated with classification information is larger than supported.</p> <p><b>Action:</b> Invoke the function with an area of the proper size. Check for possible storage overlay.</p>
8	xxxx0839	<p><b>Equate Symbol:</b> IwmRsnCodeCisfyPITooSmall</p> <p><b>Meaning:</b> Input Classify parameter list is too small.</p> <p><b>Action:</b> Invoke the function with an area of the proper size. Check for possible storage overlay.</p>
8	xxxx083A	<p><b>Equate Symbol:</b> IwmRsnCodeBadEnclave</p> <p><b>Meaning:</b> Enclave token does not pass verification.</p> <p><b>Action:</b> Check for possible storage overlay of the enclave token, or asynchronous events which may have deleted the enclave.</p>
8	xxxx0885	<p><b>Equate Symbol:</b> IwmRsnCodeDependentEnclave</p> <p><b>Meaning:</b> Service is not available for dependent enclaves.</p> <p><b>Action:</b> Avoid requesting this function for dependent enclaves.</p>
8	xxxx0893	<p><b>Equate Symbol:</b> IwmRsnCodeMissingEWLMCorr</p> <p><b>Meaning:</b> Passed classification information must contain an EWLM correlator (EWLM_CORR).</p> <p><b>Action:</b> Provide an EWLM correlator (EWLM_CORR) within the passed classification information.</p>
8	xxxx0894	<p><b>Equate Symbol:</b> IwmRsnCodeInvalidEWLMCorr</p> <p><b>Meaning:</b> Passed classification information contains an EWLM correlator (EWLM_CORR) that does not pass validity checking or that is not associated with the enclave.</p> <p><b>Action:</b> Check the specification of the EWLM correlator in the classification information.</p>
8	xxxx0895	<p><b>Equate Symbol:</b> IwmRsnCodeEWLMSErvNotEnabled</p> <p><b>Meaning:</b> Service is not enabled because caller invoked the IWM4CON service with EWLM=NO.</p> <p><b>Action:</b> Ensure that EWLM=YES is specified on the IWM4CON request to enable this service.</p>
8	xxxx089F	<p><b>Equate Symbol:</b> IwmRsnCodeMoreThanOneStart</p> <p><b>Meaning:</b> The service has been invoked with the EWLMODE=EXPLICIT_SINGLE option, but a work request is already active for the enclave. For EWLMODE=EXPLICIT_SINGLE only one work request may be active at a given point in time.</p> <p><b>Action:</b> Ensure that the IWMESTRT and IWMESTOP invocations are correctly paired and that only one work request is active.</p>

## IWMESTRT Macro

Table 38. Return and Reason Codes for the IWMESTRT Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx08A0	<p><b>Equate Symbol:</b> IwmRsnCodeNotExplicitSingle</p> <p><b>Meaning:</b> The service has been invoked with option EWLMODE=EXPLICIT_SINGLE, but the enclave has not been created with the ESTRT=EXPLICIT_SINGLE option or vice versa.</p> <p><b>Action:</b> If ESTRT=EXPLICIT_SINGLE was coded on the IWM4ECRE (or IWMECREA) call, EWLMODE=EXPLICIT_SINGLE must also be specified on the IWMESTRT call. If ESTRT=EXPLICIT OR ESTRT=IMPLIED was coded on the IWM4ECRE (or IWMECREA) call, EWLMODE=NORMAL must be specified on the IWMESTRT call (or the EWLMODE parameter can be omitted).</p>
10	—	<p><b>Equate Symbol:</b> IwmRetCodeCompError</p> <p><b>Meaning:</b> Component error.</p> <p><b>Action:</b> No action required. The function may be successful if invoked again.</p>

## Example

```

* Before calling IWMESTRT initialize classification
* attributes in CLFSY
*
      IWMESTRT ETOKEN=ENCTKN,CLSFY=CLFSY,
              WORKREQ_HDL=WRHANDLE,ARRIVALTIME=ATIME
*
* Storage areas
*
      IWMCLSFY PLISTVER=MAX,MF=(L,CLFSY,0D)
ENCTKN  DS    CL8           Enclave token
ATIME   DS    CL8           Arrival time
WRHANDLE DS    CL8           Work Request Handle

```



---

## Chapter 29. IWMESSQRY – Query Enclave State

The purpose of this service is to query whether or not the current dispatchable workunit is associated with an enclave. The output either is the enclave token or the STOKEN of the address space associated with the caller's workunit.

---

### Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Supervisor state. Any PSW key.
<b>Dispatchable unit mode:</b>	Task or SRB
<b>Cross memory mode:</b>	Non-XMEM or XMEM. Any PASN, HASN, SASN.
<b>AMODE:</b>	31-bit
<b>ASC mode:</b>	Primary
<b>Interrupt status:</b>	Enabled or disabled for I/O and external interrupts
<b>Locks:</b>	No locks are required.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

---

### Programming Requirements

1. The macro IWMYCON must be included to use this macro.
2. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
3. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, IWMRSCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.
4. Caller is responsible for error recovery.

---

### Restrictions

None.

---

### Input Register Information

Before issuing the IWMESSQRY macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

---

### Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Reason code if GR15 return code is non-zero
1	Used as work registers by the system
2-13	Unchanged
14	Used as work registers by the system
15	Return code

## IWMESQRY Macro

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

---

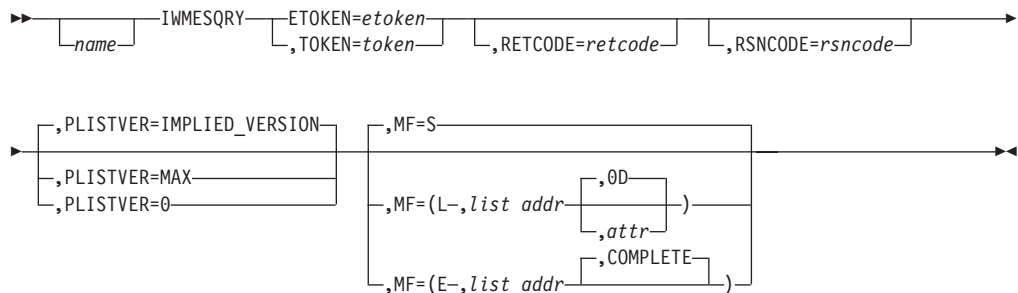
## Performance Implications

None.

---

## Syntax

### main diagram



---

## Parameters

The parameters are explained as follows:

### *name*

An optional symbol, starting in column 1, that is the name on the IWMESQRY macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

### ETOKEN=*etoken*

A required output parameter, which will receive the Enclave token if the current dispatchable workunit is associated with an Enclave. If it is not associated with an Enclave, the field is set to 0. This parameter is deprecated and supported for compatibility reasons only. The TOKEN parameter should be used instead.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

### ,MF=S

,MF=(*L*,*list addr*)

,MF=(*L*,*list addr*,*attr*)

,MF=(*L*,*list addr*,0D)

,MF=(*E*,*list addr*)

,MF=(*E*,*list addr*,COMPLETE)

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

*,list addr*

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

*,attr*

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

**,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

**,PLISTVER=IMPLIED\_VERSION**

**,PLISTVER=MAX**

**,PLISTVER=0**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form; in this way, MAX ensures that the parameter list does not overwrite nearby storage.

- **0**, if you use the currently available parameters.

**To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX
- A decimal value of 0

**,RETCODE=*retcode***

An optional output parameter into which the return code is to be copied from GPR 15.

## IWMESQRY Macro

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,RSNCODE=*rsncode***

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,TOKEN=*token***

A required output parameter which will receive either the Enclave token if the current dispatchable workunit is associated with an Enclave (as indicated by return code 0), or the STOKEN of the address space the workunit is associated with (return code 4).

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

---

## ABEND Codes

None.

---

## Return and Reason Codes

When the IWMESQRY macro returns control to your program:

- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, if you coded RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 39. Return and Reason Codes for the IWMESQRY Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> <code>IwmRetCodeOk</code> <b>Meaning:</b> Successful completion. <b>Action:</b> None required.
4	—	<b>Equate Symbol:</b> <code>IwmRetCodeWarning</code> <b>Meaning:</b> Successful completion, unusual conditions noted.
4	xxxx041C	<b>Equate Symbol:</b> <code>IwmRsnCodeNotEnclave</code> <b>Meaning:</b> The current dispatchable workunit is not associated with an Enclave. <b>Action:</b> None required.

---

## Example

To query whether the current dispatchable workunit is associated with an Enclave or an address space:

```
        IWMESQRY TOKEN=MYTOKEN,RETCODE=RC,RSNCODE=RSN
*
* Storage areas
*
MYTOKEN DS    CL8           Enclave token or STOKEN
RC       DS    F            Return code
RSN      DS    F            Reason code
```



---

## Chapter 30. IWMEUBLK – Work Request No Longer Blocked

The IWMEUBLK service allows work managers that participate in cross-platform enterprise workload management (EWLM) to indicate that processing of a work request is no longer blocked.

The counterpart of this service to indicate that the processing of a work request is blocked is IWMEBLK.

---

### Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Supervisor state or program key mask (PKM) allowing keys 0-7
<b>Dispatchable unit mode:</b>	Task or SRB
<b>Cross memory mode:</b>	Any PASN, any HASN, any SASN
<b>AMODE:</b>	31- or 64-bit
<b>ASC mode:</b>	Primary or access register (AR)
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks may be held.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

---

### Programming Requirements

1. The macro CVT must be included to use this macro.
2. The macro IWMYCON must be included to use this macro.
3. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
4. Note that the high order halfword of 31-bit register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, IWMRNSCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.
5. All character data, unless otherwise specified, is assumed to be left justified and padded with blanks on the right, as needed, to occupy the specified number of bytes.
6. Since this service may only be used by system-like code, some validity checking on the parameter list is not performed. These checks would only be needed if the macro were not used to invoke the service routine.
7. If the parameter EWLMODE=EXPLICIT\_SINGLE is specified, the following restrictions apply:
  - No other task or SRB is allowed to issue other enclave services for the same enclave concurrently.
  - The caller must be in primary ASC mode before invocation.
  - The parameter list, the classification parameters and the savearea pointed to by GPR13 must be addressable in AMODE 31 and primary ASC mode.
  - No recovery environment is set up by the service. The caller is responsible to provide an appropriate error recovery environment to handle abnormal terminations.

## IWMEUBLK Macro

- The enclave must have been created with option ESTRT=EXPLICIT\_SINGLE on the IWM4ECRE (or IWMECREA) invocation.
- The caller must provide the ETOKEN parameter.

---

## Restrictions

None.

---

## Input Register Information

If the EWLMMODE=EXPLICIT\_SINGLE parameter is specified, the caller must provide a standard 72-byte savearea pointed to by GPR13.

For all other cases the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

---

## Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Reason code if GR15 return code is non-zero
1	Used as work registers by the system
2-13	Unchanged
14	Used as work registers by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

---

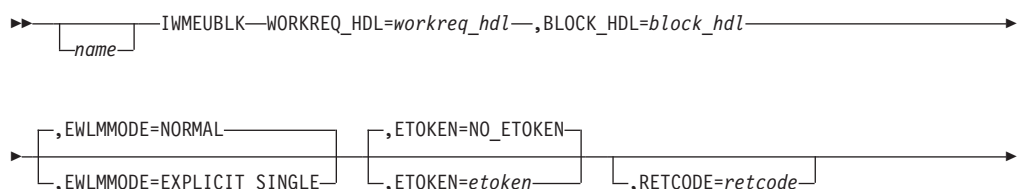
## Performance Implications

None.

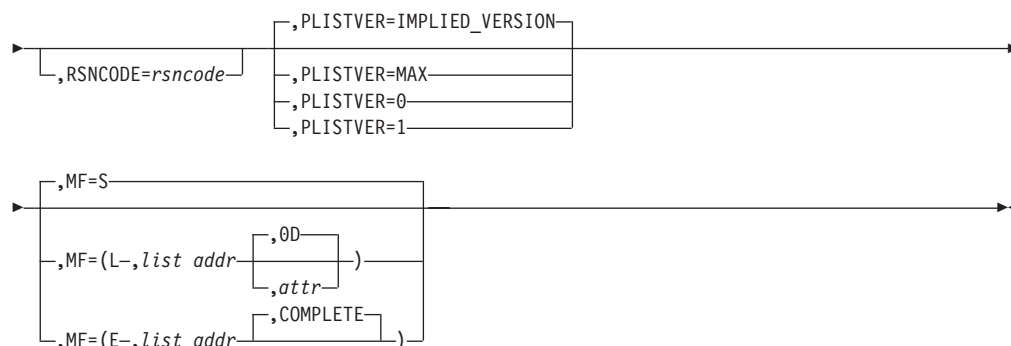
---

## Syntax

### main diagram







## Parameters

The parameters are explained as follows:

### *name*

An optional symbol, starting in column 1, that is the name on the IWMEUBLK macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

### **,BLOCK\_HDL=block\_hdl**

A required output parameter that will receive the handle identifying the blocked work request. This handle was returned from a previous call to IWMEBLK.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

### **ETOKEN=etoken**

### **ETOKEN=NO\_ETOKEN**

Is the name (RS-type), or address in register (2)-(12), of an optional 8-character input variable that contains the enclave token of the enclave under which the work request is processed. The values of ETOKEN are:

- **etoken** is required, if the option EWLMMODE=EXPLICIT\_SINGLE is specified.
- **NO\_ETOKEN** indicates that no ETOKEN is passed. This is the default.

### **,EWLMMODE=NORMAL**

### **,EWLMMODE=EXPLICIT\_SINGLE**

An optional input keyword that indicates if the parameter EXPLICIT\_SINGLE has been specified on the IWM4ECRE (or IWMECREA) call. Note that EWLMMODE=EXPLICIT\_SINGLE can only be specified if the enclave was created with the ESTRT=EXPLICIT\_SINGLE option. The values of EWLMMODE are:

- **EWLMMODE=NORMAL** indicates that the enclave was created with the ESTRT=EXPLICIT or with the ESTRT=IMPLIED option. EWLMMODE=NORMAL is the default.
- **EWLMMODE=EXPLICIT\_SINGLE** indicates that the enclave was created with the ESTRT=EXPLICIT\_SINGLE option which can only be used for a restricted environment. The caller must provide a standard 72-byte savearea, which is addressable in AMODE 31 and pointed to by GPR13. The parameter list and the classification parameters must also be addressable in AMODE 31.

### **,MF=S**

### **,MF=(L-,list addr)**

### **,MF=(L-,list addr,attr)**

**,MF=(L,list addr,0D)**

**,MF=(E,list addr)**

**,MF=(E,list addr,COMPLETE)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

**,list addr**

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

**,attr**

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

**,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

**,PLISTVER=IMPLIED\_VERSION**

**,PLISTVER=MAX**

**,PLISTVER=0**

**,PLISTVER=1**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form, when both are assembled with the same level of the system. In this way, MAX ensures that the parameter list does not overwrite nearby storage.

- **0**, if you use the currently available parameters.
- **1**, which supports both the following parameters and those from version 0:

ETOKEN  
EWLMODE

**To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX
- A decimal value of 0, or 1

**,RETCODE=retcode**

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,RSNCODE=rsncode**

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**WORKREQ\_HDL=workreq\_hdl**

A required input parameter that contains the handle which represents the work request. This handle was returned from a previous call to IWMESTRT or IWM4ECRE.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

## ABEND Codes

None.

## Return and Reason Codes

When the IWMEUBLK macro returns control to your program:

- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, if you coded RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 40. Return and Reason Codes for the IWMEUBLK Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> IwmRetCodeOk <b>Meaning:</b> Successful completion. <b>Action:</b> None required.
8	—	<b>Equate Symbol:</b> IwmRetCodeInvocError <b>Meaning:</b> Invalid invocation environment or parameters.
8	xxxx0803	<b>Equate Symbol:</b> IwmRsnCodeDisabled <b>Meaning:</b> Caller is disabled. <b>Action:</b> Avoid requesting this function while disabled.

Table 40. Return and Reason Codes for the IWMEUBLK Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0804	<p><b>Equate Symbol:</b> IwmRsnCodeLocked</p> <p><b>Meaning:</b> Caller is locked.</p> <p><b>Action:</b> Avoid requesting this function while locked.</p>
8	xxxx080B	<p><b>Equate Symbol:</b> IwmRsnCodeBadPI</p> <p><b>Meaning:</b> Error accessing parameter list.</p> <p><b>Action:</b> Check for possible storage overlay.</p>
8	xxxx0824	<p><b>Equate Symbol:</b> IwmRsnCodeAmode24</p> <p><b>Meaning:</b> Caller invoked service but was in 24 bit addressing mode.</p> <p><b>Action:</b> Request this function only when you are in 31 bit addressing mode.</p>
8	xxxx0828	<p><b>Equate Symbol:</b> IwmRsnCodeBadVersion</p> <p><b>Meaning:</b> Version number in parameter list is not valid.</p> <p><b>Action:</b> Check for possible storage overlay of the parameter list.</p>
8	xxxx083A	<p><b>Equate Symbol:</b> IwmRsnCodeBadEnclave</p> <p><b>Meaning:</b> EWLMODE=EXPLICIT_SINGLE was specified and the required enclave token does not pass verification.</p> <p><b>Action:</b> Check for possible storage overlay of the enclave token, or for asynchronous events which may have deleted the enclave.</p>
8	xxxx0895	<p><b>Equate Symbol:</b> IwmRsnCodeEWLMservNotEnabled</p> <p><b>Meaning:</b> Service is not enabled because caller invoked the IWM4CON (IWMCONN) service with EWLM=NO.</p> <p><b>Action:</b> Ensure that EWLM=YES is specified on the IWM4CON request to enable this service.</p>
8	xxxx0896	<p><b>Equate Symbol:</b> IwmRsnCodeBadWorkReqHandle</p> <p><b>Meaning:</b> Work request handle is invalid.</p> <p><b>Action:</b> Check the specification of the WORKREQ_HDL parameter.</p>
8	xxxx0898	<p><b>Equate Symbol:</b> IwmRsnCodeBadBlockHandle</p> <p><b>Meaning:</b> Block handle is invalid.</p> <p><b>Action:</b> Check the specification of the BLOCK_HDL parameter.</p>

Table 40. Return and Reason Codes for the IWMEUBLK Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx08A0	<p><b>Equate Symbol:</b> IwmRsnCodeNotExplicitSingle</p> <p><b>Meaning:</b> The service has been invoked with option EWLMMODE=EXPLICIT_SINGLE but the enclave has not been created with the ESTRT=EXPLICIT_SINGLE option or vice versa.</p> <p><b>Action:</b> If ESTRT=EXPLICIT_SINGLE was coded on the IWM4ECRE or IWMECREA call, EWLMMODE=EXPLICIT_SINGLE must also specified on the IWMEBLK call. If ESTRT=EXPLICIT OR ESTRT=IMPLIED was coded on the IWM4ECRE or IWMECREA call, EWLMMODE=NORMAL must be specified on the IWMEBLK call (or the EWLMMODE parameter can be omitted).</p>
10	—	<p><b>Equate Symbol:</b> IwmRetCodeCompError</p> <p><b>Meaning:</b> Component error.</p> <p><b>Action:</b> No action required. The function may be successful if invoked again.</p>

## Examples

```

                IWMEUBLK WORKREQ_HDL=WRHANDLE,
                BLOCK_HDL=BKHANDLE
*
* Storage areas
*
WRHANDLE DS    CL8           Work Request Handle
BKHANDLE DS    CL8           Work Request Block Handle

```



---

## Chapter 31. IWMEPT – WLM Export Service

The IWMEPT macro exports an enclave to all systems in a parallel sysplex. This enables dispatchable units on other systems to import and join the enclave. The macro returns an export token which the caller must pass to other systems where it wants to import and join the enclave.

The caller must invoke the IWMUEPT macro when other systems no longer need access to the exported enclave.

The primary address space must have connected to WLM using the IWM4CON macro.

---

### Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Supervisor state or program key mask (PKM) allowing keys 0-7.
<b>Dispatchable unit mode:</b>	Task or SRB
<b>Cross memory mode:</b>	Any HASN, any SASN. PASN must be the address space which connected to WLM.
<b>AMODE:</b>	31-bit
<b>ASC mode:</b>	Primary
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks may be held.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

---

### Programming Requirements

1. The macro CVT must be included to use this macro.
2. The macro IWMYCON must be included to use this macro.
3. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
4. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, IWMRSNCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.

---

### Restrictions

None.

---

### Input Register Information

Before issuing the IWMEPT macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

## Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Reason code if GR15 return code is non-zero
1	Used as work registers by the system
2-13	Unchanged
14	Used as work registers by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

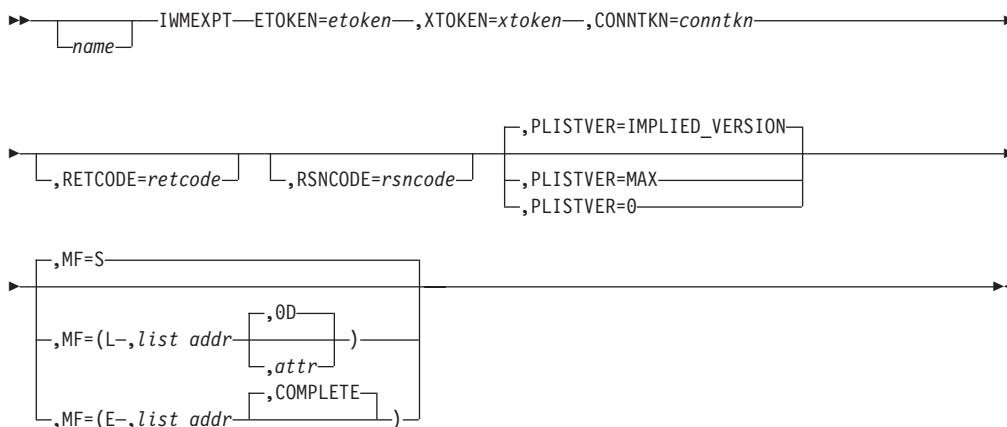
Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

## Performance Implications

None.

## Syntax

### main diagram



## Parameters

The parameters are explained as follows:

### *name*

An optional symbol, starting in column 1, that is the name on the IWMEXPT macro invocation. The name must conform to the rules for an ordinary assembler language symbol.



**,CONNTKN=conntkn**

A required input parameter that contains the connect token for the primary address space's connection to WLM.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32 bit field.

**ETOKEN=etoken**

A required input parameter that contains the enclave token of the enclave to be exported.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

**,MF=S****,MF=(L,list addr)****,MF=(L,list addr,attr)****,MF=(L,list addr,0D)****,MF=(E,list addr)****,MF=(E,list addr,COMPLETE)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

**,list addr**

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

**,attr**

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

**,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

**,PLISTVER=IMPLIED\_VERSION****,PLISTVER=MAX****,PLISTVER=0**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

## IWMEXPT Macro

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the `PLISTVER` parameter, `IMPLIED_VERSION` is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify `PLISTVER=MAX` on the list form of the macro. Specifying `MAX` ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form, when both are assembled with the same level of the system. In this way, `MAX` ensures that the parameter list does not overwrite nearby storage.

- **0**, if you use the currently available parameters.

**To code:** Specify one of the following:

- `IMPLIED_VERSION`
- `MAX`
- A decimal value of 0

**,RETCODE=*retcode***

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,RSNCODE=*rsncode***

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,XTOKEN=*xtoken***

A required output parameter that contains an export token which uniquely identifies the exported enclave throughout the parallel sysplex.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32-character field.

---

## ABEND Codes

None.

---

## Return and Reason Codes

When the `IWMEXPT` macro returns control to your program:

- GPR 15 (and *retcode*, when you code `RETCODE`) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, when you code `RSNCODE`) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 41. Return and Reason Codes for the IWMEXPT Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<p><b>Equate Symbol:</b> IwmRetCodeOk</p> <p><b>Meaning:</b> Successful completion.</p> <p><b>Action:</b> None required.</p>
4	—	<p><b>Equate Symbol:</b> IwmRetCodeWarning</p> <p><b>Meaning:</b> Successful completion, unusual conditions noted.</p>
4	xxxx0433	<p><b>Equate Symbol:</b> IwmRsnCodeEncAlreadyExported</p> <p><b>Meaning:</b> The input enclave was imported from another system. It cannot be exported by this system.</p> <p><b>Action:</b> The existing export token associated with the input enclave is returned. It can be used as long as it remains valid, which is under the control of the work manager that exported the enclave. Do not invoke IWMUEXPT to undo the export since this invocation did not establish the export.</p>
8	—	<p><b>Equate Symbol:</b> IwmRetCodeInvocError</p> <p><b>Meaning:</b> Invalid invocation environment or parameters.</p>
8	xxxx0803	<p><b>Equate Symbol:</b> IwmRsnCodeDisabled</p> <p><b>Meaning:</b> Caller is disabled.</p> <p><b>Action:</b> Avoid requesting this function while disabled.</p>
8	xxxx0804	<p><b>Equate Symbol:</b> IwmRsnCodeLocked</p> <p><b>Meaning:</b> Caller is locked.</p> <p><b>Action:</b> Avoid requesting this function while locked.</p>
8	xxxx080B	<p><b>Equate Symbol:</b> IwmRsnCodeBadPI</p> <p><b>Meaning:</b> Error accessing parameter list.</p> <p><b>Action:</b> Check for possible storage overlay.</p>
8	xxxx0821	<p><b>Equate Symbol:</b> IwmRsnCodeBadConn</p> <p><b>Meaning:</b> Input connect token does not pass validity checking.</p> <p><b>Action:</b> Make sure the primary address space connected to WLM using the IWM4CON service. Make sure the connect token returned by IWM4CON is passed correctly.</p>
8	xxxx0824	<p><b>Equate Symbol:</b> IwmRsnCodeAmode24</p> <p><b>Meaning:</b> Caller invoked service but was in 24 bit addressing mode.</p> <p><b>Action:</b> Request this function only when you are in 31 bit addressing mode.</p>
8	xxxx0825	<p><b>Equate Symbol:</b> IwmRsnCodeAscModeNotPrimary</p> <p><b>Meaning:</b> Caller invoked service but was not in primary ASC mode.</p> <p><b>Action:</b> Avoid requesting this function in this environment.</p>

Table 41. Return and Reason Codes for the IWMEXPT Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0827	<b>Equate Symbol:</b> IwmRsnCodeRsvdNot0 <b>Meaning:</b> Reserved field in parameter list was non-zero. <b>Action:</b> Check for use of keywords that are not supported by the z/OS release on which the program is running.
8	xxxx0828	<b>Equate Symbol:</b> IwmRsnCodeBadVersion <b>Meaning:</b> Version number in parameter list is not valid. <b>Action:</b> Check for possible storage overlay of the parameter list.
8	xxxx083A	<b>Equate Symbol:</b> IwmRsnCodeBadEnclave <b>Meaning:</b> Enclave token does not pass verification. <b>Action:</b> Check for possible storage overlay of the enclave token, or asynchronous events which may have deleted the enclave.
8	xxxx0840	<b>Equate Symbol:</b> IwmRsnCodeServiceNotEnabled <b>Meaning:</b> Caller's connection is not enabled for this service. <b>Action:</b> Make sure that EXPTIMPT=YES is specified on the IWM4CON macro invocation.
8	xxxx0842	<b>Equate Symbol:</b> IwmRsnCodeNoWLMConnect <b>Meaning:</b> Caller's primary address space is not connected to WLM. <b>Action:</b> Invoke the IWM4CON macro before invoking this macro.
8	xxxx0884	<b>Equate Symbol:</b> IwmRsnCodeEnclaveDefEx <b>Meaning:</b> Enclave is marked Execution Start Time to be deferred. <b>Action:</b> Such an Enclave can not be exported.
C	—	<b>Equate Symbol:</b> IwmRetCodeEnvError <b>Meaning:</b> Environmental error.
C	xxxx0C01	<b>Equate Symbol:</b> IwmRsnCodeNoStg <b>Meaning:</b> Virtual storage is not available for the request. <b>Action:</b> Process the unit-of-work on the local system or fail the unit-of-work, whichever is appropriate.
C	xxxx0C2F	<b>Equate Symbol:</b> IwmRsnCodeSystemSpace <b>Meaning:</b> The enclave is owned by a system (i.e. limited-function) address space. Exporting such an enclave is not supported. <b>Action:</b> Process the unit-of-work on the local system or fail the unit-of-work, whichever is appropriate.
C	xxxx0C37	<b>Equate Symbol:</b> IwmRsnCodeStructureFull <b>Meaning:</b> The coupling facility structure is full. <b>Action:</b> Process the unit-of-work on the local system or fail the unit-of-work, whichever is appropriate.

Table 41. Return and Reason Codes for the IWMEXPT Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
C	xxxx0C36	<p><b>Equate Symbol:</b> lwmRsnCodeStructureUnavailable</p> <p><b>Meaning:</b> WLM does not have access to its coupling facility structure.</p> <p><b>Action:</b> Process the unit-of-work on the local system or fail the unit-of-work, whichever is appropriate. Check for WLM or XES messages which describe the problem.</p>
10	—	<p><b>Equate Symbol:</b> lwmRetCodeCompError</p> <p><b>Meaning:</b> Component error.</p> <p><b>Action:</b> Contact your system programmer.</p>

---

## Example

```

                IWMEXPT ETOKEN=ENCLAVET,XTOKEN=EXPORTT,CONNTKN=CONNECTT
*
* Storage areas
*
ENCLAVET DS    CL8           Enclave token
EXPORTT  DS    CL32         Export token
CONNECTT DS    CL4           Connect token

```



---

## Chapter 32. IWMIMPT – WLM Import Service

The IWMIMPT macro imports an enclave that has been previously exported using the IWMEPXT macro.

The caller must invoke the IWMUIMPT macro when it no longer needs access to the enclave.

The primary address space must have connected to WLM using the IWM4CON macro.

---

### Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Supervisor state or program key mask (PKM) allowing keys 0-7.
<b>Dispatchable unit mode:</b>	Task or SRB
<b>Cross memory mode:</b>	Any HASN, any SASN. PASN must be the address space which connected to WLM.
<b>AMODE:</b>	31-bit
<b>ASC mode:</b>	Primary
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks may be held.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

---

### Programming Requirements

1. The macro CVT must be included to use this macro.
2. The macro IWMYCON must be included to use this macro.
3. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
4. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, IWMSRNCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.

---

### Restrictions

None.

---

### Input Register Information

Before issuing the IWMIMPT macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

---

### Output Register Information

When control returns to the caller, the GPRs contain:

## IWMIMPT Macro

Register	Contents
0	Reason code if GR15 return code is non-zero
1	Used as work registers by the system
2-13	Unchanged
14	Used as work registers by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

---

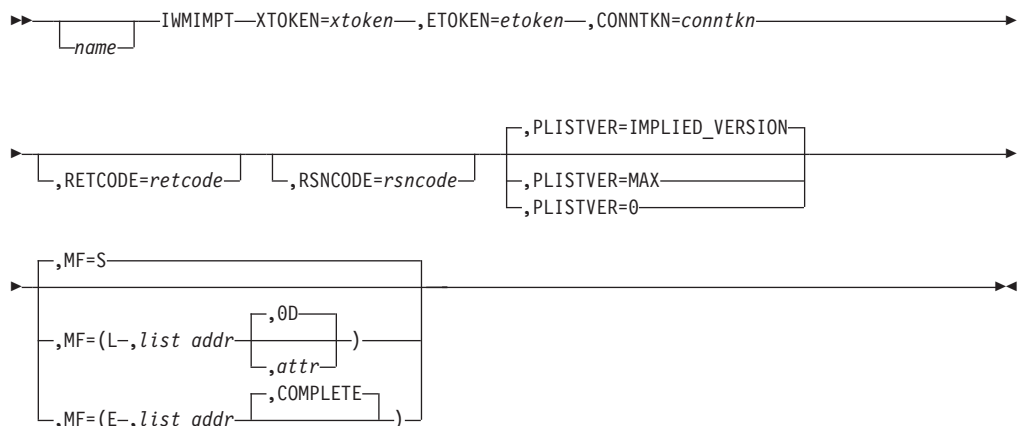
## Performance Implications

None.

---

## Syntax

### main diagram



---

## Parameters

The parameters are explained as follows:

### *name*

An optional symbol, starting in column 1, that is the name on the IWMIMPT macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

### **,CONNTKN=conntkn**

A required input parameter that contains the connect token for the primary address space's connection to WLM.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32 bit field.



**,ETOKEN=*etoken***

A required output parameter that contains the enclave token for the imported enclave. The caller can pass this token as input to all enclave services except IWM4EDEL. The enclave token is valid on the local system only.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

**,MF=S****,MF=(L,*list addr*)****,MF=(L,*list addr,attr*)****,MF=(L,*list addr,OD*)****,MF=(E,*list addr*)****,MF=(E,*list addr,COMPLETE*)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

**,*list addr***

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

**,*attr***

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

**,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

**,PLISTVER=IMPLIED\_VERSION****,PLISTVER=MAX****,PLISTVER=0**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

## IWMIMPT Macro

If you can tolerate the size change, IBM recommends that you always specify `PLISTVER=MAX` on the list form of the macro. Specifying `MAX` ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form, when both are assembled with the same level of the system. In this way, `MAX` ensures that the parameter list does not overwrite nearby storage.

- `0`, if you use the currently available parameters.

**To code:** Specify one of the following:

- `IMPLIED_VERSION`
- `MAX`
- A decimal value of `0`

**,RETCODE=***retcode*

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,RSNCODE=***rsncode*

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**XTOKEN=***xtoken*

A required input parameter that contains the export token which identifies the exported enclave.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32-character field.

---

## ABEND Codes

None.

## Return and Reason Codes

When the IWMIMPT macro returns control to your program:

- GPR 15 (and *retcode*, when you code RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, when you code RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 42. Return and Reason Codes for the IWMIMPT Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> <code>IwmRetCodeOk</code> <b>Meaning:</b> Successful completion. <b>Action:</b> None required.
4	—	<b>Equate Symbol:</b> <code>IwmRetCodeWarning</code> <b>Meaning:</b> Successful completion, unusual conditions noted.
4	xxxx0432	<b>Equate Symbol:</b> <code>IwmRsnCodeUnknownExportToken</code> <b>Meaning:</b> No enclave matching the export token was found. The enclave may have been unexported or deleted, or the WLM coupling facility structure may have been lost. <b>Action:</b> Discontinue processing the unit of work.
8	—	<b>Equate Symbol:</b> <code>IwmRetCodeInvocError</code> <b>Meaning:</b> Invalid invocation environment or parameters.
8	xxxx0803	<b>Equate Symbol:</b> <code>IwmRsnCodeDisabled</code> <b>Meaning:</b> Caller is disabled. <b>Action:</b> Avoid requesting this function while disabled.
8	xxxx0804	<b>Equate Symbol:</b> <code>IwmRsnCodeLocked</code> <b>Meaning:</b> Caller is locked. <b>Action:</b> Avoid requesting this function while locked.
8	xxxx080B	<b>Equate Symbol:</b> <code>IwmRsnCodeBadPI</code> <b>Meaning:</b> Error accessing parameter list. <b>Action:</b> Check for possible storage overlay.
8	xxxx0821	<b>Equate Symbol:</b> <code>IwmRsnCodeBadConn</code> <b>Meaning:</b> Input connect token does not pass validity checking. <b>Action:</b> Make sure the primary address space connected to WLM using the IWM4CON service. Make sure the connect token returned by IWM4CON is passed correctly.
8	xxxx0824	<b>Equate Symbol:</b> <code>IwmRsnCodeAmode24</code> <b>Meaning:</b> Caller invoked service but was in 24 bit addressing mode. <b>Action:</b> Request this function only when you are in 31 bit addressing mode.

Table 42. Return and Reason Codes for the IWMIMPT Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0825	<b>Equate Symbol:</b> IwmRsnCodeAscModeNotPrimary <b>Meaning:</b> Caller invoked service but was not in primary ASC mode. <b>Action:</b> Avoid requesting this function in this environment.
8	xxxx0827	<b>Equate Symbol:</b> IwmRsnCodeRsvdNot0 <b>Meaning:</b> Reserved field in parameter list was non-zero. <b>Action:</b> Check for use of keywords that are not supported by the OS/390 release on which the program is running.
8	xxxx0828	<b>Equate Symbol:</b> IwmRsnCodeBadVersion <b>Meaning:</b> Version number in parameter list is not valid. <b>Action:</b> Check for possible storage overlay of the parameter list.
8	xxxx0836	<b>Equate Symbol:</b> IwmRsnCodeMaxEnclave <b>Meaning:</b> Enclave could not be created because the enclave limit has been reached. <b>Action:</b> Check for possible problems wherein enclaves are not being deleted as expected or excessive numbers of enclaves are being created in a loop.
8	xxxx0840	<b>Equate Symbol:</b> IwmRsnCodeServiceNotEnabled <b>Meaning:</b> Caller's connection is not enabled for this service. <b>Action:</b> Make sure that EXPTIMPT=YES is specified on the IWM4CON macro invocation.
8	xxxx0842	<b>Equate Symbol:</b> IwmRsnCodeNoWLMConnect <b>Meaning:</b> Caller's primary address space is not connected to WLM. <b>Action:</b> Invoke the IWM4CON macro before invoking this macro.
8	xxxx0870	<b>Equate Symbol:</b> IwmRsnCodeBadExportToken <b>Meaning:</b> The export token is not validly formatted. <b>Action:</b> Check for possible storage overlay of the export token.
C	—	<b>Equate Symbol:</b> IwmRetCodeEnvError <b>Meaning:</b> Environmental error.
C	xxxx0C01	<b>Equate Symbol:</b> IwmRsnCodeNoStg <b>Meaning:</b> Virtual storage is not available for the request. <b>Action:</b> Contact your system programmer.
C	xxxx0C36	<b>Equate Symbol:</b> IwmRsnCodeStructureUnavailable <b>Meaning:</b> WLM does not have access to its coupling facility structure. <b>Action:</b> Check for WLM or XES messages which describe the problem.

Table 42. Return and Reason Codes for the IWMIMPT Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
C	xxxx0C38	<p><b>Equate Symbol:</b> IwmRsnCodeUplevelObject</p> <p><b>Meaning:</b> The multisystem enclave requires functions that are not available on this level of the operating system.</p> <p><b>Action:</b> Do not process the work request on this system.</p>
C	xxxx0C39	<p><b>Equate Symbol:</b> IwmRsnCodeTooManySystems</p> <p><b>Meaning:</b> The sysplex has exceeded 32 systems with unique names. This can occur when a system is relPLeD into the sysplex with a different SYSNAME or CPU Adjustment factor.</p> <p><b>Action:</b> Do not process the work request on this system.</p>
10	—	<p><b>Equate Symbol:</b> IwmRetCodeCompError</p> <p><b>Meaning:</b> Component error.</p> <p><b>Action:</b> Contact your system programmer.</p>

---

## Example

```

IWMIMPT XTOKEN=EXPORTT,ETOKEN=ENCLAVET,CONNTKN=CONNECTT
*
* Storage areas
*
ENCLAVET DS    CL8           Enclave token
EXPORTT  DS    CL32          Export token
CONNECTT DS    CL4           Connect token

```

## IWMIMPT Macro

---

## Chapter 33. IWMMABNL – Record Abnormal Event

You can use IWMMABNL to record an abnormal event that has occurred for work requests. The information is kept in the input monitoring environment. The abnormal condition supplements any existing abnormal conditions recorded in the input monitoring environment.

The abnormal events are transferred (with all other information) to any parent monitoring environment with the IWMMXFER FUNCTION=RETURN parameter. These abnormal events are recorded in the monitoring environment so that the caller can get an indication that further requests might fail due to some abnormal situation. This supplements any information a caller receives from the return code.

---

### Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Supervisor state, or problem state. PSW key must either be 0 or match the value supplied on IWM4MCRE for the input monitoring token when MONENVKEYP=PSWKEY is specified.
<b>Dispatchable unit mode:</b>	Task or SRB
<b>Cross memory mode:</b>	Any PASN, any HASN, any SASN
<b>AMODE:</b>	31-bit
<b>ASC mode:</b>	Primary mode
<b>Interrupt status:</b>	Enabled
<b>Locks:</b>	Unlocked
<b>Control parameters:</b>	Control parameters must be in the primary address space.

---

### Programming Requirements

- You must include the IWMYCON mapping macro in the calling program.
- If the key specified on IWM4MCRE was a user key, then either primary or secondary addressability must exist to the performance block IWM4MCRE obtained.
- If you specify MONENVKEYP=VALUE, then the caller must be in supervisor state or have PKM authority to the key specified by MONENVKEY.
- If you specify MONENVKEYP=UNKNOWN, then the caller must be in supervisor state or have PKM authority to key 0.
- The caller must serialize to prevent any delay monitoring services from being invoked concurrently for the environment represented by the monitoring token.
- Only limited validity checking is done on the input monitoring token.
- The caller is responsible for error recovery.

---

### Restrictions

None.

---

## Input Register Information

Before issuing the IWMMABNL macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

---

## Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Reason code if the return code in GPR 15 is not 0, otherwise, used as a work register by the system.
1	Used as a work register by the system.
2 - 13	Unchanged
14	Used as a work register by the system.
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0 - 1	Used as a work register by the system.
2 - 13	Unchanged
14 - 15	Used as a work register by the system.

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

---

## Performance Implications

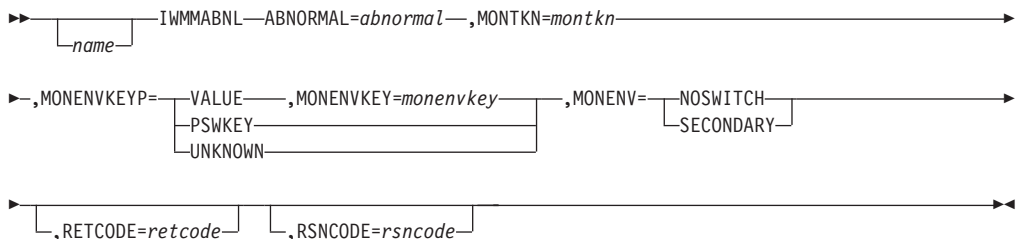
None.

---

## Syntax

The syntax of the IWMMABNL macro is as follows:

### main diagram



---

## Parameters

The parameters are explained as follows:



**ABNORMAL=abnormal**

Required input parameter specifying the abnormal mask reflecting the abnormality. Macro IWMYCON contains the defined abnormal masks. The mask variable names begin with IWMABNL, for example - IWMABNL\_SCOPE\_LOCALMVS.

**To code:** Specify the name (RS-type), or address in register (2)-(12) of a 32-bit field containing the abnormal mask.

**,MONTKN=montkn**

Required input parameter specifying the monitoring token for the environment affected by the abnormality.

**To code:** Specify the name (RS-type), or address in register (2)-(12) of a 32 bit field containing the monitoring token.

**,MONENVKEYP=VALUE****,MONENVKEYP=PSWKEY****,MONENVKEYP=UNKNOWN**

Required input parameter that specifies whether a key switch is needed to access the input monitoring environment.

Use MONENVKEYP=VALUE to indicate that the key is being passed explicitly in MONENVKEY.

Use MONENVKEYP=PSWKEY to indicate that the current PSW key should be used. If you use this parameter, the input monitoring environment must have been created with the same key as the current PSW key.

Use MONENVKEYP=UNKNOWN to indicate that the key associated with the input monitoring environment is unknown. If you use this parameter, the caller must be in supervisor state or have PKM authority to key 0.

**,MONENVKEY=monenvkey**

Required input parameter for MONENVKEYP=VALUE that specifies the key in which the input monitoring environment must be accessed. If you use this parameter, the caller must be in supervisor state or have PKM authority to the key specified.

**To code:** Specify an 8 bit name (RS-type), or address in register (2)-(12), of the monitoring environment key. The leftmost (high order) 4 bits contain the key value. Note that this uses the machine orientation for keeping the storage key in the high order bits.

**,MONENV=NOSWITCH****,MONENV=SECONDARY**

Required keyword input which describes whether an address space switch is needed to access the input monitoring environment.

Use MONENV=NOSWITCH to indicate that no space switch is needed to access the input monitoring environment. This is appropriate if the input monitoring environment was established (by IWM4MCRE) to be used by routines in a specific system key or if it was established to be used in a specific user key in the current primary.

Use MONENV=SECONDARY to indicate that the input monitoring environment was established in current secondary (for use by a specific user key).

**,RETCODE=retcode addr**

Optional output parameter that specifies where the system is to store the return code. The return code is also in GPR 15.

## IWMMABNL Macro

**To code:** Specify the name (RS-type) or address (using a register from 2 to 12) of a fullword to contain the return code.

**,RSNCODE=rsncode addr**

Optional output parameter that specifies where the system is to store the reason code. The reason code is also in GPR 0.

**To code:** Specify the name (RS-type) or address (using a register from 2 to 12) of a fullword to contain the reason code (if any).

---

## ABEND Codes

None.

---

## Return and Reason Codes

When IWMMABNL macro returns control to your program, GPR 15 contains a return code. When the return code is non-zero, then GPR 0 contains a reason code.

Hexadecimal Return Code	Hexadecimal Reason Code	Meaning
00		<b>Meaning:</b> Successful completion. <b>Action:</b> None.
04	0402	<b>Meaning:</b> Warning. Input monitoring token indicates no monitoring environment was established. <b>Action:</b> None.
08	0820	<b>Meaning:</b> Program error. The monitoring environment does not pass short form verification.

---

## Example

To record an abnormal event that has occurred for a work request associated with a monitoring token defined in the field MONTKN1, specify:

```
IWMMABNL ABNORMAL=(R7),  
          MONTKN=MONTKN1,  
          MONENVKEYP=PSWKEY,  
          MONENV=NOSWITCH,  
          RETCODE=RCODE,RSNCODE=RSN
```

No address space switch is required, so you can specify MONENV=NOSWITCH.

---

## Chapter 34. IWMMDELE – Delete Monitoring Environment

Use this macro to delete the monitoring environment. You should invoke IWMMDELE during address space shutdown processing.

---

### Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Supervisor state, or PKM key 0 - 7.
<b>Dispatchable unit mode:</b>	Task or SRB
<b>Cross memory mode:</b>	Any PASN, any SASN. HASN must match the HASN when IWM4MCRE was invoked.
<b>AMODE:</b>	31-bit
<b>ASC mode:</b>	Primary
<b>Interrupt status:</b>	Enabled
<b>Locks:</b>	Unlocked, but FRRs may be established.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

---

### Programming Requirements

1. You must include the CVT and IWMYCON mapping macros in the calling program.
2. The caller must serialize to prevent any delay monitoring services from being invoked concurrently or subsequently for the environment represented by the monitoring token
3. Do not invoke IWMMDELE while in a RTM termination routine (resource manager) for the task owning the monitoring environment since MVS will have its own resource cleanup routine and unpredictable results would occur. It is legitimate to use this service while in a recovery routine, however, or in mainline processing.

---

### Restrictions

1. If the key specified on IWM4MCRE was a user key (8-F), then all of the following must be true:
  - Caller must be in non-cross-memory mode (PASN=SASN=HASN). This implies that the current primary must match the primary at the time that IWM4MCRE was invoked.
  - Must be in task mode (not SRB)
  - Current task must match the task at the time that IWM4MCRE was invoked.

---

### Input Register Information

Before issuing the IWMMDELE macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

---

### Output Register Information

When control returns to the caller, the GPRs contain:

## IWMMDELE Macro

Register	Contents
0	Reason code if the return code in GPR 15 is not 0, otherwise, used as a work register by the system.
1	Used as a work register by the system.
2 - 13	Unchanged
14	Used as a work register by the system.
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0 - 1	Used as a work register by the system.
2 - 13	Unchanged
14 - 15	Used as a work register by the system.

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

---

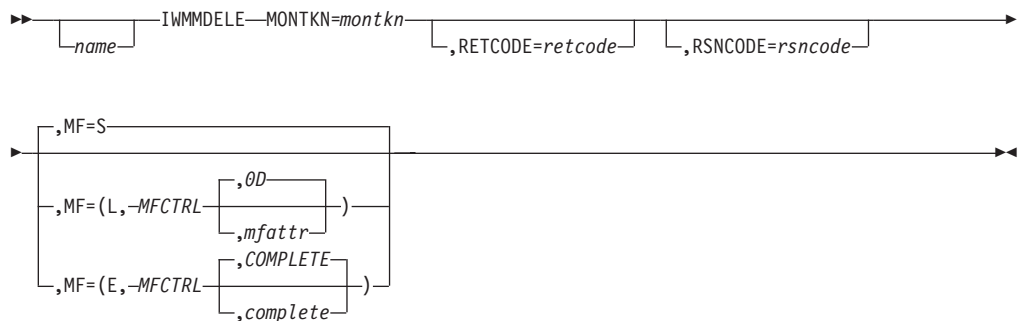
## Performance Implications

None.

---

## Syntax

### main diagram




---

## Parameters

The parameters are explained as follows:

### **,MONTKN=montkn**

Required input parameter that specifies the monitoring token obtained from IWM4MCRE.

**To code:** Specify the name (RS-type) or address (using a register from 2 to 12) of a 32-bit field containing the monitoring token.

### **,RETCODE=retcode addr**

Optional output parameter that specifies where the system is to store the return code. The return code is also in GPR 15.

**To code:** Specify the name (RS-type) or address (using a register from 2 to 12) of a fullword to contain the return code.

**,RSNCODE=rsncode addr**

Optional output parameter that specifies where the system is to store the reason code. The reason code is also in GPR 0.

**To code:** Specify the name (RS-type) or address (using a register from 2 to 12) of a fullword to contain the reason code (if any).

**,MF=S**

**,MF=(L,mfctrl,mfattr)**

**,MF=(E,mfctrl,COMPLETE)**

Use MF=S to specify the standard form, which places parameters into an inline parameter list and invokes the IWM4CON macro service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require re-entrant code. The list form defines an area of storage that the execute form uses to store the parameters.

Use MF=E to specify the execute form of the macro. Use the execute form with the list form of the macro for applications that require re-entrant code. The execute form stores the parameters into the storage area defined by the list form and generates the macro invocation to transfer control to the service.

**,mfctrl**

Use this output parameter to specify the name of the storage area to contain the parameters.

**To code:** Specify the name (RS-type) or address (using a register from 2 to 12) of the storage area containing the parameter list.

**,mfattr**

Use this input parameter to specify the name of a 1 to 60 character storage area that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *,mfattr* the system provides a value of 0D, which forces the parameter on a doubleword boundary.

**,COMPLETE**

Use this input parameter to specify that the system check for required parameters and supply defaults for omitted optional parameter.

---

## ABEND Codes

None.

---

## Return and Reason Codes

When IWMMDLE macro returns control to your program, GPR 15 contains a return code. When the return code is non-zero, then GPR 0 contains a reason code.

Hexadecimal Return Code	Hexadecimal Reason Code	Meaning
00		<b>Meaning:</b> Successful completion.
04	0401	<b>Meaning:</b> Warning. The system does not support delay monitoring services. This return code is only set when the MVS release is prior to MVS/ESA SP 5.2.
04	0402	<b>Meaning:</b> Warning. Input monitoring token indicates no monitoring environment was established.

## IWMMDELE Macro

Hexadecimal Return Code	Hexadecimal Reason Code	Meaning
04	0403	<b>Meaning:</b> Warning. Input monitoring token does not reflect an allocated monitoring environment owned by the current home address space.
08	0802	<b>Meaning:</b> Program error. Caller is in cross-memory mode while the token was obtained in user key.
08	0803	<b>Meaning:</b> Program error. Caller is disabled.
08	0804	<b>Meaning:</b> Program error. Caller is locked.
08	0805	<b>Meaning:</b> Program error. Input monitoring token reflects a switch continuation.
08	0806	<b>Meaning:</b> Program error. Input monitoring token reflects a continuation to a dependent monitoring environment.
08	0808	<b>Meaning:</b> Program error. Input monitoring token reflects a continuation from a parent monitoring environment.
08	0809	<b>Meaning:</b> Program error. Caller is in SRB mode, while the token was obtained in a user key (8-F).
08	080A	<b>Meaning:</b> Program error. Current task is not the current owner, while the token was obtained in a user key (8-F).
08	080B	<b>Meaning:</b> Program error. Error accessing parameter list.
08	0823	<b>Meaning:</b> Program error. Caller invoked service while dynamic address translation was disabled.
08	0824	<b>Meaning:</b> Program error. Caller invoked service but was in 24 bit addressing mode.
08	0825	<b>Meaning:</b> Program error. Caller invoked service but was not in Primary ASC mode.
08	0826	<b>Meaning:</b> Program error. Caller invoked service while task termination is in progress for the task associated with the owner.
08	0827	<b>Meaning:</b> Program error. Reserved field in parameter list was non-zero.
08	0828	<b>Meaning:</b> Program error. Version number in parameter list is not valid.
08	082A	<b>Meaning:</b> Program error. Input monitor token is related to a parent monitoring environment.

---

### Example

To delete a monitoring environment, where the monitoring token is in register 7, specify:

```
IWMMDELE MONTKN=(R7),  
          RETCODE=RCODE,RSNCODE=RSN
```

---

## Chapter 35. IWMEXTR – Monitor Environment Extract Service

The purpose of this service is to extract information about the monitoring environment which was previously passed through IWM4MINI/IWMMRELA. When IWMMRELA was invoked for a management monitoring environment, owner token, owner data and abnormal conditions are always available. Arrival time, userid, and transaction name are only available when IWM4MINI was previously invoked. Arrival time, however, is only available for management monitoring environments.

When the service class token is requested for a management monitoring environment, the value may represent a token from a prior active policy. Furthermore, when the monitoring environment was established via IWMMRELA, the token may be zero, which does not represent a valid service class or report class. IWMWQRY may be used to obtain the service and/or report class name, along with other information about these classes. The SERVCLS keyword is not applicable for report-only monitoring environments. The returned token is zero, which does not represent a valid service class.

The ENCLAVE\_TOKEN and ASID keywords are only applicable for report-only monitoring environments.

When no output keywords are specified, the service merely checks whether a monitoring environment was established and passes short form checking.

---

### Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Either problem state or supervisor state. Any PSW key.
<b>Dispatchable unit mode:</b>	Task or SRB
<b>Cross memory mode:</b>	Any PASN, any HASN, any SASN
<b>AMODE:</b>	31-bit
<b>ASC mode:</b>	Primary
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	Suspend locks are allowed.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

---

### Programming Requirements

1. The macro IWMYCON must be included to use this macro.
2. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
3. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, IWMSRNCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.

---

## Restrictions

1. Caller is responsible for error recovery
2. Only limited checking is done against the input monitoring token.
3. If the key specified on IWM4MCRE was a user key (8-F), then the primary addressability must exist to the performance block IWM4MCRE obtained. This condition is satisfied by ensuring that current primary matches primary at the time that IWM4MCRE was invoked. If this service is invoked in a subspace, the condition may be satisfied by ensuring that the performance block is shared with the base space.
4. The caller must serialize to prevent any delay monitoring services from being invoked concurrently for the environment represented by the monitoring token.
5. This macro may only be used on z/OS R2 or higher levels for ENCLAVE\_TOKEN and ASID keywords.
6. This macro supports multiple versions. Some keywords are only supported by certain versions. Refer to the PLISTVER parameter description for further information.

---

## Input Register Information

Before issuing the IWMSEXTR macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

---

## Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Reason code if GR15 return code is non-zero
1	Used as work registers by the system
2-13	Unchanged
14	Used as work registers by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

---

## Performance Implications

None.

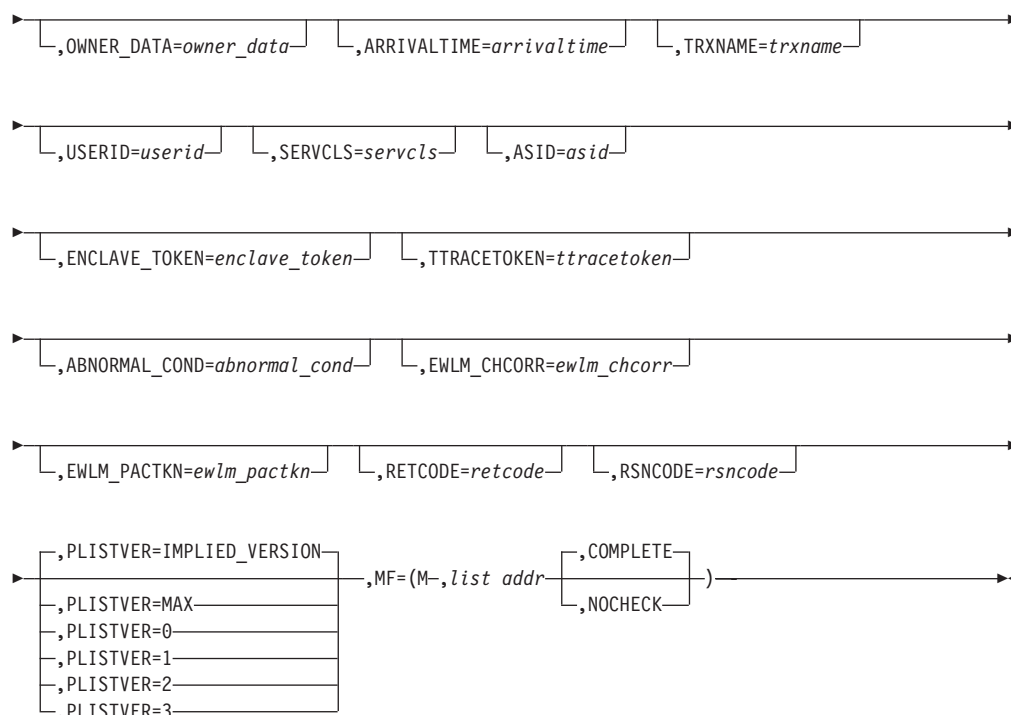
---

## Syntax

### main diagram







## Parameters

The parameters are explained as follows:

### *name*

An optional symbol, starting in column 1, that is the name on the IWMSEXTR macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

### **,ABNORMAL\_COND=***abnormal\_cond*

An optional output parameter, which contains the current information about abnormal conditions which were either recorded in the input monitoring environment or which were propagated to it via IWMSEXFER Function(Return). Multiple conditions may exist.

The mask, `Iwmmabnl_Scope_LocalMVS`, may be used to determine whether an abnormality which only affects work on the current MVS image was recorded.

The mask, `Iwmmabnl_Scope_Sysplex`, may be used to determine whether an abnormality which affects work on all MVS images in the sysplex was recorded.

To determine whether a condition was recorded, merely AND the field supplied for ABNORMAL\_COND with the relevant mask. The result will be nonzero when the condition is true, zero when the condition is false.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32 bit field.

### **,ARRIVALTIME=***arrivaltime*

An optional output parameter, which contains the work arrival time in STCK format.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 64-bit field.

## IWMEXTR Macro

### **,ASID=*asid***

An optional output parameter, which contains the address space ID. When the monitoring environment is not associated with an address space, the output will be a halfword of binary zeros. This keyword is available on z/OS R2 and higher.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 16-bit field.

### **,ENCLAVE\_TOKEN=*enclave\_token***

An optional output parameter, which contains the enclave token. When the monitoring environment is not associated with an enclave, the output will be a doubleword of binary zeros.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 64 bit field.

### **,EWLM\_CHCORR=*ewlm\_chcorr***

An optional output parameter, which contains the cross-platform Enterprise Workload Management (EWLM) correlator of the work request associated with the monitoring environment.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a character field.

### **,EWLM\_PACTKN=*ewlm\_pactkn***

An optional output parameter, which contains the cross-platform Enterprise Workload Management (EWLM) parent correlator token of the work request associated with the monitoring environment.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a character field.

### **MONTKN=*montkn***

A required input parameter, which contains the delay monitoring token

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32 bit field.

### **,OWNER\_DATA=*owner\_data***

An optional output parameter, which is to receive the data established by the user/owner of the monitoring environment. The format is undefined to MVS. When the monitoring environment is not associated with an OWNER\_TOKEN value, the output will be a word of binary zeros.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32 bit field.

### **,OWNER\_TOKEN=*owner\_token***

An optional output parameter, which is to receive the token established by the user/owner of the monitoring environment. The format is undefined to MVS. When the monitoring environment is not associated with an OWNER\_TOKEN value, the output will be a word of binary zeros.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32 bit field.

### **,PLISTVER=IMPLIED\_VERSION**

#### **,PLISTVER=MAX**

#### **,PLISTVER=0**

#### **,PLISTVER=1**

#### **,PLISTVER=3**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an

optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form; in this way, MAX ensures that the parameter list does not overwrite nearby storage.

- **0**, which supports all parameters except those specifically referenced in higher versions.
- **1**, which supports both the following parameters and those from version 0:

TTRACETOKEN

- **2**, which supports both the following parameters and those from version 0 and 1:

ASID

ENCLAVE\_TOKEN

- **3**, which supports both the following parameters and those from version 0, 1, and 2:

EWLM\_CHCORR

EWLM\_PACTKN

**To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX
- A decimal value of 0, 1, 2, or 3

**,RETCODE=retcode**

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,RSNCODE=rsncode**

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,SERVCLS=servcls**

An optional output parameter, which contains the service class token. When the monitoring environment is not associated with a service class token, the output will be a word of binary zeros.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32 bit field.

**,TRXNAME=trxname**

An optional output parameter, which contains the transaction name. The field will be all blanks when NO\_TRXNAME was specified on IWM4MINI.

## IWMEXTR Macro

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

**,TTRACETOKEN=*ttracetoken***

An optional output parameter, which contains the transaction trace token associated with the work request. The field will be all zero when NO\_TTRACETOKEN was specified on IWM4MINI.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32-character field.

**,USERID=*userid***

An optional output parameter, which contains the local userid associated with the work request. The field will be all blanks when NO\_USERID was specified on IWM4MINI.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

---

## ABEND Codes

None.

---

## Return and Reason Codes

When the IWMEXTR macro returns control to your program:

- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, if you coded RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 43. Return and Reason Codes for the IWMEXTR Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> IwmRetCodeOk <b>Meaning:</b> Successful completion. <b>Action:</b> None required.
4	—	<b>Equate Symbol:</b> IwmRetCodeWarning <b>Meaning:</b> Successful completion, unusual conditions noted.
4	xxxx0402	<b>Equate Symbol:</b> IwmRsnCodeNoMonEnv <b>Meaning:</b> Monitoring token indicates that no monitoring environment exists. <b>Action:</b> None required.
4	xxxx040C	<b>Equate Symbol:</b> IwmRsnCodeMonEnvLacksInfo <b>Meaning:</b> Input monitoring environment does not contain the necessary information to return the data requested. <b>Action:</b> None required.
8	—	<b>Equate Symbol:</b> IwmRetCodeInvocError <b>Meaning:</b> Invalid invocation environment or parameters.

Table 43. Return and Reason Codes for the IWMSEXTR Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0820	<p><b>Equate Symbol:</b> IwmRsnCodeBadMonEnv</p> <p><b>Meaning:</b> Monitoring environment does not pass short form verification.</p> <p><b>Action:</b> Check for possible storage overlay.</p>
8	xxxx0895	<p><b>Equate Symbol:</b> IwmRsnCodeEWLMServNotEnabled</p> <p><b>Meaning:</b> Service is not enabled because monitoring environment can not be associated with EWLM work requests.</p> <p><b>Action:</b> Specify the parameter EWLM_CHCORR or EWLM_PACTKN only when the monitoring environment is created with IWMMCREA EWLM=YES or the address space is connected with IWMCONN EWLM=YES and the connect token is passed to IWMMCREA when creating the monitoring environment.</p>



---

## Chapter 36. IWMMNTFY – Notify of Work Execution Completion

The primary purpose of this service is to notify MVS that the execution phase for a work request associated with a monitoring environment has just completed. This may represent the entire work request OR merely a subset of it. An indication is also given as to whether the monitoring environment should be disassociated from the work request or not. When DISASSOCIATE(YES) is specified, this service will render the information associated with the monitoring environment unpredictable. To associate a work request with the monitoring environment following use of DISASSOCIATE(YES), first use Initialize Mode(Reset) or Relate/Transfer.

---

### Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Supervisor state. PSW key must either be 0 or match the value supplied on IWM4CON when a connect token is passed. PSW key must either be 0 or match the value supplied on IWM4MCRE. PSW key must be 0-7. See restrictions below.
<b>Dispatchable unit mode:</b>	Task or SRB
<b>Cross memory mode:</b>	Any PASN, any HASN, any SASN
<b>AMODE:</b>	31-bit
<b>ASC mode:</b>	Primary
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	LOCAL lock held
<b>Control parameters:</b>	Control parameters must be in the primary address space.

---

### Programming Requirements

1. The macro CVT must be included to use this macro.
2. The macro IWMYCON must be included to use this macro.
3. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
4. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, IWMRSNCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.

---

### Restrictions

1. Caller is responsible for error recovery
2. Though the caller is required to be enabled, this is not checked. Violation of this restriction may cause disabled program checks which would be the responsibility of the caller's recovery to handle.
3. The monitoring environment must contain the information saved by IWM4MINI, not IWMMRELA
4. The current PSW key must be 0 or match the key specified on IWM4MCRE provided the latter is a system key (0-7).

## IWMMNTFY Macro

5. This service cannot be used for Report-Only Monitoring Environments
6. If the key specified on IWM4MCRE was a user key (8-F), then:
  - PSW key must be 0
  - Current primary must match the primary at the time that IWM4MCRE was invoked. Calling from a subspace is not supported.
7. If a connect token is passed to IWMMNTFY, then:
  - The connect token must be enabled for using the WLM Work Management services (specifying WORK\_MANAGER=YES on IWM4CON).
  - If the key specified on IWM4CON was a user key (8-F), then:
    - PSW key must be 0
    - Current primary must match the primary at the time that IWM4CON was invoked. Calling from a subspace is not supported.
8. The caller must serialize to prevent any delay monitoring services from being invoked concurrently for the environment represented by the monitoring token
9. This macro supports multiple versions. Some keywords are only supported by certain versions. Refer to the PLISTVER parameter description for further information.

---

## Input Register Information

Before issuing the IWMMNTFY macro, the caller must ensure that the following general purpose registers (GPRs) contain the specified information:

<b>Register</b>	<b>Contents</b>
<b>13</b>	The address of a 72-byte standard save area in the primary address space

Before issuing the IWMMNTFY macro, the caller does not have to place any information into any AR unless using it in register notation for a particular parameter, or using it as a base register.

---

## Output Register Information

When control returns to the caller, the GPRs contain:

<b>Register</b>	<b>Contents</b>
<b>0-1</b>	Used as work registers by the system
<b>2-13</b>	Unchanged
<b>14-15</b>	Used as work registers by the system.

When control returns to the caller, the ARs contain:

<b>Register</b>	<b>Contents</b>
<b>0-1</b>	Used as work registers by the system
<b>2-13</b>	Unchanged
<b>14-15</b>	Used as work registers by the system.

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

---

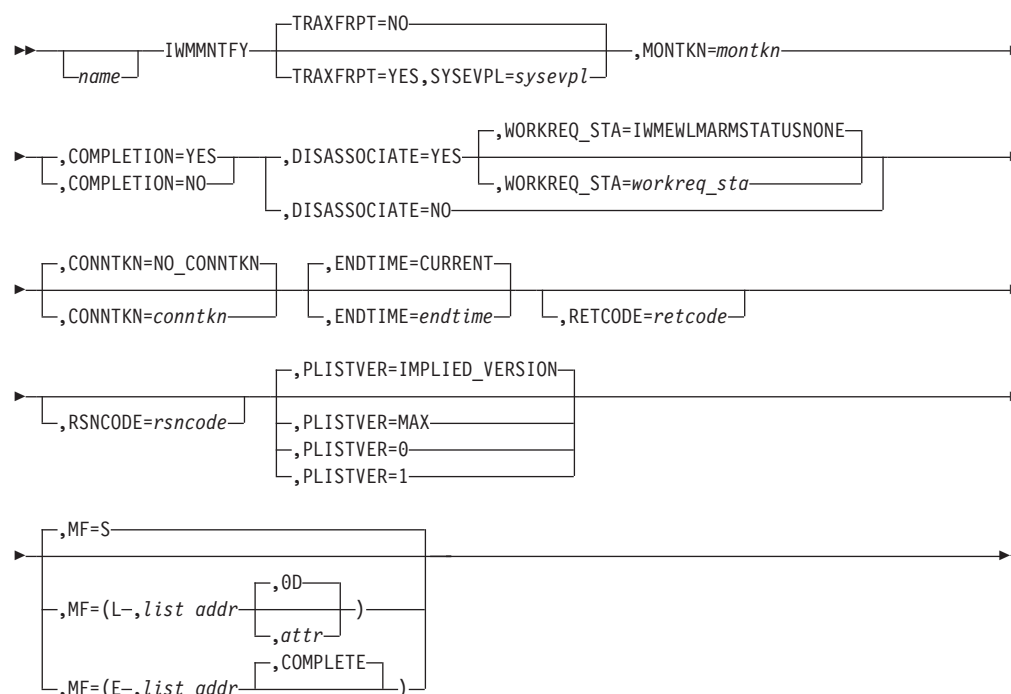
## Performance Implications

None.



## Syntax

### main diagram



## Parameters

The parameters are explained as follows:

### *name*

An optional symbol, starting in column 1, that is the name on the IWMNTFY macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

### **,COMPLETION=YES**

### **,COMPLETION=NO**

A required parameter, which indicates whether the (or possibly one of several) major execution phase(s) is(are) now complete.

### **,COMPLETION=YES**

indicates that execution for an entire phase of processing has now completed. Typically IWMNTFY Completion(Yes) would be issued as a result of the completion of the transaction program for the work request. When a work request is comprised of several (typically cascaded) transaction programs, each would correspond to an invocation of IWMNTFY Completion(Yes).

The execution time, as given by the difference between the IWMNTFY ENDTIME value and the Execution start time (established via IWM4MINI), will be added to the running total execution time for the service class. There may still be "output" processing left to perform for the work request, which time would be accounted for via IWMRPT. There may also be operations corresponding to hardening of the database data outside the scope of NOTIFY.

### **,COMPLETION=NO**

indicates that this invocation of Notify does not correspond to the completion of an entire execution segment. Instead this invocation of Notify corresponds to the portion of the work request represented by the monitoring environment. For example, use Completion(No) when this portion of processing behaves like a subroutine in the execution phase, which is therefore a subset of the execution time passed in another NOTIFY.

The execution time, as given by the difference between the IWMMNTFY ENDTIME value and the Execution start time (established via IWM4MINI), will be treated separately from that passed for Completion(Yes), since otherwise there would be double-counting.

### **,CONNTKN=*conntkn***

### **,CONNTKN=NO\_CONNTKN**

An optional input parameter, which is returned by IWM4CON. The default is NO\_CONNTKN. indicates that no connect token is passed.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32 bit field.

### **,DISASSOCIATE=YES**

### **,DISASSOCIATE=NO**

A required parameter, which indicates whether the work request should be disassociated from the monitoring environment or not.

### **,DISASSOCIATE=YES**

indicates that the work request should be disassociated from the monitoring environment.

### **,DISASSOCIATE=NO**

indicates that the work request should not be disassociated from the monitoring environment.

### **,ENDTIME=*endtime***

### **,ENDTIME=CURRENT**

An optional input parameter, which specifies the ending execution time for the transaction in STCK format. The default is CURRENT. indicates that the current time should be used.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 64 bit field.

### **,MF=S**

### **,MF=(L,*list addr*)**

### **,MF=(L,*list addr,attr*)**

### **,MF=(L,*list addr,OD*)**

### **,MF=(E,*list addr*)**

### **,MF=(E,*list addr,COMPLETE*)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

*,list addr*

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

*,attr*

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

**,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

**,MONTKN=*montkn***

A required input parameter, which contains the delay monitoring token

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32 bit field.

**,PLISTVER=IMPLIED\_VERSION**

**,PLISTVER=MAX**

**,PLISTVER=0**

**,PLISTVER=1**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form; in this way, MAX ensures that the parameter list does not overwrite nearby storage.

- **0**, which supports all parameters except those specifically referenced in higher versions.
- **1**, which supports both the following parameters and those from version 0:

WORKREQ\_STA

**To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX
- A decimal value of 0, or 1

## IWMMNTFY Macro

**,RETCODE=***retcode*

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,RSNCODE=***rsncode*

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,SYSEVPL=***sysevpl*

When TRAXFRPT=YES is specified, a required input parameter, which is the fully initialized SYSEVENT parameter list, as mapped by IHATRBPL.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 40-character field.

**TRAXFRPT=NO**

**TRAXFRPT=YES**

An optional parameter, which indicated prior to z/OS R3 whether a SYSEVENT TRAXFRPT should be issued when the system was in compatibility mode. This has become irrelevant. However, for compatibility reasons TRAXFRPT can still be set but has no effect. The default is TRAXFRPT=NO.

**TRAXFRPT=NO**

indicates that no SYSEVENT TRAXFRPT should be issued.

**TRAXFRPT=YES**

indicated prior to z/OS R3 that a SYSEVENT TRAXFRPT should be issued when the system was in compatibility mode. This has become irrelevant. However, for compatibility reasons TRAXFRPT can still be set but has no effect.

**,WORKREQ\_STA=***workreq\_sta*

**,WORKREQ\_STA=IWMEWLMARMSTATUSNONE**

When DISASSOCIATE=YES is specified, an optional input parameter, which contains the completion status code of the work request. Available completion status codes (defined in macro IWMYCON) are: \* IwmEwlmArmStatusGood(0), \* IwmEwlmArmStatusAborted(1), \* IwmEwlmArmStatusFailed(2) or \* IwmEwlmArmStatusUnknown(3) The codes above correspond to status codes in the OpenGroup ARM 4.0 Standard (for the meaning of the status codes see the ARM 4.0 Standard at <http://www.opengroup.org/management/arm>). The default is IWMEWLMARMSTATUSNONE. indicates that the COMPLETION parameter value and internal information in the Monitoring Environment will be examined to determine the status of the work request. If COMPLETION=YES is specified and no abnormal event was recorded for the monitoring environment via the IWMMABNL service, the completion status IwmEwlmArmStatusGood will be reported to EWLM. If an abnormal event was reported via IWMMABNL or COMPLETION=NO was specified, the completion status IwmEwlmArmStatusFailed will be reported to EWLM.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

---

## ABEND Codes

None.

## Return and Reason Codes

When the IWMNTFY macro returns control to your program:

- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, if you coded RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 44. Return and Reason Codes for the IWMNTFY Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> <code>IwmRetCodeOk</code> <b>Meaning:</b> Successful completion. <b>Action:</b> None required.
4	—	<b>Equate Symbol:</b> <code>IwmRetCodeWarning</code> <b>Meaning:</b> Successful completion, unusual conditions noted.
4	xxxx0401	<b>Equate Symbol:</b> <code>IwmRsnCodeNoWLM</code> <b>Meaning:</b> System does not support delay monitoring services. This return code is only set when the MVS release is prior to MVS/ESA SP5.1.0. <b>Action:</b> None required.
4	xxxx0404	<b>Equate Symbol:</b> <code>IwmRsnCodeCompatNoSyseventRqd</code> <b>Meaning:</b> Reserved
4	xxxx0405	<b>Equate Symbol:</b> <code>IwmRsnCodeGoalNoMonEnv</code> <b>Meaning:</b> System is in goal mode but the input monitoring token indicates no monitoring environment was established, hence MVS did not receive the information. <b>Action:</b> None required.
8	—	<b>Equate Symbol:</b> <code>IwmRetCodeInvocError</code> <b>Meaning:</b> Invalid invocation environment or parameters.
8	xxxx080C	<b>Equate Symbol:</b> <code>IwmRsnCodeMonEnvLacksData</code> <b>Meaning:</b> Input monitoring environment does not contain the necessary information. <b>Action:</b> Ensure that the monitoring environment was established with the necessary information.
8	xxxx080F	<b>Equate Symbol:</b> <code>IwmRsnCodeNoUserKeyNtfy</code> <b>Meaning:</b> User key routine not allowed to issue Notify. <b>Action:</b> Avoid requesting this function in user key.
8	xxxx0820	<b>Equate Symbol:</b> <code>IwmRsnCodeBadMonEnv</code> <b>Meaning:</b> Input monitoring environment does not pass short form validity checking. <b>Action:</b> Check for possible storage overlay.

## IWMMNTFY Macro

Table 44. Return and Reason Codes for the IWMMNTFY Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0821	<p><b>Equate Symbol:</b> IwmRsnCodeBadConn</p> <p><b>Meaning:</b> Input connect token does not pass validity checking.</p> <p><b>Action:</b> Check for possible storage overlay.</p>
8	xxxx082D	<p><b>Equate Symbol:</b> IwmRsnCodeExStTimeGTEndTime</p> <p><b>Meaning:</b> Input execution start time later than end time.</p> <p><b>Action:</b> Check for possible storage overlay of the parameter list or variable.</p>
8	xxxx0840	<p><b>Equate Symbol:</b> IwmRsnCodeServiceNotEnabled</p> <p><b>Meaning:</b> Caller's space connection is not enabled for this service.</p> <p><b>Action:</b> Avoid requesting this function under the input connection. IWM4CON options must be specified previously to enable this service.</p>
8	xxxx087E	<p><b>Equate Symbol:</b> IwmRsnCodeRoMonEnv</p> <p><b>Meaning:</b> Input monitoring environment is Report-Only.</p> <p><b>Action:</b> Avoid requesting this function for Report-Only PBs.</p>
8	xxxx0895	<p><b>Equate Symbol:</b> IwmRsnCodeEWLMServNotEnabled</p> <p><b>Meaning:</b> Service is not enabled because monitoring environment can not be associated with EWLM work requests.</p> <p><b>Action:</b> Specify the parameter WORKREQ_STA only when the monitoring environment is created with IWMMCREA EWLM=YES or the address space is connected with IWMCONN EWLM=YES and the connect token is passed to IWMMCREA when creating the monitoring environment.</p>
8	xxxx0897	<p><b>Equate Symbol:</b> IwmRsnCodeTranStatusInvalid</p> <p><b>Meaning:</b> Passed work request completion status is not valid.</p> <p><b>Action:</b> Check the EWLM ARM interface specification for valid completion status values.</p>
C	—	<p><b>Equate Symbol:</b> IwmRetCodeEnvError</p> <p><b>Meaning:</b> Environmental error.</p>
C	xxxx0C02	<p><b>Equate Symbol:</b> IwmRsnCodeReportingSusp</p> <p><b>Meaning:</b> Reserved SYSEVENT TRAXFRPT invoked, but reporting is temporarily suspended for one of the following reasons:</p> <ul style="list-style-type: none"> <li>•</li> <li>• RMF workload activity reporting is not active</li> <li>• There is no installation control specification (IEAICSxx parmlib member with RPGN specified for some subsystem other than TSO) in effect. No data reported but a later reissue could be successful.</li> </ul> <p><b>Action:</b> Invoke the function when the conditions are alleviated.</p>

Table 44. Return and Reason Codes for the IWMNTFY Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
C	xxxx0C03	<p><b>Equate Symbol:</b> lwmRsnCodeSyseventNoWorkElt</p> <p><b>Meaning:</b> SYSEVENT TRAXFRPT invoked, but no work element was available to save the input information.</p> <p><b>Action:</b> Invoke the function when the conditions are alleviated. This condition may be due to a common storage shortage condition.</p>
C	xxxx0C04	<p><b>Equate Symbol:</b> lwmRsnCodeNtfyNoWorkElt</p> <p><b>Meaning:</b> Notify routine invoked, but no work element was available to save the input information.</p> <p><b>Action:</b> Invoke the function when the conditions are alleviated. This condition may be due to a common storage shortage condition.</p>
C	xxxx0C06	<p><b>Equate Symbol:</b> lwmRsnCodeNoEndTime</p> <p><b>Meaning:</b> No end time was supplied to the service and STCK gave a non-zero condition code.</p> <p><b>Action:</b> No action required.</p>

## IWMMNTFY Macro



---

## Chapter 37. IWMMRELA – Relate Monitoring Environment Service

The calling subsystem work manager can use IWMMRELA to relate two different monitoring environments that are associated with the same work request. IWMMRELA initializes a monitoring environment, called a dependent monitoring environment, and associates it with a previously established monitoring environment, called a parent monitoring environment.

You can use IWMMRELA when you do not have direct access to the information required by IWM4MINI. If the caller has the monitoring token for a parent monitoring environment that is previously established for the same work request, you should provide it in the PARENTMONTKN parameter. If the caller does not pass the parent monitoring token, you can use PARENTP=FINDACTIVE to specify that the parent monitoring environment is the active monitoring environment owned by the home address space and which is associated with the TCB provided via PARENTTCB.

IWMMRELA must be used together with IWMMXFER to ensure that the dependent monitoring environment is a valid representation for the work request.

Optionally with this macro, you can use the OWNER\_TOKEN and OWNER\_DATA parameters to use the monitoring environment for your own purposes. You could use the token/data to keep your own information.

---

### Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Problem state, or supervisor state. PSW key must either be 0, or match the value specified on IWM4MCRE.
<b>Dispatchable unit mode:</b>	Task or SRB
<b>Cross memory mode:</b>	Any PASN, any HASN, any SASN
<b>AMODE:</b>	31-bit
<b>ASC mode:</b>	Primary
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	Unlocked when PARENT=FINDACTIVE is specified, otherwise, no restrictions.
<b>Control parameters:</b>	Control parameters must be in the primary address space, except the TCB, if specified, must reside in current home address space.

---

### Programming Requirements

1. You must include the IWMYCON mapping macro in the calling program.
2. If the key specified on IWM4MCRE for the input MONTKN was a user key (8-F), then the following must be true:
  - If you specify PARENTP=YES, then:
    - Primary addressability must exist to the performance block IWM4MCRE obtained (represented by the input MONTKN). You could do this by ensuring that current primary matches primary at the time that

## IWMMRELA Macro

IWM4MCRE was invoked. If this service is invoked in a subspace, the condition may be satisfied by ensuring that the performance block is shared with the base space.

- You cannot specify the list form of this macro. With PARENTP=YES, IWMMRELA produces an inline expansion rather than an out-of-line service, so you do not need a parameter list. Registers 0, 1, 14, and 15 are not preserved across the expansion.
  - If you specify PARENTP=FINDACTIVE, then the caller must be in non-cross-memory mode (PASN=SASN=HASN). That is, the current primary (and home) must match the primary (and home) at the time that IWM4MCRE was invoked.
3. If the key specified on IWM4MCRE for the parent environment was a user key (8-F), then either primary or secondary addressability must exist to the monitoring environment for the parent environment.
  4. Both monitoring environments must be established on the same MVS image.
  5. The caller must serialize to prevent any delay monitoring services from being invoked concurrently for the monitoring environment.
  6. When PARENTP=YES, the caller must provide recovery.

---

## Restrictions

None.

---

## Input Register Information

Before issuing the IWMMRELA macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

---

## Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Reason code if GR 15 return code is non-zero.
1	Used as a work register by the system.
2 - 13	Unchanged
14	Used as a work register by the system.
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0 - 1	Used as a work register by the system.
2 - 13	Unchanged
14 - 15	Used as a work register by the system.

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

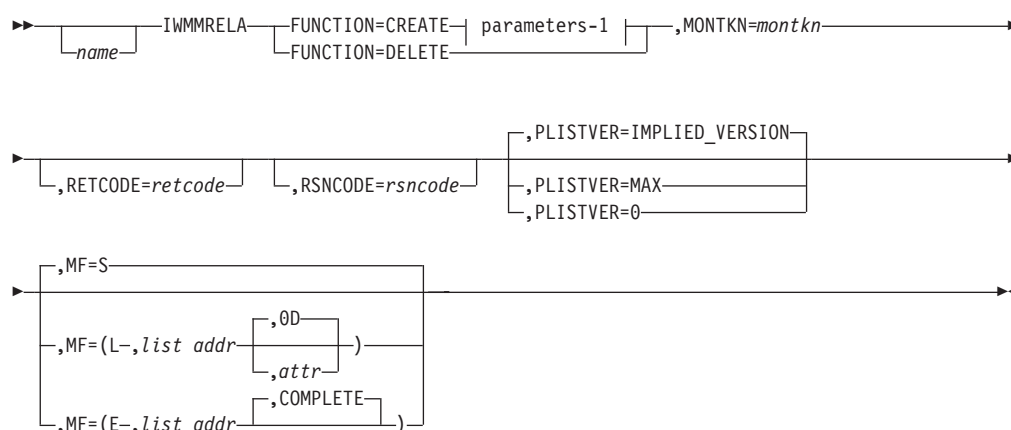
---

## Performance Implications

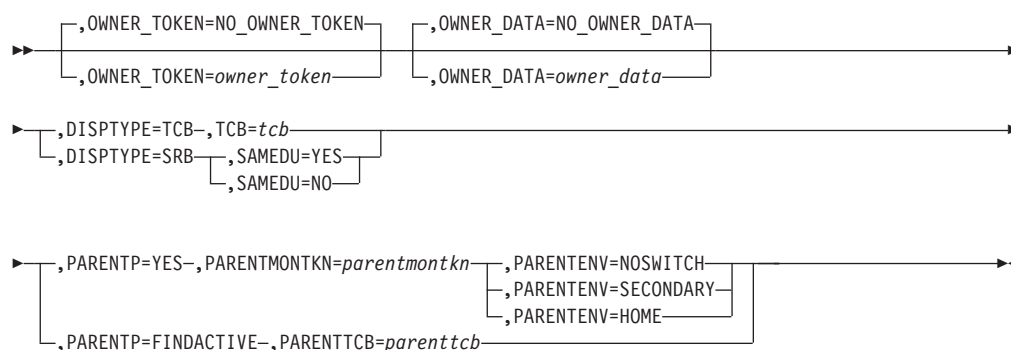
None.

## Syntax

### main diagram



### parameters-1



## Parameters

The parameters are explained as follows:

### *name*

An optional symbol, starting in column 1, that is the name on the IWMMRELA macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

### **,DISPTYPE=TCB**

### **,DISPTYPE=SRB**

When FUNCTION=CREATE is specified, a required parameter, which describes the dispatchable units which participate in processing work requests associated with the monitoring environment represented by the monitoring token (MONTKN).

### **,DISPTYPE=TCB**

indicates that work requests run in TCB mode under a TCB within the current home address space. Note that in cross-memory mode, this may be different from the current primary address space.

### **,DISPTYPE=SRB**

## IWMMRELA Macro

indicates that work requests run in SRB mode within the current home address space.

### **FUNCTION=CREATE**

### **FUNCTION=DELETE**

A required parameter, which indicates whether the relationship is being established or inactivated.

### **FUNCTION=CREATE**

indicates that the relationship is being established.

### **FUNCTION=DELETE**

which indicates that the relationship is being inactivated.

Note that this produces an inline expansion rather than an out-of-line service, so that no parameter list is needed. Therefore, the MF keyword is not supported when this option is specified. Registers 0, 1, 14, and 15 are not preserved across the expansion.

### **,MF=S**

### **,MF=(L,list addr)**

### **,MF=(L,list addr,attr)**

### **,MF=(L,list addr,0D)**

### **,MF=(E,list addr)**

### **,MF=(E,list addr,COMPLETE)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

### **,list addr**

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

### **,attr**

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

### **,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

### **,MONTKN=montkn**

A required input parameter, which contains the delay monitoring token for the dependent environment.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32-bit field.

**,OWNER\_DATA=owner\_data**

**,OWNER\_DATA=NO\_OWNER\_DATA**

When FUNCTION=CREATE is specified, an optional input parameter, which contains data maintained by the user or owner of the monitoring environment. The format is undefined to MVS. The default is NO\_OWNER\_DATA which indicates that no owner data is provided.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32-bit field.

**,OWNER\_TOKEN=owner\_token**

**,OWNER\_TOKEN=NO\_OWNER\_TOKEN**

When FUNCTION=CREATE is specified, an optional input parameter, which contains a token maintained by the user or owner of the monitoring environment. The format is undefined to MVS. The default is NO\_OWNER\_TOKEN which indicates that no owner token is provided on this service.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32-bit field.

**,PARENTENV=NOSWITCH**

**,PARENTENV=SECONDARY**

**,PARENTENV=HOME**

When PARENTP=YES and FUNCTION=CREATE are specified, a required parameter, which describes whether a space switch is needed to access the parent monitoring environment.

**,PARENTENV=NOSWITCH**

indicates that NO space switch is needed to access the parent monitoring environment. This would be appropriate if the parent monitoring environment was established (by IWM4MCRE) to be used by routines in a specific system key or if it was established to be used in a specific user key in the current primary.

**,PARENTENV=SECONDARY**

indicates that the parent monitoring environment was established in current secondary (for use by a specific user key).

**,PARENTENV=HOME**

indicates that the parent monitoring environment was established in current home (for use by a specific user key). Use of this option requires that the program must reside in the MVS common area.

**,PARENTMONTKN=parentmontkn**

When PARENTP=YES and FUNCTION=CREATE are specified, a required input parameter, which contains the delay monitoring token for the parent environment, for example, the monitoring environment which was established earlier and contains the characteristics to be inherited.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32-bit field.

**,PARENTP=YES**

**,PARENTP=FINDACTIVE**

When FUNCTION=CREATE is specified, a required parameter, which describes whether the parent monitoring environment is known or not.

### **,PARENTP=YES**

indicates that the parent monitoring environment is known.

Note, that this produces an inline expansion rather than an out-of-line service, so that no parameter list is needed. Therefore, the MF keyword is not supported when this option is specified. Registers 0, 1, 14, and 15 are not preserved across the expansion.

### **,PARENTP=FINDACTIVE**

indicates that the parent monitoring environment is unknown, but requests that the input monitoring environment be related to the active monitoring environment owned by the current HOME address space and which is associated with the TCB specified by PARENTTCB and which has no further continuations to other monitoring environments. When no such monitoring environment exists, the input monitoring environment will be related to the current home address space.

### **,PARENTTCB=*parenttcb***

When PARENTP=FINDACTIVE and FUNCTION=CREATE are specified, a required input parameter, which defines the TCB owned by the current home address space associated with a monitoring environment via Initialize/Relate Disptype=TCB,TCB= . This TCB need not be the owner of the monitoring environment. Note that this name is not the pointer to the TCB, but the name of the data area containing the TCB. A typical invocation might replace xTCB with TCB.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a character field.

### **,PLISTVER=IMPLIED\_VERSION**

#### **,PLISTVER=MAX**

#### **,PLISTVER=0**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form, when both are assembled with the same level of the system. In this way, MAX ensures that the parameter list does not overwrite nearby storage.

- **0**, if you use the currently available parameters.

**To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX
- A decimal value of 0

**,RETCODE=*retcode***

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,RSNCODE=*rsncode***

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,SAMEDU=YES****,SAMEDU=NO**

When DISPTYPE=SRB and FUNCTION=CREATE are specified, a required parameter, which describes whether the dependent monitoring environment associated with MONTKN is running under the same dispatchable unit as the parent. In that case, it would behave as a "subroutine" and execute on the same processor (CP, a.k.a. CPU) as the parent environment.

**,SAMEDU=YES**

indicates that the work request runs as a subroutine of the parent.

YES may not be specified when PARENTP(FINDACTIVE) is coded.

**,SAMEDU=NO**

indicates that the work request runs in SRB mode and is independent of the parent dispatchable unit.

**,TCB=*tcb***

When DISPTYPE=TCB and FUNCTION=CREATE are specified, a required input parameter, which defines the TCB within the current home address space which will serve the work request. Note that this name is not the pointer to the TCB, but the name of the data area containing the TCB. A typical invocation might replace xTCB with TCB.

Generally, the input TCB specified should be the TCB under which the work request (e.g. transaction program) runs and under which the delay information is recorded (in spite of the fact that task switches may occur).

**To code:** Specify the RS-type address, or address in register (2)-(12), of a character field.

## ABEND Codes

None.

## Return and Reason Codes

When the IWMMRELA macro returns control to your program:

- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, if you coded RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

## IWMMRELA Macro

Table 45. Return and Reason Codes for the IWMMRELA Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> IwmRetCodeOk <b>Meaning:</b> Successful completion.
4	—	<b>Equate Symbol:</b> IwmRetCodeWarning <b>Meaning:</b> Successful completion, unusual conditions noted.
4	xxxx0401	<b>Equate Symbol:</b> IwmRsnCodeNoWLM <b>Meaning:</b> System does not support delay monitoring services. This return code is only set when the MVS release is prior to MVS/ESA SP5.1.0.
4	xxxx0402	<b>Equate Symbol:</b> IwmRsnCodeNoMonEnv <b>Meaning:</b> Input monitoring token indicates no monitoring environment was established.
4	xxxx0406	<b>Equate Symbol:</b> IwmRsnCodeNoParEnv <b>Meaning:</b> No parent monitoring environment was established. The input dependent monitoring environment is now related to the Home address space.
8	—	<b>Equate Symbol:</b> IwmRetCodeInvocError <b>Meaning:</b> Invalid invocation environment or parameters
8	xxxx0802	<b>Equate Symbol:</b> IwmRsnCodeXmemUserKeyTkn <b>Meaning:</b> Caller is in cross-memory mode while the token was obtained in user key.
8	xxxx0803	<b>Equate Symbol:</b> IwmRsnCodeDisabled <b>Meaning:</b> Caller is disabled
8	xxxx0804	<b>Equate Symbol:</b> IwmRsnCodeLocked <b>Meaning:</b> Caller is locked
8	xxxx080B	<b>Equate Symbol:</b> IwmRsnCodeBadPI <b>Meaning:</b> Error accessing parameter list.
8	xxxx081A	<b>Equate Symbol:</b> IwmRsnCodeCallerNotAuthDepEnv <b>Meaning:</b> Caller is not authorized to update the dependent monitoring environment
8	xxxx0820	<b>Equate Symbol:</b> IwmRsnCodeBadMonEnv <b>Meaning:</b> Monitoring environment does not pass verification.
8	xxxx0822	<b>Equate Symbol:</b> IwmRsnCodeBadParEnv <b>Meaning:</b> Parent monitoring environment does not pass verification.
8	xxxx0823	<b>Equate Symbol:</b> IwmRsnCodeDatoff <b>Meaning:</b> Caller invoked service while DATOFF
8	xxxx0824	<b>Equate Symbol:</b> IwmRsnCodeAmode24 <b>Meaning:</b> Caller invoked service but was in 24 bit addressing mode.
8	xxxx0825	<b>Equate Symbol:</b> IwmRsnCodeAscModeNotPrimary <b>Meaning:</b> Caller invoked service but was not DAT on Primary ASC mod



Table 45. Return and Reason Codes for the IWMRELA Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0827	<b>Equate Symbol:</b> IwmRsnCodeRsvdNot0 <b>Meaning:</b> Reserved field in parameter list was non-zero
8	xxxx0828	<b>Equate Symbol:</b> IwmRsnCodeBadVersion <b>Meaning:</b> Version number in parameter list is not valid.
8	xxxx0829	<b>Equate Symbol:</b> IwmRsnCodeBadOptions <b>Meaning:</b> Parameter list omits required parameters or supplies mutually exclusive parameters or provides data associated with options not selected.
8	xxxx087E	<b>Equate Symbol:</b> IwmRsnCodeRoMonEnv <b>Meaning:</b> Monitoring environment is report only
8	xxxx087F	<b>Equate Symbol:</b> IwmRsnCodeRoParEnv <b>Meaning:</b> Parent monitoring environment is report only
8	xxxx08A4	<b>Equate Symbol:</b> IwmRsnCodeBPParEnv <b>Meaning:</b> Parent monitoring environment is buffer pool management only
10	—	<b>Equate Symbol:</b> IwmRetCodeCompError: <b>Meaning:</b> Component error

## Example

To relate two monitoring environments where an address space switch is not required, specify:

```
IWMRELA FUNCTION=CREATE,MONTKN=(R7),PARENTP=YES,
        PARENTMONTKN=(R8),PARENTENV=NOSWITCH,
        DISPTYPE=SRB,SAMEDU=YES,
        RETCODE=RCODE,RSNCODE=RSN
```

## IWMMRELA Macro

---

## Chapter 38. IWMMSWCH – Switch Monitoring Environment

The IWMMSWCH macro allows the caller to indicate that the delay information for a work request may now also reside in another monitoring environment which is not related (via IWMMRELA) to the current environment. You can also use IWMMSWCH to indicate that there is no further information for the current work request beyond the current monitoring environment.

The scope of this service is restricted to the input monitoring environment; no other monitoring environments are accessed or otherwise involved.

---

### Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Problem state, or supervisor state. PSW key must either be 0, or match the value specified on IWM4MCRE.
<b>Dispatchable unit mode:</b>	Task or SRB
<b>Cross memory mode:</b>	Any PASN, any HASN, any SASN
<b>AMODE:</b>	31-bit
<b>ASC mode:</b>	Primary
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	Suspend locks are allowed, as are FRRs
<b>Control parameters:</b>	Control parameters must be in the primary address space.

---

### Programming Requirements

1. You must include the IWMYCON and CVT mapping macros in the calling program.
2. The caller is responsible for error recovery.
3. If you specify FUNCTION=CONTINUE, you cannot specify the list form of this macro. With FUNCTION=CONTINUE, IWMMSWCH produces an inline expansion rather than an out-of-line service, so you do not need a parameter list. Registers 0, 1, 14, and 15 are not preserved across the expansion.
4. If the key specified on IWM4MCRE was a user key (8 - F) then primary addressability must exist to the monitoring environment IWM4MCRE obtained. You could do this by making sure the primary address space matches the primary at the time IWM4MCRE was invoked.
5. The caller must serialize to prevent any delay monitoring services from being invoked concurrently for the monitoring environment.

---

### Restrictions

You cannot use FUNCTION=CONTINUE when there is an outstanding continuation established by the IWMMXFER macro FUNCTION=CONTINUE.

---

### Input Register Information

Before issuing the IWMMSWCH macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

## Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Reason code if the return code in GPR 15 is not 0, otherwise, used as a work register by the system.
1	Used as a work register by the system.
2 - 13	Unchanged
14	Used as a work register by the system.
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0 - 1	Used as a work register by the system.
2 - 13	Unchanged
14 - 15	Used as a work register by the system.

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

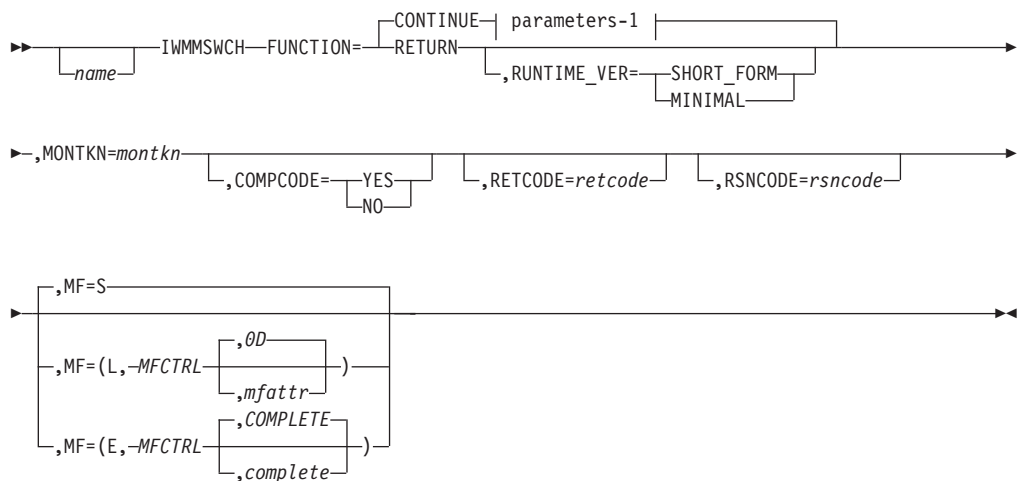
## Performance Implications

None.

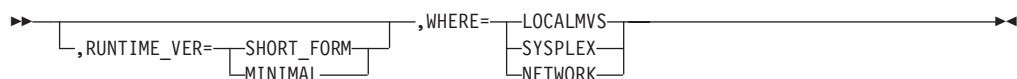
## Syntax

The syntax of the IWMMSWCH macro is as follows:

### main diagram



### parameters-1



## Parameters

The parameters are explained as follows:

### **,FUNCTION=CONTINUE**

### **,FUNCTION=RETURN**

Required input parameter that specifies where there may be one or more monitoring environments which represent current information about the work request. FUNCTION indicates further continuations, and does not deal with any parent environment that may exist.

Use FUNCTION=CONTINUE to indicate that the current monitoring environment continues elsewhere.

If you specify FUNCTION=CONTINUE, you cannot specify the MF keyword. With FUNCTION=CONTINUE, IWMMSWCH produces an inline expansion rather than an out-of-line service, so that you do not need a parameter list. Registers 0, 1, 14, and 15 are not preserved across the expansion.

Use FUNCTION=RETURN to indicate that continuations of the current monitoring environment have completed. Registers 0, 1, 14, and 15 are not preserved across the expansion.

### **,RUNTIME\_VER=SHORT\_FORM**

### **,RUNTIME\_VER=MINIMAL**

Optional parameter that specifies what level of runtime verification will be performed.

RUNTIME\_VER=SHORT\_FORM indicates that checking should verify that a monitoring environment is established and passes a short form of verification prior to being used.

RUNTIME\_VER=MINIMAL indicates that checking should assume that if a monitoring environment is created, it is valid and useable.

### **,WHERE=LOCALMVS**

### **,WHERE=SYSPLEX**

### **,WHERE=NETWORK**

Required input parameter for FUNCTION=CONTINUE that specifies where there may be another monitoring environment.

Use WHERE=LOCALMVS to indicate that another monitoring environment may exist on the current MVS.

Use WHERE=SYSPLEX to indicate that another monitoring environment may exist in the current sysplex, but is not expected to be on the current MVS image.

Use WHERE=NETWORK to indicate that another monitoring environment may exist, but is not expected to be in the current MVS sysplex.

### **,MONTKN=*montkn***

Required input parameter that specifies the monitoring token.

**To code:** Specify the RS-type name or address (using a register from 2 to 12) of a 32 bit field containing the monitoring token.

### **,COMPCODE=NO**

### **,COMPCODE=YES**

Optional input parameter that specifies whether you need completion status for IWMMSWCH.

## IWMMSWCH Macro

COMPCODE=NO specifies that you do not need completion status. Registers 0, 15 cannot be used as reason code and return code registers upon completion of the macro expansion. If you specify COMPCODE=NO, you cannot specify RETCODE nor RSNCODE.

COMPCODE=YES specifies that you need completion status.

### **,RETCODE=***retcode addr*

Optional output parameter that specifies where the system is to store the return code. The return code is also in GPR 15.

**To code:** Specify the name (RS-type) or address (using a register from 2 to 12) of a fullword to contain the return code.

### **,RSNCODE=***rsncode addr*

Optional output parameter that specifies where the system is to store the reason code. The reason code is also in GPR 0.

**To code:** Specify the name (RS-type) or address (using a register from 2 to 12) of a fullword to contain the reason code (if any).

### **,MF=S**

**MF=(L,***mfctrl,mfattr***)**

**MF=(E,***mfctrl,COMPLETE***)**

Use MF=S to specify the standard form, which places parameters into an inline parameter list and invokes the IWM4CON macro service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require re-entrant code. The list form defines an area of storage that the execute form uses to store the parameters.

Use MF=E to specify the execute form of the macro. Use the execute form with the list form of the macro for applications that require re-entrant code. The execute form stores the parameters into the storage area defined by the list form and generates the macro invocation to transfer control to the service.

### **,mfctrl**

Use this output parameter to specify the name of the storage area to contain the parameters.

**To code:** Specify the name (RS-type) or address (using a register from 2 to 12) of the storage area containing the parameter list.

### **,mfattr**

Use this input parameter to specify the name of a 1 to 60 character storage area that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *,mfattr* the system provides a value of 0D, which forces the parameter on a doubleword boundary.

### **,COMPLETE**

Use this input parameter to specify that the system check for required parameters and supply defaults for omitted optional parameter.

---

## ABEND Codes

None.

---

## Return and Reason Codes

When IWMMSWCH macro returns control to your program, GPR 15 contains a return code. When the return code is non-zero, then GPR 0 contains a reason code.

Hexadecimal Return Code	Hexadecimal Reason Code	Meaning
00		<b>Meaning:</b> Successful completion.
04	0401	<b>Meaning:</b> Warning. The system does not support delay monitoring services. This return code is only set when the MVS release is prior to MVS/ESA SP 5.2
04	0402	<b>Meaning:</b> Warning. Input monitoring token indicates no monitoring environment was established.
04	0407	<b>Meaning:</b> Warning. Switch return was from a monitoring environment with an outstanding continuation.
08	081C	<b>Meaning:</b> Program error. Outstanding continuation exists.
08	0820	<b>Meaning:</b> Program error. Monitoring environment does not pass short form verification.

---

## Example

To indicate that the current monitoring environment continues only once elsewhere in the sysplex, specify:

```
IWMMSWCH FUNCTION=CONTINUE,WHERE=SYSPLEX,MONTKN=(R7),
          RETCODE=RCODE,RSNCODE=RSN
```





---

## Chapter 39. IWMMXDC – Exit For Collecting Resource Data

IWMMXDC invokes the resource data collection exit specified. This exit returns information about the usage of the buffer pool or other resources which may be responsible for delays to work requests. The return and reason codes for IWMMXDC are those set by the exit invoked.

The exit environment is described in the following. The parameter list is in the same key as the PSW at the time of invocation and is in pageable storage addressable from the current address space. Upon entry to the exit, the register contents are as follows:

- Register 0 = not defined
- Register 1 contains the address of a parameter list as formatted by the list form of this macro, IWMMXDC MF=(L).
- Registers 2-13 = not defined
- Register 14 = return address
- Register 15 = entry-point address

Upon entry to the exit, the access register contents are undefined.

Upon return from the exit, the register contents are as follows:

- Register 0 = Reason code if GR15 return code is non-zero
- Registers 1-14 = not defined (need not be restored to value on entry)
- Register 15 = Return code

Upon return from the exit, the access register contents are unchanged.

---

### Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Supervisor state with the same PSW key as at the time of registration (IWM4MREG).
<b>Dispatchable unit mode:</b>	Task or SRB
<b>Cross-memory mode:</b>	PASN=HASN=SASN. The current home address space must be the same as at the time of registration (IWM4MREG).
<b>AMODE:</b>	31-bit
<b>ASC mode:</b>	Primary
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks may be held. FRRs may be established.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

---

### Programming Requirements

The list form of the macro is intended for use by products supplying a data collection exit.

The execute form of the macro is intended for use by z/OS.

The assembler execute form only initializes the parameter list and calls the exit routine. The following restrictions apply:

- The invoker must save registers required before invoking the macro

## IWMMXDC Macro

- The invoker must restore registers required immediately after invoking the macro, without depending on the exit to preserve any registers
- The invoker must copy the output results directly from the parameter list to local variables and NOT use macro keywords to do this

---

## Restrictions

None.

---

## Input Register Information

Before issuing the IWMMXDC macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

---

## Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Reason code if GR15 return code is non-zero
1	Used as work register by the system
2-13	Unpredictable in assembler form, unchanged in PL/X form
14	Used as work registers by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

---

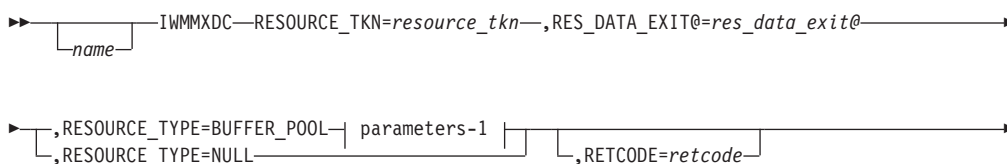
## Performance Implications

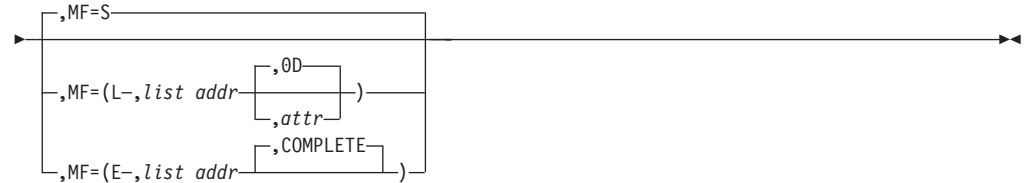
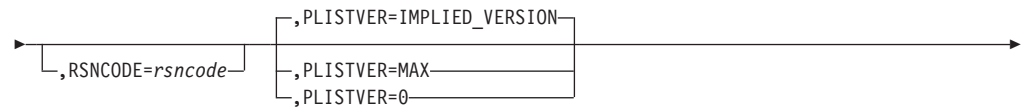
None.

---

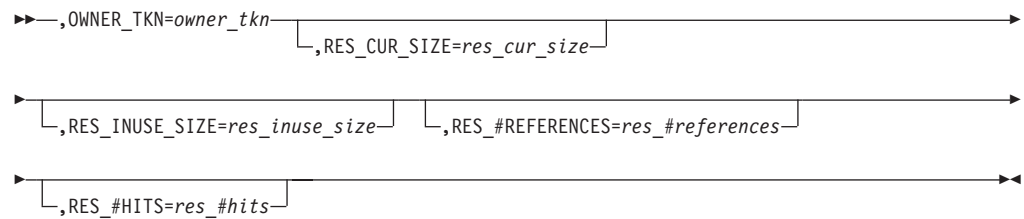
## Syntax

### main diagram





### parameters-1



## Parameters

The parameters are explained as follows:

### *name*

An optional symbol, starting in column 1, that is the name on the IWMXDC macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

### **,MF=S**

**,MF=(L,list addr)**

**,MF=(L,list addr,attr)**

**,MF=(L,list addr,OD)**

**,MF=(E,list addr)**

**,MF=(E,list addr,COMPLETE)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

### *,list addr*

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).



last invocation of the data collection exit. The corresponding field in the parameter list (IWMXDC MF=(L)) must be set by the exit.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

**,RES\_CUR\_SIZE=***res\_cur\_size*

When RESOURCE\_TYPE=BUFFER\_POOL is specified, an optional output parameter, which contains the current size (in 4K pages) associated with the specified resource. The corresponding field in the parameter list (IWMXDC MF=(L)) must be set by the exit.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

**,RES\_DATA\_EXIT@=***res\_data\_exit@*

A required input parameter that contains the address of the resource data collection exit to be invoked.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a pointer field.

**,RES\_INUSE\_SIZE=***res\_inuse\_size*

When RESOURCE\_TYPE=BUFFER\_POOL is specified, an optional output parameter, which contains the current in use size (in 4K pages) associated with the specified resource. The corresponding field in the parameter list (IWMXDC MF=(L)) must be set by the exit.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

**RESOURCE\_TKN=***resource\_tkn*

A required input parameter, which contains the associated WLM resource token which is returned by the registration (IWM4MREG).

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

**,RESOURCE\_TYPE=BUFFER\_POOL**

**,RESOURCE\_TYPE=NULL**

A required parameter, which indicates the type of resource being registered.

**,RESOURCE\_TYPE=BUFFER\_POOL**

indicates that a bufferpool is being collected.

**,RESOURCE\_TYPE=NULL**

indicates that no exit is to be called.

**,RETCODE=***retcode*

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,RSNCODE=***rsncode*

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

---

## ABEND Codes

None.

## Return and Reason Codes

When the IWMMXDC macro returns control to your program:

- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, if you coded RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 46. Return and Reason Codes for the IWMMXDC Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<p><b>Equate Symbol:</b> IwmRetCodeOk</p> <p><b>Meaning:</b> Successful completion.</p> <p><b>Action:</b> None required.</p>

## Example

To register a resource for delay monitoring, specify:

```

        IWMMXDC                                X
            RESOURCE_TKN=RSCTOKEN,             X
            RES_DATA_EXIT@=DATAEXIT@,          X
            RESOURCE_TYPE=BUFFER_POOL,         X
            OWNER_TKN=OWNERTKN,                X
            RETCODE=RC,RSNCODE=RSN
*
* Storage areas
*
RSCTOKEN DS    CL8           WLM resource token
*
DATAEXIT@ DS   AL4           contains the address of the
*                               Resource Data Collection Exit
*                               to be invoked
OWNERTKN DS    CL8           Contains data maintained by
*                               the user
RC        DS    F            Return code
RSN       DS    F            Reason code
    
```

---

## Chapter 40. IWMMXFER – Transfer Monitoring Environment

The purpose of this service is to reflect that the delay information for a work request may now also reside in a dependent monitoring environment (CONTINUE) OR that delay information is no longer present in a dependent monitoring environment (RETURN).

The two monitoring environments referred to above must be related by a previous IWMMRELA invocation. This service requires as input the monitoring token for the dependent environment, which is accessed, but the parent environment must also be updated. This implies that the user must have addressability and update access to the parent monitoring environment. PARENTKEYP and PARENTENV keywords are provided to accommodate these requirements. These restrictions apply even when the Relate was performed using the FINDACTIVE option, though when the monitoring environment is related to the address space characteristics, no key or addressability requirements exist beyond those for the dependent monitoring environment.

---

### Environment

The requirements for the caller are:

**Minimum authorization:**

- Either problem state or supervisor state.
- PSW key must either be 0 or match the value supplied on IWM4MCRE for the (dependent) monitoring token.
- PARENTKEYP(VALUE) may only be specified in supervisor state or with PKM authority to the key specified by PARENTKEY. Note that the key for IWMMXFER is located in bit positions 0-3 (using 0 origin), which is the machine orientation to keeping keys, not the "natural" way of declaring the key value.
- PARENTKEYP(UNKNOWN) may only be specified in supervisor state or with PKM authority to key 0.
- When PARENTKEYP(PSWKEY) is specified, the PSW key must either be 0 or match the value supplied on IWM4MCRE for the parent monitoring environment.
- If FUNCTION=RETURN is specified and the passed MONTKN is associated with an ARM work request (EWLM=YES was specified on IWM4CON (or IWMCONN) and the monitoring environment was created using that CONNTKN), the caller must be in supervisor state or have PKM authority to key 0.

---

<b>Dispatchable unit mode:</b>	Task or SRB
<b>Cross memory mode:</b>	Any PASN, any HASN, any SASN
<b>AMODE:</b>	31-bit
<b>ASC mode:</b>	Primary
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks may be held.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

---

### Programming Requirements

1. The macro IWMYCON must be included to use this macro.

## IWMMXFER Macro

2. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
3. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, IWMRNSCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.
4. Note that specification of FUNCTION(CONTINUE) produces an inline expansion rather than an out-of-line service. Registers 0, 1, 14, and 15 are not preserved across the expansion.

---

## Restrictions

1. If the key specified on IWM4MCRE for the dependent monitoring environment was a user key (8-F), then primary addressability must exist to the performance block IWM4MCRE obtained. This condition is satisfied by ensuring that current primary matches primary at the time that IWM4MCRE was invoked. If this service is invoked in a subspace, the condition may be satisfied by ensuring that the performance block is shared with the base space.
2. If the key specified on IWM4MCRE for the parent environment was a user key (8-F), then either primary OR secondary addressability must exist to the performance block for the parent environment.
3. When FUNCTION(CONTINUEIRETURN) are used, the caller is responsible for error recovery
4. When FUNCTION(CONTINUE) is used, the caller is responsible to ensure that the parent monitoring environment does not already have a continuation (via a previous IWMMXFER or IWMMSWCH) to another (or other) dependent monitoring environment(s).
5. Both monitoring environments must be established on the same MVS image.
6. The caller must serialize to prevent any delay monitoring services from being invoked concurrently for the dependent monitoring environment.
7. The caller and/or the owner of the parent environment must ensure that parent environment is not deleted while between the time that IWMMXFER FUNCTION(CONTINUE) is used and the time that either IWMMXFER FUNCTION(RETURN) is used against the dependent monitoring environment OR IWMMSWCH FUNCTION(RETURN) is used against the parent monitoring environment.
8. Only limited validity checking is done on the input monitoring tokens.
9. This macro supports multiple versions. Some keywords are only supported by certain versions. Refer to the PLISTVER parameter description for further information.

---

## Input Register Information

Before issuing the IWMMXFER macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

---

## Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Reason code if GR15 return code is non-zero
1	Used as work registers by the system
2-13	Unchanged



- 14 Used as work registers by the system
- 15 Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

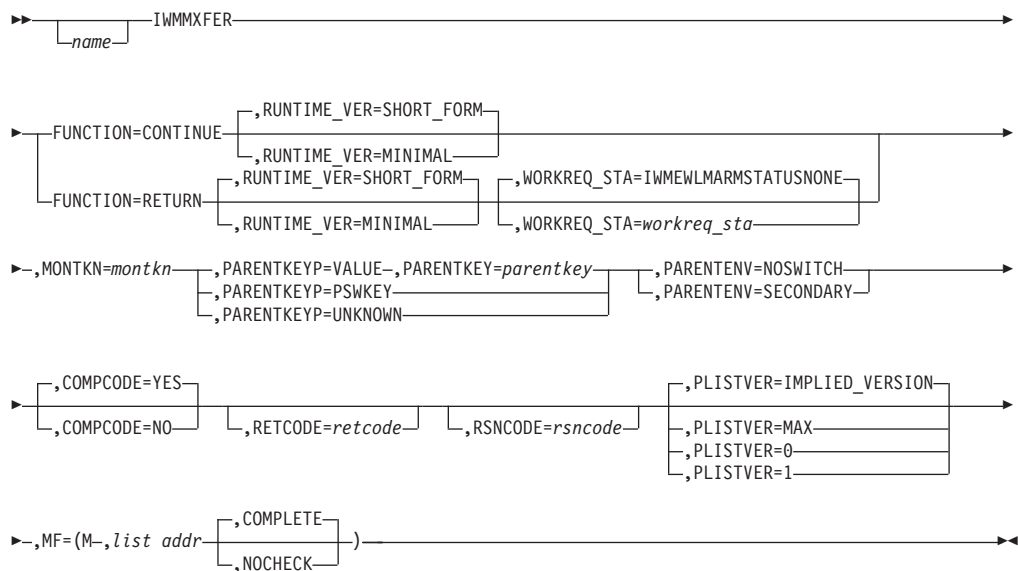
Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

## Performance Implications

None.

## Syntax

### main diagram



## Parameters

The parameters are explained as follows:

### *name*

An optional symbol, starting in column 1, that is the name on the IWMMXFER macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

### **,COMPCODE=YES**

### **,COMPCODE=NO**

An optional parameter, which indicates whether completion status for this service is needed. The default is COMPCODE=YES.

### **,COMPCODE=YES**

indicates that completion status is needed.

### **,COMPCODE=NO**

indicates that completion status is not needed. Registers 0, 15 cannot be used as reason code and return code registers upon completion of the macro expansion. For this reason neither RETCODE NOR RSNCODE may be specified when COMPCODE(NO) is specified.

### **FUNCTION=CONTINUE**

### **FUNCTION=RETURN**

A required parameter, which indicates whether the dependent environment is continuing from or returning to the parent environment.

### **FUNCTION=CONTINUE**

indicates that this is a unique continuation of the work request which is reflected in the dependent monitoring environment.

Note that the parent environment may continue to be active on behalf of the work request.

Note that specification of FUNCTION(CONTINUE) produces an inline expansion rather than an out-of-line service. Registers 0, 1, 14, and 15 are not preserved across the expansion.

### **FUNCTION=RETURN**

indicates that the work request is returning to a previously established parent monitoring environment.

Use of this option indicates that the dependent environment no longer represents the work request.

Note that specification of FUNCTION(RETURN) produces an inline expansion rather than an out-of-line service. Registers 0, 1, 14, and 15 are not preserved across the expansion.

### **,MONTKN=*montkn***

A required input parameter, which contains the delay monitoring token for the dependent environment.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32 bit field.

### **,PARENTENV=NOSWITCH**

### **,PARENTENV=SECONDARY**

A required parameter, which describes whether a space switch is needed to access the parent monitoring environment.

### **,PARENTENV=NOSWITCH**

indicates that NO space switch is needed to access the parent monitoring environment. This would be appropriate if the parent monitoring environment was established (by IWM4MCRE) to be used by routines in a specific system key or if it was established to be used in a specific user key in the current primary.

### **,PARENTENV=SECONDARY**

indicates that the parent monitoring environment was established in current secondary (for use by a specific user key).

### **,PARENTKEY=*parentkey***

When PARENTKEYP=VALUE is specified, a required input parameter, which contains the key in which the parent monitoring environment must be accessed.

Use of this keyword value requires that the invoker be in supervisor state or that the caller have PKM authority to the key specified. The high order 4 bits (i.e. bits 0-3) contain the key value.

Note that this is different from the "natural" way of declaring the key, and uses the machine orientation for keeping the storage key in the high order bits.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8 bit field.

**,PARENTKEYP=VALUE**

**,PARENTKEYP=PSWKEY**

**,PARENTKEYP=UNKNOWN**

A required parameter, which describes whether a key switch is needed to access the parent monitoring environment.

**,PARENTKEYP=VALUE**

indicates that the key is being passed explicitly via PARENTKEY.

**,PARENTKEYP=PSWKEY**

indicates that the current PSW key should be used. Use of this keyword value requires that the parent monitoring environment was established with the same key as the current PSW.

**,PARENTKEYP=UNKNOWN**

indicates that the key associated with the parent monitoring environment is unknown. Use of this keyword value requires that the invoker be in supervisor state or that the caller have PKM authority to key 0.

**,PLISTVER=IMPLIED\_VERSION**

**,PLISTVER=MAX**

**,PLISTVER=0**

**,PLISTVER=1**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form; in this way, MAX ensures that the parameter list does not overwrite nearby storage.

- **0**, which supports all parameters except those specifically referenced in higher versions.
- **1**, which supports both the following parameters and those from version 0:

WORKREQ\_STA

**To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX

## IWMMXFER Macro

- A decimal value of 0, or 1

### **,RETCODE=retcode**

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

### **,RSNCODE=rsncode**

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

### **,RUNTIME\_VER=SHORT\_FORM**

### **,RUNTIME\_VER=MINIMAL**

When FUNCTION=CONTINUE is specified, an optional parameter, which indicates what level of runtime verification will be performed. The default is RUNTIME\_VER=SHORT\_FORM.

### **,RUNTIME\_VER=SHORT\_FORM**

indicates that checking should verify that a monitoring environment is established and passes a short form of verification prior to being used.

### **,RUNTIME\_VER=MINIMAL**

indicates that checking will only be done to verify that a monitoring environment may be established, assuming that it would be valid and useable if established.

### **,RUNTIME\_VER=SHORT\_FORM**

### **,RUNTIME\_VER=MINIMAL**

When FUNCTION=RETURN is specified, an optional parameter, which indicates what level of runtime verification will be performed. The default is RUNTIME\_VER=SHORT\_FORM.

### **,RUNTIME\_VER=SHORT\_FORM**

indicates that checking should verify that a monitoring environment is established and passes a short form of verification prior to being used.

### **,RUNTIME\_VER=MINIMAL**

indicates that checking will only be done to verify that a monitoring environment may be established, assuming that it would be valid and useable if established.

### **,WORKREQ\_STA=workreq\_sta**

### **,WORKREQ\_STA=IWMEWLMARMSTATUSNONE**

When FUNCTION=RETURN is specified, an optional input parameter, which contains the completion status code of the work request. Available completion status codes (defined in macro IWMYCON) are: \* IwmEwlmArmStatusGood(0), \* IwmEwlmArmStatusAborted(1), \* IwmEwlmArmStatusFailed(2) or \* IwmEwlmArmStatusUnknown(3) The codes above correspond to status codes in the OpenGroup ARM 4.0 Standard (for the meaning of the status codes see the ARM 4.0 Standard at <http://www.opengroup.org/management/arm>). The default is IWMEWLMARMSTATUSNONE. indicates that internal information in the Monitoring Environment will be examined to determine the status of the work request: if no abnormal event was recorded for the monitoring environment via the IWMMABNL service, the completion status IwmEwlmArmStatusGood will be reported to EWLM. If an abnormal event was reported via IWMMABNL, the completion status IwmEwlmArmStatusFailed will be reported to EWLM.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

## ABEND Codes

None.

## Return and Reason Codes

When the IWMXFER macro returns control to your program:

- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, if you coded RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 47. Return and Reason Codes for the IWMXFER Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> <i>IwmRetCodeOk</i> <b>Meaning:</b> Successful completion. <b>Action:</b> None required.
4	—	<b>Equate Symbol:</b> <i>IwmRetCodeWarning</i> <b>Meaning:</b> Successful completion, unusual conditions noted.
4	xxxx0402	<b>Equate Symbol:</b> <i>IwmRsnCodeNoMonEnv</i> <b>Meaning:</b> Input monitoring token indicates no monitoring environment was established. <b>Action:</b> None required.
8	—	<b>Equate Symbol:</b> <i>IwmRetCodeInvocError</i> <b>Meaning:</b> Invalid invocation environment or parameters.
8	xxxx081F	<b>Equate Symbol:</b> <i>IwmRsnCodeNoRelate</i> <b>Meaning:</b> NO Parent environment exists since Relate Function(Continue) has not been performed or has not been performed subsequent to a Relate Function(Delete). <b>Action:</b> Check for possible storage overlay and whether Relate Function(Continue) has been used properly.
8	xxxx0820	<b>Equate Symbol:</b> <i>IwmRsnCodeBadMonEnv</i> <b>Meaning:</b> Monitoring environment does not pass verification.
8	xxxx0822	<i>IwmRsnCodeBadParEnv</i> : Parent monitoring environment does not pass verification. <b>Action:</b> Check for possible storage overlay.

## IWMMXFER Macro

Table 47. Return and Reason Codes for the IWMMXFER Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0895	<p><b>Equate Symbol:</b> IwmRsnCodeEWLMServNotEnabled</p> <p><b>Meaning:</b> Service is not enabled because monitoring environment can not be associated with EWLM work requests.</p> <p><b>Action:</b> Specify the parameter WORKREQ_STA only when the monitoring environment is created with IWMMCREA EWLM=YES or the address space is connected with IWMCNN EWLM=YES and the connect token is passed to IWMMCREA when creating the monitoring environment.</p>
8	xxxx0897	<p><b>Equate Symbol:</b> IwmRsnCodeTranStatusInvalid</p> <p><b>Meaning:</b> Passed work request completion status is not valid.</p> <p><b>Action:</b> Check the EWLM ARM interface specification for valid completion status values.</p>

---

## Chapter 41. IWMMXRA – Exit For Adjusting Resources

IWMMXRA invokes the resource adjustment exit. This exit makes the adjustments indicated for the buffer pool or other resources which may be responsible for delays to work requests. The return or reason codes for IWMMXRA are those set by the exit invoked.

The exit environment is described in the following. The parameter list is in pageable storage addressable in the current address space, but is not guaranteed to be in the key of the exit because it is not expected to be changed by the exit. Upon entry to the exit, the register contents are as follows:

- Register 0 = not defined
- Register 1 contains the address of a parameter list as formatted by the list form of this macro, IWMMXRA MF=(L).
- Registers 2-13 = not defined
- Register 14 = return address
- Register 15 = entry-point address

Upon entry to the exit, the access register contents are undefined.

Upon return from the exit, the register contents are as follows:

- Register 0 = Reason code if GR15 return code is non-zero
- Registers 1-14 = not defined (need not be restored to value on entry)
- Register 15 = Return code

Upon return from the exit, the access register contents are unchanged.

---

### Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Supervisor state with the same PSW key as at the time of registration (IWM4MREG).
<b>Dispatchable unit mode:</b>	Task or SRB
<b>Cross-memory mode:</b>	PASN=HASN=SASN. The current home address space must be the same as at the time of registration (IWM4MREG).
<b>AMODE:</b>	31-bit
<b>ASC mode:</b>	Primary
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks may be held. FRRs may be established.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

---

### Programming Requirements

The list form of the macro is intended for use by products supplying a resource adjustment exit.

The execute form of the macro is intended for use by z/OS.

The assembler execute form only initializes the parameter list and calls the exit routine. The following restrictions apply:

- The invoker must save registers required before invoking the macro

## IWMMXRA Macro

- The invoker must restore registers required immediately after invoking the macro, without depending on the exit to preserve any registers.

---

## Restrictions

None.

---

## Input Register Information

Before issuing the IWMMXRA macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

---

## Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Reason code if GR15 return code is non-zero
1	Used as work register by the system
2-13	Unpredictable in assembler form, unchanged in PL/X form
14	Used as work registers by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

---

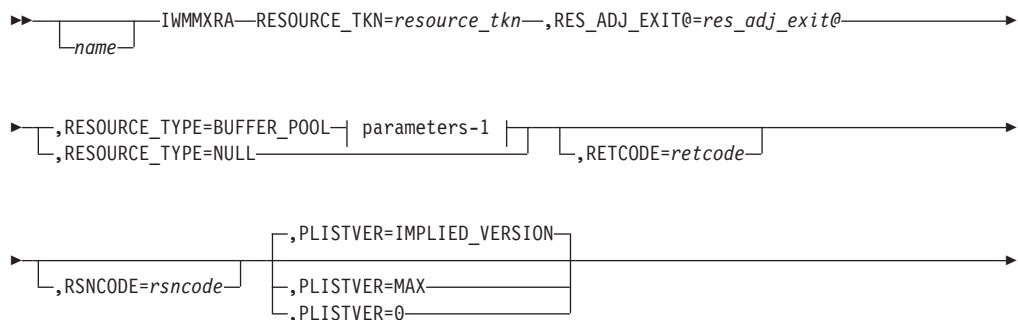
## Performance Implications

None.

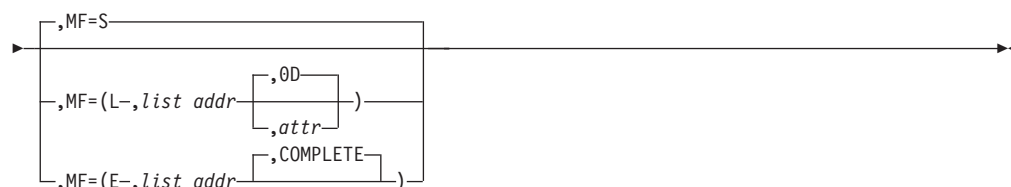
---

## Syntax

### main diagram







### parameters-1

```

  >>—,OWNER_TKN=owner_tkn—,RES_CUR_SIZE=res_cur_size—,RES_NEW_SIZE=res_new_size—
  
```

## Parameters

The parameters are explained as follows:

### *name*

An optional symbol, starting in column 1, that is the name on the IWMXRA macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

### **,MF=S**

**,MF=(L,*list addr*)**

**,MF=(L,*list addr*,*attr*)**

**,MF=(L,*list addr*,**OD**)**

**,MF=(E,*list addr*)**

**,MF=(E,*list addr*,**COMPLETE**)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

### *,list addr*

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

### *,attr*

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

### **,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

## IWMMXRA Macro

| **,OWNER\_TKN=owner\_tkn**

| When RESOURCE\_TYPE=BUFFER\_POOL is specified, a required input  
| parameter, which contains data associated with the resource that was passed to  
| registration (IWM4MREG).

| **To code:** Specify the RS-type address, or address in register (2)-(12), of an  
| 8-character field.

| **,PLISTVER=IMPLIED\_VERSION**

| **,PLISTVER=MAX**

| **,PLISTVER=0**

| An optional input parameter that specifies the version of the macro. PLISTVER  
| determines which parameter list the system generates. PLISTVER is an  
| optional input parameter on all forms of the macro, including the list form. When  
| using PLISTVER, specify it on all macro forms used for a request and with the  
| same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

| If you can tolerate the size change, IBM recommends that you always specify  
| PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that  
| the list-form parameter list is always long enough to hold all the parameters  
| you might specify on the execute form, when both are assembled with the  
| same level of the system. In this way, MAX ensures that the parameter list  
| does not overwrite nearby storage.

- **0**, if you use the currently available parameters.

| **To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX
- A decimal value of 0

| **,RES\_ADJ\_EXIT@=res\_adj\_exit@**

| A required input parameter that contains the address of the resource adjustment  
| exit to be invoked.

| **To code:** Specify the RS-type address, or address in register (2)-(12), of a  
| pointer field.

| **,RES\_CUR\_SIZE=res\_cur\_size**

| When RESOURCE\_TYPE=BUFFER\_POOL is specified, a required input  
| parameter, which contains the current size (in 4K pages) associated with the  
| specified resource.

| **To code:** Specify the RS-type address, or address in register (2)-(12), of a  
| fullword field.

| **,RES\_NEW\_SIZE=res\_new\_size**

| When RESOURCE\_TYPE=BUFFER\_POOL is specified, a required input  
| parameter, which contains the new size (in 4K pages) to be associated with the  
| specified resource.

| **To code:** Specify the RS-type address, or address in register (2)-(12), of a  
| fullword field.

**RESOURCE\_TKN=resource\_tkn**

A required input parameter, which contains the associated WLM resource token which is returned by the registration (IWM4MREG).

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

**,RESOURCE\_TYPE=BUFFER\_POOL****,RESOURCE\_TYPE=NULL**

A required parameter, which indicates the type of resource being registered.

**,RESOURCE\_TYPE=BUFFER\_POOL**

indicates that buffer pool adjustments are needed.

**,RESOURCE\_TYPE=NULL**

indicates that no exit is to be called.

**,RETCODE=retcode**

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,RSNCODE=rsncode**

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

---

## ABEND Codes

None.

---

## Return and Reason Codes

When the IWMXRA macro returns control to your program:

- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, if you coded RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code.

Table 48. Return and Reason Codes for the IWMXRA Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> IwmRetCodeOk <b>Meaning:</b> Successful completion. <b>Action:</b> None required.

---

## Example

To call a resource adjustment exit, specify:

```

IWMXRA                                     X
RES_ADJ_EXIT@=ADJEXIT@,                   X
OWNER_TKN=OWNERTKN,                        X
RESOURCE_TYPE=BUFFER_POOL,                 X
RESOURCE_TKN=RSCTOKEN,                     X
RES_CUR_SIZE=RESCURSIZE,                   X
RES_NEW_SIZE=RESNEWSIZE,                   X

```

## IWMMXRA Macro

```
|
|
|                               RETCODE=RC,RSNCODE=RSN
|
|
| *
| * Storage areas
| *
| RESCURSIZE DS FL4           contains the current size of the
| *                               specified resource in 4K pages
| RESNEWSIZE DS FL4           contains the new target size of the
| *                               specified resource in 4K pages
| ADJEXIT@   DS AL4           contains the address of the
| *                               Resource Adjustment Exit
| *                               to be invoked
| OWNERTKN   DS CL8           Contains data maintained by
| *                               the user
| RSCTOKEN   DS CL8           WLM resource token
| RC         DS F             Return code
| RSN        DS F             Reason code
|
```

---

## Chapter 42. IWMPACT – Activate Service Policy

The Activate Service Policy routine is given control from the IWMPACT macro. The Activate Service Policy macro will complete the parameter list with caller-provided data and generate a stacking, program call to the activate policy service.

The purpose of this routine is to activate a service policy in the sysplex. The name of the service policy to be activated must be provided as input. The specified policy must exist in the current WLM service definition installed on the WLM couple data set.

The Activate Service Policy service causes a service policy to be activated synchronously. In other words, control will not be returned to the caller until the policy has been activated on all systems in the sysplex or for some reason the policy activation could not be completed.

Note that only a single policy activation request can be processed at any one time. Therefore, if a previous policy activation request is being processed and a new activation request is issued, the new request will be rejected with an appropriate return and reason code. This will occur regardless of whether the two requests were issued on the same system or different systems in the sysplex. The user can optionally request that the name of the system where another policy activation is taking place be returned in the variable specified in keyword `SYSTEM_NAME`.

The Activate Service Policy macro is provided in list, execute, and standard form. The list form accepts no variable parameters and is used only to reserve space for the activate policy parameter list. The standard form is provided for use with routines which do not require reentrant code. The execute form is provided for use with the list format for reentrant routines.

---

### Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Problem state. Any PSW key
<b>Dispatchable unit mode:</b>	Task
<b>Cross memory mode:</b>	PASN=HASN=SASN
<b>AMODE:</b>	31-bit
<b>ASC mode:</b>	Primary
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks may be held.
<b>Control parameters:</b>	All parameter areas must reside in current primary.

---

### Programming Requirements

1. The macro `CVT` must be included to use this macro.
2. This functions provided by this macro are only supported on MVS/ESA SP5.2.0 or later. If code with this macro expansion is executed on lower MVS/ESA releases (version 3 or later) a return code of 4 and reason code of xxxx041A (constant `lwmRsnCodeNoPolMgt`) will be returned indicating that the service is not available. This macro should NOT be executed on MVS versions 1 or 2.
3. The macro `IWMYCON` must be included to use this macro.

## IWMPACT Macro

4. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
5. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, IWMSRNCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.

---

## Restrictions

The caller cannot have an EUT FRR established.

---

## Input Register Information

Before issuing the IWMPACT macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

---

## Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Reason code if GR15 return code is non-zero
1	Used as work registers by the system
2-13	Unchanged
14	Used as a work register by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

---

## Performance Implications

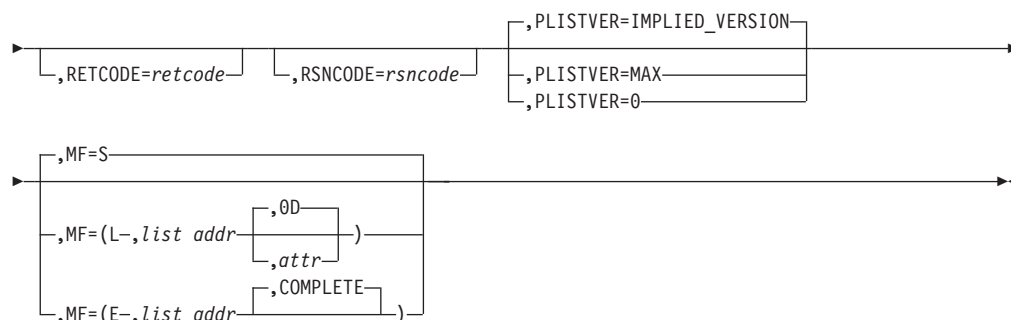
None.

---

## Syntax

### main diagram

```
➔ ┌───┴───┐ IWMPACT ─ POLICY_NAME=policy_name ─┬───┴───┐ ➔  
   └───┬───┘ └───┬───┘  
      └───┬───┘ └───┬───┘  
      name     ,SYSTEM_NAME=system_name
```



## Parameters

The parameters are explained as follows:

### *name*

An optional symbol, starting in column 1, that is the name on the IWMPACT macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

### **,MF=S**

**,MF=(L,*list addr*)**

**,MF=(L,*list addr*,*attr*)**

**,MF=(L,*list addr*,**0D**)**

**,MF=(E,*list addr*)**

**,MF=(E,*list addr*,**COMPLETE**)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

### *,list addr*

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

### *,attr*

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

### **,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

### **,PLISTVER=IMPLIED\_VERSION**

### **,PLISTVER=MAX**

### **,PLISTVER=0**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form; in this way, MAX ensures that the parameter list does not overwrite nearby storage.

- **0**, if you use the currently available parameters.

**To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX
- A decimal value of 0

### **POLICY\_NAME=*policy\_name***

A required input parameter, variable specifying the name of the service policy to be activated. The specified service policy must exist in the current service definition that is installed on the WLM couple data set.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

### **,RETCODE=*retcode***

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

### **,RSNCODE=*rsncode***

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

### **,SYSTEM\_NAME=*system\_name***

An optional output parameter, variable where the name of the system where another policy activation is taking place will be returned. This variable is only filled in when a return code of 4 and a reason code of xxxx0415 is returned.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

---

## ABEND Codes

### Reason Code (Hex)

#### Explanation

**0Axx0005** An attempt to reference caller's parameters caused an OC4 abend.



## Return and Reason Codes

When the IWMPACT macro returns control to your program:

- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, if you coded RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 49. Return and Reason Codes for the IWMPACT Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> <code>IwmRetCodeOk</code> <b>Meaning:</b> Successful completion. <b>Action:</b> None required.
4	—	<b>Equate Symbol:</b> <code>IwmRetCodeWarning</code> <b>Meaning:</b> Successful completion, unusual conditions noted.
4	xxxx0414	<b>Equate Symbol:</b> <code>IwmRsnCodeNullCDS</code> <b>Meaning:</b> This request could not be completed because no Service definition has been installed on the WLM CDS. <b>Action:</b> None required.
4	xxxx0415	<b>Equate Symbol:</b> <code>IwmRsnCodePolicyActInProgress</code> <b>Meaning:</b> This request could not be completed because another policy activation request is currently being processed. If specified, the <code>SYSTEM_NAME</code> parameter will contain the name of the system on which policy activation is in progress. <b>Action:</b> None required. If this service is re-invoked at a later time it may be successful.
4	xxxx0416	<b>Equate Symbol:</b> <code>IwmRsnCodePolicyUndefined</code> <b>Meaning:</b> The service policy specified could not be found in the service definition currently installed on the WLM couple data set. The service policy was not activated. <b>Action:</b> None required. Verify that the policy was specified correctly.
4	xxxx041A	<b>Equate Symbol:</b> <code>IwmRsnCodeNoPolMgt</code> <b>Meaning:</b> System does not support policy management services. This return code is only set when the MVS release is prior to MVS/ESA SP5.2.0. <b>Action:</b> None required.
8	—	<b>Equate Symbol:</b> <code>IwmRetCodeInvocError</code> <b>Meaning:</b> Invalid invocation environment or parameters.
8	xxxx0801	<b>Equate Symbol:</b> <code>IwmRsnCodeSrbMode</code> <b>Meaning:</b> Caller is in SRB mode. <b>Action:</b> Avoid requesting this function while in SRB mode.

Table 49. Return and Reason Codes for the IWMPACT Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0803	<b>Equate Symbol:</b> lwmRsnCodeDisabled <b>Meaning:</b> Caller is disabled. <b>Action:</b> Avoid requesting this function while disabled.
8	xxxx0804	<b>Equate Symbol:</b> lwmRsnCodeLocked <b>Meaning:</b> Caller is locked. <b>Action:</b> Avoid requesting this function while locked.
8	xxxx0810	<b>Equate Symbol:</b> lwmRsnCodeEutFrr <b>Meaning:</b> Caller has EUT FRR established. <b>Action:</b> Avoid requesting this function with an EUT FRR set.
8	xxxx0823	<b>Equate Symbol:</b> lwmRsnCodeDatoff <b>Meaning:</b> Caller invoked service while DATOFF <b>Action:</b> Avoid requesting this function in this environment.
8	xxxx0824	<b>Equate Symbol:</b> lwmRsnCodeAmode24 <b>Meaning:</b> Caller invoked service but was in 24 bit addressing mode. <b>Action:</b> Request this function only when you are in 31 bit addressing mode.
8	xxxx0827	<b>Equate Symbol:</b> lwmRsnCodeRsvdNot0 <b>Meaning:</b> Reserved field in parameter list was non-zero. <b>Action:</b> Check for possible storage overlay of the parameter list.
8	xxxx0828	<b>Equate Symbol:</b> lwmRsnCodeBadVersion <b>Meaning:</b> Version number in parameter list is not valid or the length specified is incorrect. <b>Action:</b> Check for possible storage overlay of the parameter list.
8	xxxx0841	<b>Equate Symbol:</b> lwmRsnCodeXMemMode <b>Meaning:</b> Caller is in cross memory mode. <b>Action:</b> Invoke the function in non-cross memory mode.
C	—	<b>Equate Symbol:</b> lwmRetCodeEnvError <b>Meaning:</b> Environmental error.
C	xxxx0C01	<b>Equate Symbol:</b> lwmRsnCodeNoStg <b>Meaning:</b> Storage is not available for the request. <b>Action:</b> There is a storage shortage. The function may work successfully at a later time.

Table 49. Return and Reason Codes for the IWMPACT Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
C	xxxx0C0E	<p><b>Equate Symbol:</b> lwmRsnCodeInsufAccess</p> <p><b>Meaning:</b> The RACF facility class is active and a profile has been defined for the MVSADMIN.WLM.POLICY RACF facility class profile to which the caller does not have sufficient (update) access.</p> <p><b>Action:</b> Invoke the function when the conditions are alleviated.</p>
C	xxxx0C0F	<p><b>Equate Symbol:</b> lwmRsnCodeCDSNotAvail</p> <p><b>Meaning:</b> A couple data set for WLM has not been defined or it has been defined but this system does not have connectivity to the data set.</p> <p><b>Action:</b> No action required.</p>
C	xxxx0C10	<p><b>Equate Symbol:</b> lwmRsnCodeCDSTooSmall</p> <p><b>Meaning:</b> WLM CDS is too small to process the request.</p> <p><b>Action:</b> No action required.</p>
C	xxxx0C11	<p><b>Equate Symbol:</b> lwmRsnCodeOneSystemUnable</p> <p><b>Meaning:</b> At least one system in the sysplex was unable to activate the policy. One or more systems may have been successful in activating the policy.</p> <p><b>Action:</b> No action required.</p>
C	xxxx0C13	<p><b>Equate Symbol:</b> lwmRsnCodePolicyNotAvail</p> <p><b>Meaning:</b> The service policy specified could not be verified because the service definition retrieved from WLM CDS has failed validation. The service policy was not activated.</p> <p><b>Action:</b> No action required.</p>
10	—	<p><b>Equate Symbol:</b> lwmRetCodeCompError</p> <p><b>Meaning:</b> Component error.</p> <p><b>Action:</b> No action required. The function may be successful if invoked again.</p>



---

## Chapter 43. IWMPQRY – Query Active Service Policy

The query active service policy routine is given control from the IWMPQRY macro. The query active service policy macro will complete the parameter list with callerprovided data and generate a stacking, space switching, program call to the query service.

The purpose of this routine is to return a representation of the active policy which could be used to explain how the system/sysplex is being managed and could be used in conjunction with current measurements to evaluate the condition of the system/sysplex. The information returned is not serialized upon return to the caller, and so may be out-of-date due to a change in policy.

The Query Active Service Policy macro is provided in list, execute, and standard form. The list form accepts no variable parameters and is used only to reserve space for the query parameter list. The standard form is provided for use with routines which do not require reentrant code. The execute form is provided for use with the list format for reentrant routines.

The parameter list must be in the caller's primary address space or be addressable by the dispatchable unit access list.

---

### Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Supervisor state or program key mask (PKM) allowing keys 0-7.
<b>Dispatchable unit mode:</b>	Task
<b>Cross memory mode:</b>	Any PASN, any HASN, any SASN
<b>AMODE:</b>	31-bit
<b>ASC mode:</b>	Primary or access register (AR) If in Access Register ASC mode, specify SYSSTATE ASCENV=AR before invoking this macro.
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks may be held.
<b>Control parameters:</b>	All parameter areas must reside in current primary or be addressable by the dispatchable unit access list.

---

### Programming Requirements

1. The macro CVT must be included to use this macro.
2. This macro may only be used on MVS/ESA SP5.1.0 or later.
3. The macro IWMYCON must be included to use this macro.
4. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
5. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, IWMRNCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.

## Restrictions

The caller cannot have an EUT FRR established.

## Input Register Information

Before issuing the IWMPQRY macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

## Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Reason code if GR15 return code is non-zero
1	Used as work registers by the system
2-13	Unchanged
14	Used as a work register by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

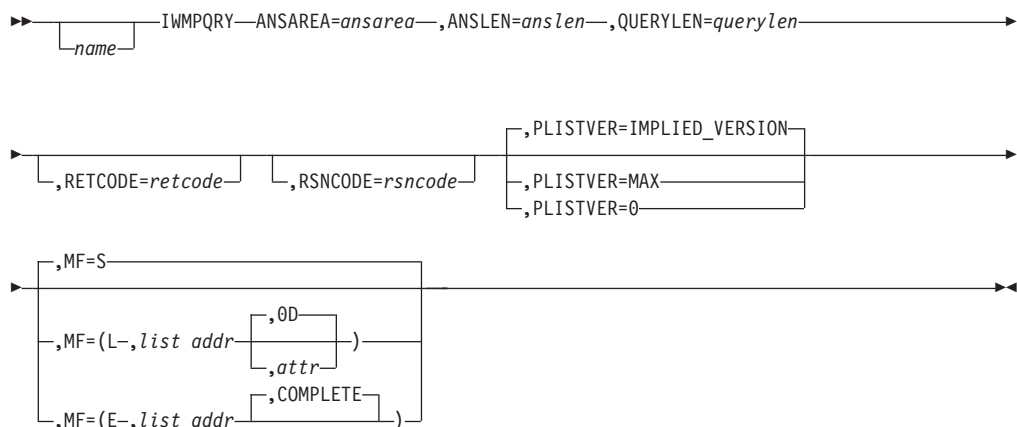
Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

## Performance Implications

None.

## Syntax

### main diagram



## Parameters

The parameters are explained as follows:

### *name*

An optional symbol, starting in column 1, that is the name on the IWMPQRY macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

### **ANSAREA=ansarea**

A required output parameter, variable specifying an area to contain the data being returned by IWMPQRY. The answer area is defined by the IWMSVPOL macro.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a character field.

### **,ANSLEN=anslen**

A required input parameter, variable which contains the length of the area provided to contain the data being returned by IWMPQRY.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

### **,MF=S**

### **,MF=(L,list addr)**

### **,MF=(L,list addr,attr)**

### **,MF=(L,list addr,0D)**

### **,MF=(E,list addr)**

### **,MF=(E,list addr,COMPLETE)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

### *,list addr*

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

### *,attr*

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

### **,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

## IWMPQRY Macro

### **,PLISTVER=IMPLIED\_VERSION**

### **,PLISTVER=MAX**

### **,PLISTVER=0**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form; in this way, MAX ensures that the parameter list does not overwrite nearby storage.

- **0**, if you use the currently available parameters.

**To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX
- A decimal value of 0

### **,QUERYLEN=querylen**

A required output parameter, variable which contains the number of bytes needed to contain the policy information.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

### **,RETCODE=retcode**

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

### **,RSNCODE=rsncode**

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

---

## ABEND Codes

### Reason Code (Hex)

#### Explanation

**0Axx0005** An attempt to reference caller's parameters caused an OC4 abend.

---

## Return and Reason Codes

When the IWMPQRY macro returns control to your program:

- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, if you coded RSNCODE) contains reason code.



The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 50. Return and Reason Codes for the IWMPQRY Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> lwmRetCodeOk <b>Meaning:</b> Successful completion. <b>Action:</b> None required.
4	—	<b>Equate Symbol:</b> lwmRetCodeWarning <b>Meaning:</b> Successful completion, unusual conditions noted.
4	xxxx0401	<b>Equate Symbol:</b> lwmRsnCodeNoWLM <b>Meaning:</b> System does not support policy management services. This return code is only set when the MVS release is prior to MVS/ESA SP5.1.0. <b>Action:</b> None required.
4	xxxx040A	<b>Equate Symbol:</b> lwmRsnCodeOutputAreaTooSmall <b>Meaning:</b> The output area supplied is too small to receive all the available information. <b>Action:</b> None required. If necessary, reinvoke the service with an output area of sufficient size to receive all information.
4	xxxx0423	<b>Equate Symbol:</b> lwmRsnCodeDefaultPolicy <b>Meaning:</b> The default policy was returned. <b>Action:</b> None required.
8	—	<b>Equate Symbol:</b> lwmRetCodeInvocError <b>Meaning:</b> Invalid invocation environment or parameters.
8	xxxx0801	<b>Equate Symbol:</b> lwmRsnCodeSrbMode <b>Meaning:</b> Caller is in SRB mode. <b>Action:</b> Avoid requesting this function while in SRB mode.
8	xxxx0803	<b>Equate Symbol:</b> lwmRsnCodeDisabled <b>Meaning:</b> Caller is disabled. <b>Action:</b> Avoid requesting this function while disabled.
8	xxxx0804	<b>Equate Symbol:</b> lwmRsnCodeLocked <b>Meaning:</b> Caller is locked. <b>Action:</b> Avoid requesting this function while locked.
8	xxxx0810	<b>Equate Symbol:</b> lwmRsnCodeEutFrr <b>Meaning:</b> Caller has EUT FRR established. <b>Action:</b> Avoid requesting this function with an EUT FRR set.
8	xxxx0830	<b>Equate Symbol:</b> lwmRsnCodeBadAlet <b>Meaning:</b> Caller has passed an invalid ALET. <b>Action:</b> Check for possible storage overlay of the parameter list or variable.

## IWMPQRY Macro

---

## Chapter 44. IWMRCOLL – Collecting Workload Activity Data

With the IWMRCOLL macro, a performance monitor can get the following Workload activity information:

- Resource consumption information
- Response time and distribution information
- General delay information
- Subsystem work manager delay state information

For a detailed description, refer to Chapter 9, “Using the Workload Reporting Services,” on page 91.

To help the caller keep track of changes in workload management, this service returns a token, ANSTOKN. ANSTOKN is a required input on all subsequent calls to IWMRCOLL. When a change occurs in workload management, such as when a new policy is activated, IWMRCOLL returns a new token value. The caller’s code should check the reason codes to see if the ANSTOKN has changed since the last call to IWMRCOLL. If the token has changed, the performance monitor should reset its reporting interval. If the token has not changed, the performance monitor can continue with its existing reporting interval.

There are also some ENF event codes to keep track of changes in workload management. For information about the ENF codes, see *z/OS MVS Programming: Authorized Assembler Services Reference EDT-IXG*.

The caller must provide a storage area in the ANSAREA=ansarea and the length of that area in the ANSLEN=anslen for IWMRCOLL to place the workload activity information. This area may reside in either address space related storage or dataspace storage. IWMRCOLL returns the information, which is mapped by IWMWRCAA.

You must also specify the MINLEN and MAXLEN parameters. IWMRCOLL fills in the minimum and maximum amount of storage required for the answer area.

The caller should issue the IWMPQRY macro for active service policy information to map the workload activity information.

If the caller does not provide enough storage to contain all of the workload activity data, no data is returned. IWMRCOLL returns the minimum length of the storage required in the ANSLEN field, and issues the appropriate return and reason code.

---

### Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Supervisor state or program key mask (PKM) allowing keys 0-7
<b>Dispatchable unit mode:</b>	Task
<b>Cross memory mode:</b>	Any PASN, any HASN, any SASN
<b>AMODE:</b>	31-bit
<b>ASC mode:</b>	Primary or access register. If in access register ASC mode, specify SYSSTATE ASCENV = AR before invoking IWMRCOLL.

## IWMRCOLL Macro

<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks held.
<b>Control parameters:</b>	Control parameters must be in the primary address space.  The caller of IWMRCOLL must provide storage for an answer area mapped by IWMWRCOA. This answer area may reside in the caller's primary address space, or in a dataspace accessible via the current unit of work's dispatchable unit access list (DUaI).

---

## Programming Requirements

You must include the CVT and the IWMYCON mapping macros in the program.

---

## Restrictions

The caller cannot have an EUT FRR established.

---

## Input Register Information

Before issuing the IWMRCOLL macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, using it as a base register, or using it to provide the ALET of the storage area.

---

## Output Register Information

When control returns to the caller, the GPRs contain:

<b>Register</b>	<b>Contents</b>
<b>0</b>	Reason code if the GP 15 return code is non-zero.
<b>1</b>	Used as work registers by the system.
<b>2 - 13</b>	Unchanged
<b>14</b>	Used as a work register by the system.
<b>15</b>	Return code

When control returns to the caller, the ARs contain:

<b>Register</b>	<b>Contents</b>
<b>0 - 1</b>	Used as work registers by the system.
<b>2 - 13</b>	Unchanged
<b>14 - 15</b>	Used as work registers by the system.

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

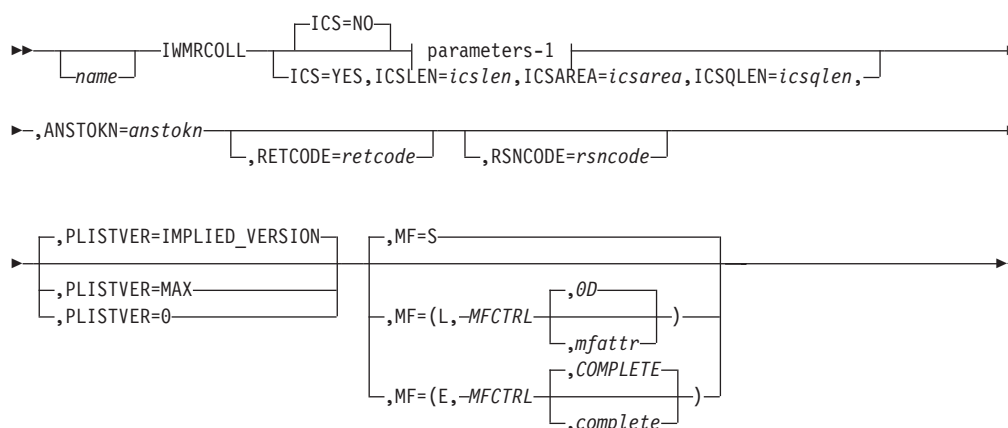
---

## Performance Implications

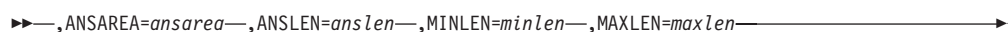
None.

## Syntax

### main diagram



### parameters-1



## Parameters

The parameters are explained as follows:

### *name*

An optional symbol, starting in column 1, that is the name on the IWMRCOLL macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

### **,ANSAREA=ansarea**

When ICS=NO is a required output parameter that contains the address of a storage area to hold the information returned by IWMRCOLL. The area is mapped by the IWMWRCAA mapping macro.

**To code:** Specify the RS-type address, or address in register (2)-(12) of a character field.

### **,ANSLEN=anslen**

When ICS=NO is specified, a required input parameter that contains the length of the storage area (answer area) you are providing on ANSAREA.

**To code:** Specify the RS-type address, or address in register (2)-(12) of a fullword field.

### **,ANSTOKN=anstoken,**

A required input/output parameter that contains a token value. On your first call to IWMRCOLL, you specify ANSTOKEN as an output parameter. IWMRCOLL provides a token value that is required for subsequent calls to IWMRCOLL.

**To code:** Specify the RS-type address, or address in register (2)-(12) of an 8-character field.

### **ICS=NO**

## IWMRCOLL Macro

### **ICS=YES**

An optional parameter that specifies whether IWMRCOLL should return ICS information, or workload activity information. The default is ICS=NO.

### **ICS=NO**

Specifies that IWMRCOLL should return workload activity information.

### **ICS=YES**

Specifies that IWMRCOLL should return ICS information. It is valid for systems prior to z/OS R3 only.

### **,ICSAREA=icsarea**

When ICS=YES is specified, a required output parameter that is valid for systems prior to z/OS R3 only.

**To code:** Specify the RS-type address, or address in register (2)-(12) of a character field.

### **,ICSLLEN=icslen**

When ICS=YES is specified, a required input parameter that is valid for systems prior to z/OS R3 only.

**To code:** Specify the RS-type address, or address in register (2)-(12) of a fullword field.

### **,ICSQLEN=icsqlen**

When ICS=YES is specified, a required output parameter that is valid for systems prior to z/OS R3 only.

**To code:** Specify the RS-type address, or address in register (2)-(12) of a fullword field.

### **,MAXLEN=maxlen**

When ICS=NO is specified, a required output parameter that contains the maximum length of the storage area required by IWMRCOLL to contain all the performance data for transactions that might run while the ANSTOKN is valid.

**To code:** Specify the RS-type address, or address in register (2)-(12) of a fullword field.

### **,MF=S**

**,MF=(L,list addr)**

**,MF=(L,list addr, attr)**

**,MF=(L,list addr)OD**

**,MF=(E,list addr)**

**,MF=(E,list addr,COMPLETE)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

**,list addr**

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

**,attr**

An optional 1 to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of X'0F' to force the parameter list to a word boundary, or X'0D' to force the parameter list to a doubleword boundary. If you do not code *,attr*, the system provides a value of X'0D'.

**,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

**,MINLEN=*minlen***

When ICS=NO is specified, a required output parameter that contains the minimum length of the storage area required by IWMRCOLL to contain all existing performance data.

**To code:** Specify the RS-type address, or address in register (2)-(12) of a fullword field.

**,PLISTVER=IMPLIED\_VERSION****,PLISTVER=MAX****,PLISTVER=0**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

**IMPLIED\_VERSION**

Is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.

**MAX**

Specify PLISTVER=MAX if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form, when both are assembled with the same level of the system. In this way, MAX ensures that the parameter list does not overwrite nearby storage.

**0** Specify PLISTVER=0 to use the currently available parameters.

**To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX
- A decimal value of 0

**,RETCODE=*retcode***

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

## IWMRCOLL Macro

**,RSNCODE=rsncode**

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

---

## ABEND Codes

None.

---

## Return and Reason Codes

When the IWMRCOLL macro returns control to your program:

- GPR 15 (and *retcode*, when you code RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, when you code RSNCODE) contains a reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 51. Return and Reason Codes for the IWMRCOLL Macro

Return Code	Reason Code	Equate Symbol, Meaning and Action
0	-	<b>Equate Symbol:</b> IwmRetCodeOk <b>Meaning:</b> Successful completion. All requested data returned. <b>Action:</b> None required.
4	-	<b>Equate Symbol:</b> IwmRetCodeWarning <b>Meaning:</b> Successful completion, unusual conditions noted.
4	xxxx0401	<b>Equate Symbol:</b> IwmRsnCodeNoWLM <b>Meaning:</b> The system does not support workload activity data collection. The collect output area has not been modified. This return code is set only when the MVS release is prior to MVS/ESA SP5.1.0. <b>Action:</b> None required.
4	xxxx040A	<b>Equate Symbol:</b> IwmRsnCodeOutputAreaTooSmall <b>Meaning:</b> The output area supplied is too small to receive all the available information. The correct answer area length is returned in the MINLEN and MAXLEN fields. <b>Action:</b> None required. If necessary, reinvoke the service with an output area of sufficient size to receive all information.
4	xxxx040D	<b>Equate Symbol:</b> IwmRsnCodeIcsDefault <b>Meaning:</b> The system default ICS is in effect. No ICS information is returned. <b>Action:</b> None required.



Table 51. Return and Reason Codes for the IWMRCOLL Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning and Action
4	xxxx040E	<p><b>Equate Symbol:</b> IwmRsnCodeIcsAreaTooSmall</p> <p><b>Meaning:</b> The output area supplied is too small to receive all the available ICS information. The correct answer area length is returned in the ICSQLEN field.</p> <p><b>Action:</b> None required. If necessary, reinvoke the service with an output area of sufficient size to receive all information.</p>
4	xxxx040F	<p><b>Equate Symbol:</b> IwmRsnCodeStateInvDatRet</p> <p><b>Meaning:</b> The token value specified on the ANSTOKN keyword is associated with a WLM state that is no longer valid. The new system state is represented by the token returned in the ANSTOKN field. The answer area provided is large enough to contain the available data. However, the new answer area lengths are returned in the MINLEN and MAXLEN fields, or in the ICSQLEN field.</p> <p><b>Action:</b> Reinvoke the service with the token passed with the ANSTOKN keyword.</p>
8	-	<p><b>Equate Symbol:</b> IwmRetCodeInvocError</p> <p><b>Meaning:</b> Invalid invocation environment or parameters.</p>
8	xxxx0801	<p><b>Equate Symbol:</b> IwmRsnCodeSrbMode</p> <p><b>Meaning:</b> Caller is in SRB mode.</p> <p><b>Action:</b> Avoid requesting this function while in SRB mode.</p>
8	xxxx0803	<p><b>Equate Symbol:</b> IwmRsnCodeDisabled</p> <p><b>Meaning:</b> Caller is disabled.</p> <p><b>Action:</b> Avoid requesting this function while disabled.</p>
8	xxxx0804	<p><b>Equate Symbol:</b> IwmRsnCodeLocked</p> <p><b>Meaning:</b> Caller is locked.</p> <p><b>Action:</b> Avoid requesting this function while locked.</p>
8	xxxx0810	<p><b>Equate Symbol:</b> IwmRsnCodeEutFrr</p> <p><b>Meaning:</b> Caller has an EUT FRR set.</p> <p><b>Action:</b> Avoid requesting this function with an EUT FRR set.</p>
8	xxxx0830	<p><b>Equate Symbol:</b> IwmRsnCodeBadAlet</p> <p><b>Meaning:</b> Caller specified an invalid alet for the storage pointed to by the ANSAREA keyword.</p> <p><b>Action:</b> Check for possible storage overlay of the parameter list or variable.</p>

## IWMRCOLL Macro

Table 51. Return and Reason Codes for the IWMRCOLL Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning and Action
8	xxxx0832	<p><b>Equate Symbol:</b> IwmRsnCodeStateInvNoDatRet</p> <p><b>Meaning:</b> The token value specified on the ANSTOKN keyword is associated with a WLM state that is no longer valid. A new token has been returned. The storage provided is not large enough to contain all of the data available because of the state change. No data was returned. The length of the new answer area required is returned in the MINLEN and MAXLEN fields, or the ICSQLEN field.</p> <p><b>Action:</b> Reinvoke the service with an output area of sufficient size to receive all information and the token passed with the ANSTOKN keyword.</p>
8	xxxx0833	<p><b>Equate Symbol:</b> IwmRsnCodeNotInCompatMode</p> <p><b>Meaning:</b> ICS information cannot be returned because WLM is not running in compatibility mode.</p> <p><b>Action:</b> Reinvoke the service by using ICS=NO.</p>
8	xxxx0835	<p><b>Equate Symbol:</b> IwmRsnCodeBadIcsAlet</p> <p><b>Meaning:</b> Caller specified an invalid alet for the storage pointed to by the ICSAREA keyword.</p> <p><b>Action:</b> Check for possible storage overlay of the parameter list or variable.</p>
C	-	<p><b>Equate Symbol:</b> IwmRetCodeEnvError</p> <p><b>Meaning:</b> Environmental error</p>
C	xxxx0C01	<p><b>Equate Symbol:</b> IwmRsnCodeNoStg</p> <p><b>Meaning:</b> Storage is not available for the request.</p> <p><b>Action:</b> There is a storage shortage. The function may work successfully at a later time.</p>
C	xxxx0C0A	<p><b>Equate Symbol:</b> IwmRsnCodeSuspended</p> <p><b>Meaning:</b> Data collection is suspended as a result of a component error. No data can be returned for this IWMRCOLL invocation, future invocations may be successful.</p> <p><b>Action:</b> Reinvoke this service.</p>
10	-	<p><b>Equate Symbol:</b> IwmRetCodeCompError</p> <p><b>Meaning:</b> Component error</p> <p><b>Action:</b> No action required. The function may work successfully if invoked again.</p>

## Example

For workload activity information from a system running in goal mode, specify:

```
IWMRCOLL ICS=NO,ANSAREA=(R6),ANSLEN=(R8),
          MINLEN=QMINLEN,MAXLEN=QMAXLEN,
          RSNCODE=RSN,MF=(E,MFRCOLL)
```

---

## Chapter 45. IWMRESET – Change a Job

The IWMRESET macro allows the caller to perform the same functions as the RESET system command. If the system is running in workload management goal mode mode, the caller can:

- Change the service class of work currently in execution, with the SRVCLASS keyword. Resetting to a new service class also resumes quiesced work.
- Quiesce work currently in execution, with the QUIESCE keyword.
- Reclassify work currently in execution according to the service policy in effect, with the RESUME keyword. The RESUME keyword also resumes quiesced work.

The system does not allow every address space to be reset. The IWMRESET service has the same restrictions as the RESET system command. Refer to *z/OS MVS System Commands* for more information.

---

### Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Supervisor state or program key mask (PKM) allowing keys 0-7.
<b>Dispatchable unit mode:</b>	Task
<b>Cross memory mode:</b>	PASN=HASN=SASN
<b>AMODE:</b>	31-bit
<b>ASC mode:</b>	Primary
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks may be held.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

---

### Programming Requirements

1. The macro CVT must be included to use this macro.
2. The macro IWMYCON must be included to use this macro.
3. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
4. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, IWMRSNCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.

---

### Restriction

The caller cannot have an EUT FRR established.

---

### Input Register Information

Before issuing the IWMRESET macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

## Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Reason code if GR15 return code is non-zero
1	Used as work registers by the system
2-13	Unchanged
14	Used as work registers by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

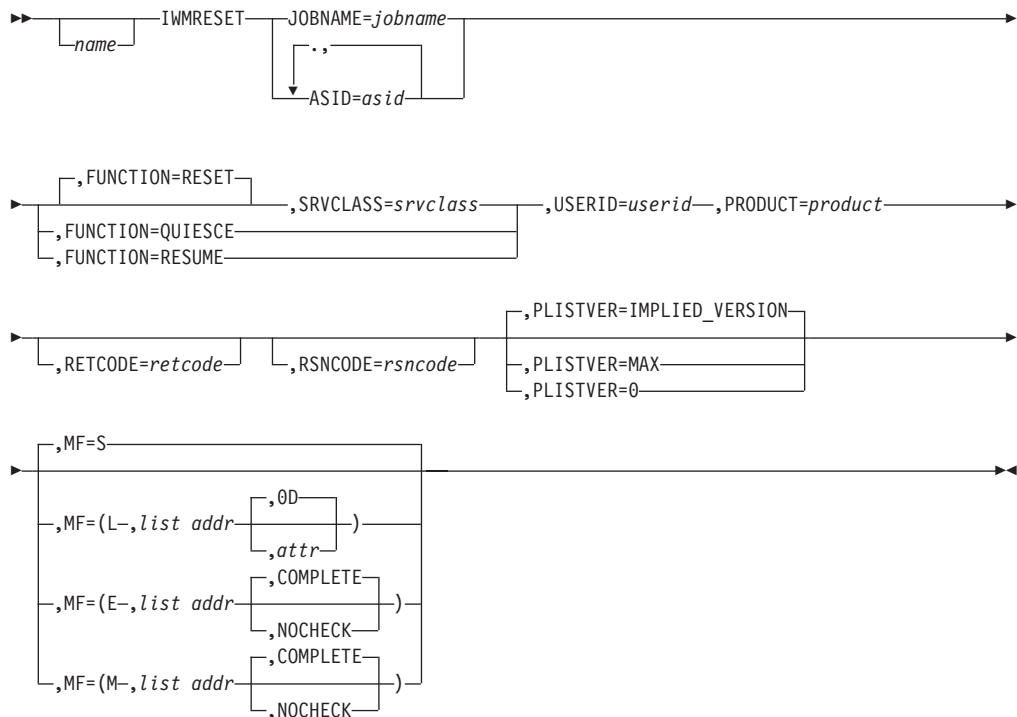
Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

## Performance Implications

None.

## Syntax

### main diagram



## Parameters

The parameters are explained as follows:

### *name*

An optional symbol, starting in column 1, that is the name on the IWMRESET macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

### **ASID=***asid*

A parameter which contains the address space identifier (ASID) of the job you want to change.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a halfword field.

### **,FUNCTION=RESET**

### **,FUNCTION=QUIESCE**

### **,FUNCTION=RESUME**

An optional parameter, which indicates the function to perform against the job. The default is FUNCTION=RESET.

### **,FUNCTION=RESET**

Requests that the job's service class or performance group be changed.

### **,FUNCTION=QUIESCE**

Requests that the job be quiesced. If the job is non-swappable, it is given the lowest possible performance characteristics.

### **,FUNCTION=RESUME**

Requests that the job be reclassified according to the service policy in effect. This undoes a prior request to reset the job to a particular service class, or to quiesce the job.

### **JOBNAME=***jobname*

A required input parameter which contains the jobname of the job you want to change. If there is more than one executing job with this jobname, you must specify the ASID parameter to identify the specific job.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

### **,MF=S**

### **,MF=(L,list addr)**

### **,MF=(L,list addr,attr)**

### **,MF=(L,list addr,OD)**

### **,MF=(E,list addr)**

### **,MF=(E,list addr,COMPLETE)**

### **,MF=(E,list addr,NOCHECK)**

### **,MF=(M,list addr)**

### **,MF=(M,list addr,COMPLETE)**

### **,MF=(M,list addr,NOCHECK)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The

## IWMRESET Macro

list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

Use MF=M together with the list and execute forms of the macro for service routines that need to provide different options according to user-provided input. Use the list form to define a storage area; use the modify form to set the appropriate options; then use the execute form to call the service.

IBM recommends that you use the modify and execute forms of IWMRESET in the following order:

- Use IWMRESET ...MF=(M,list-addr,COMPLETE) specifying appropriate parameters, including all required ones.
- Use IWMRESET ...MF=(M,list-addr,NOCHECK), specifying the parameters that you want to change.
- Use IWMRESET ...MF=(E,list-addr,NOCHECK), to execute the macro.

*,list addr*

The name of a storage area to contain the parameters. For MF=S, MF=E, and MF=M, this can be an RS-type address or an address in register (1)-(12).

*,attr*

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

**,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

**,NOCHECK**

Specifies that the system is not to check for required parameters and is not to supply defaults for omitted optional parameters.

**,PLISTVER=IMPLIED\_VERSION**

**,PLISTVER=MAX**

**,PLISTVER=0**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that

the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form; in this way, MAX ensures that the parameter list does not overwrite nearby storage.

- 0, if you use the currently available parameters.

**To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX
- A decimal value of 0

**,PRODUCT=product**

A required input parameter, which contains the product name that is requesting the job be changed. The product name is included in the SMF 90 subtype 30 record created by IWMRESET.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

**,RETCODE=retcode**

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,RSNCODE=rsncode**

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,SRVCLASS=svclass**

When FUNCTION=RESET is specified, a required input parameter which is the service class to be assigned to the job. Resetting to a new service class also resumes quiesced work.

When you reset a server to a new service class, the goals associated with that service class are ignored. However the resource group associated with the new service class is honored. The one exception where the goal for a server is honored is when the transactions it is serving have been assigned a discretionary goal.

**To code:** Specify the RS-type address of an 8-character field.

**,USERID=userid**

A required input parameter, which contains the id of the user who is requesting the job be changed. The userid is included in the SMF 90 subtype 30 record created by IWMRESET. If there is no userid available, the caller should pass blanks.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

---

## ABEND Codes

None.

---

## Return and Reason Codes

When the IWMRESET macro returns control to your program:

- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.

## IWMRESET Macro

- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, if you coded RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 52. Return and Reason Codes for the IWMRESET Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> <code>IwmRetCodeOk</code> <b>Meaning:</b> Successful completion. <b>Action:</b> None required.
4	—	<b>Equate Symbol:</b> <code>IwmRetCodeWarning</code> <b>Meaning:</b> Successful completion, unusual conditions noted.
4	xxxx0408	<b>Equate Symbol:</b> <code>IwmRsnCodeWorkNotFound</code> <b>Meaning:</b> A job matching the input job name or ASID was not found. <b>Action:</b> The caller should report the error appropriately.
8	—	<b>Equate Symbol:</b> <code>IwmRetCodeInvocError</code> <b>Meaning:</b> Invalid invocation environment or parameters.
8	xxxx0801	<b>Equate Symbol:</b> <code>IwmRsnCodeSRBMode</code> <b>Meaning:</b> Caller is in SRB mode. <b>Action:</b> Avoid requesting this function while in SRB mode.
8	xxxx0803	<b>Equate Symbol:</b> <code>IwmRsnCodeDisabled</code> <b>Meaning:</b> Caller is disabled. <b>Action:</b> Avoid requesting this function while disabled.
8	xxxx0804	<b>Equate Symbol:</b> <code>IwmRsnCodeLocked</code> <b>Meaning:</b> Caller is locked. <b>Action:</b> Avoid requesting this function while locked.
8	xxxx080B	<b>Equate Symbol:</b> <code>IwmRsnCodeBadPI</code> <b>Meaning:</b> Error accessing parameter list. <b>Action:</b> Check for possible storage overlay.
8	xxxx0810	<b>Equate Symbol:</b> <code>IwmRsnCodeEutFrr</code> <b>Meaning:</b> Caller has EUT FRR established. <b>Action:</b> Avoid requesting this function with an EUT FRR set.
8	xxxx0824	<b>Equate Symbol:</b> <code>IwmRsnCodeAmode24</code> <b>Meaning:</b> Caller invoked service but was in 24 bit addressing mode. <b>Action:</b> Request this function only when you are in 31 bit addressing mode.



Table 52. Return and Reason Codes for the IWMRESET Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0825	<b>Equate Symbol:</b> lwmRsnCodeAscModeNotPrimary <b>Meaning:</b> Caller invoked service but was not in primary ASC mode. <b>Action:</b> Avoid requesting this function in this environment.
8	xxxx0827	<b>Equate Symbol:</b> lwmRsnCodeRsvdNot0 <b>Meaning:</b> Reserved field in parameter list was non-zero. <b>Action:</b> Check for use of keywords that are not supported by the MVS release on which the program is running.
8	xxxx0828	<b>Equate Symbol:</b> lwmRsnCodeBadVersion <b>Meaning:</b> Version number in parameter list is not valid. <b>Action:</b> Check for possible storage overlay of the parameter list.
8	xxxx0829	<b>Equate Symbol:</b> lwmRsnCodeBadOptions <b>Meaning:</b> Parameter list omits required parameters. <b>Action:</b> Check for possible storage overlay of the parameter list.
8	xxxx0841	<b>Equate Symbol:</b> lwmRsnCodeXMemMode <b>Meaning:</b> Caller is in cross memory mode. <b>Action:</b> Invoke the function in non-cross memory mode.
C	—	<b>Equate Symbol:</b> lwmRetCodeEnvError <b>Meaning:</b> Environmental error.
C	xxxx0C28	<b>Equate Symbol:</b> lwmRsnCodeBadServiceClass <b>Meaning:</b> The input service class name is not defined in the active workload manager policy. <b>Action:</b> Record or report the error if appropriate.
C	xxxx0C2D	<b>Equate Symbol:</b> lwmRsnCodeBadPerformanceGroup <b>Meaning:</b> Reserved
C	xxxx0C2E	<b>Equate Symbol:</b> lwmRsnCodeWrongMode <b>Meaning:</b> The caller tried to perform a goal-mode function in compatibility mode, or vice versa. <b>Action:</b> Record or report the error if appropriate.
C	xxxx0C2F	<b>Equate Symbol:</b> lwmRsnCodeSystemSpace <b>Meaning:</b> The input address space is either a system component address space or a privileged address space. It cannot be reset.  With APAR OA12625 installed, the restriction for privileged address spaces has been removed, meaning that privileged address spaces can be reset. System component address space still cannot be reset. <b>Action:</b> Record or report the error if appropriate.

## IWMRESET Macro

Table 52. Return and Reason Codes for the IWMRESET Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
C	xxxx0C30	<p><b>Equate Symbol:</b> IwmRsnCodeDuplicateJobs</p> <p><b>Meaning:</b> There is more than one job active with the same jobname.</p> <p><b>Action:</b> Specify the ASID parameter to identify the specific job.</p>
C	xxxx0C31	<p><b>Equate Symbol:</b> IwmRsnCodeWrongASID</p> <p><b>Meaning:</b> The active job in the specified address space has a different jobname than the one passed by the caller.</p> <p><b>Action:</b> Record or report the error if appropriate.</p>
C	xxxx0C32	<p><b>Equate Symbol:</b> IwmRsnCodeNotEligibleForSrvClass</p> <p><b>Meaning:</b> The active job in the specified address space is not eligible for reset into the specified system service class. Only address spaces created with the ASCRE HIPRI attribute are eligible for reset into the SYSTEM service class.</p> <p><b>Action:</b> Record or report the error if appropriate.</p>
10	—	<p><b>Equate Symbol:</b> IwmRetCodeCompError</p> <p><b>Meaning:</b> Component error.</p> <p><b>Action:</b> Contact your system programmer.</p>

## Example

To change the service class of the job executing in the ASID at location JOBASID:

```
IWMRESET ASID=JOBASID,SRVCLASS=SCNAME,USERID=USR,
        PRODUCT=PROD
```

```
*
* Storage areas
*
JOBASID DS H           Contains the address space id
*                               of the job
SCNAME DS CL8         Contains the service class name
*                               to assign to the job
USR DS CL8            Contains the id of the user who
*                               is requesting the change
PROD DS CL8           Contains the product name of
*                               the code invoking IWMRESET
```

---

## Chapter 46. IWMRPT – Report on Work Request Completion

The primary purpose of this service is to allow MVS to obtain the total response time for a completed work request and its corresponding service class and (when customer specified) its report class.

The second purpose in using this service is to allow MVS to know which address spaces were involved in serving the service class.

When a monitoring token is provided, the third purpose in using this service is to allow MVS to know that the monitoring environment should no longer be associated with the now completed work request. The use of this service will render the information associated with the monitoring environment unpredictable. To associate a work request with the monitoring environment following use of Report, first use Initialize Mode(Reset) or Relate/Transfer.

---

### Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Supervisor state. PSW key must either be 0 or match the value supplied on IWM4CON. PSW key must either be 0 or match the value supplied on IWM4MCRE when a monitoring token is passed. PSW key must be 0-7. See restrictions below.
<b>Dispatchable unit mode:</b>	Task or SRB
<b>Cross memory mode:</b>	Any PASN, any HASN, any SASN
<b>AMODE:</b>	31-bit
<b>ASC mode:</b>	Primary
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	LOCAL lock held
<b>Control parameters:</b>	Control parameters must be in the primary address space.

---

### Programming Requirements

1. The macro CVT must be included to use this macro.
2. The macro IWMYCON must be included to use this macro.
3. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
4. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, IWMSRNCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.
5. All character data, unless otherwise specified, is assumed to be left justified and padded with blanks on the right, as needed, to occupy the specified number of bytes.

---

### Restrictions

1. Caller is responsible for error recovery

## IWMRPT Macro

2. Though the caller is required to be enabled, this is not checked. Violation of this restriction may cause disabled program checks which would be the responsibility of the caller's recovery to handle.
3. If a delay monitoring token is provided, then
  - The caller must serialize to prevent any delay monitoring services from being invoked concurrently for the environment represented by the monitoring token.
  - The monitoring environment must contain the information saved by IWM4MINI, not IWMMRELA
  - If the key specified on IWM4MCRE was a system key (0-7), then the current PSW key must be 0 or match the key specified on IWM4MCRE.
  - If the key specified on IWM4MCRE was a user key (8-F), then:
    - PSW key must be 0
    - Current primary must match the primary at the time that IWM4MCRE was invoked. Calling from a subspace is not supported.
4. If the key specified on IWM4CON for the input connect token was a user key (8-F), then:
  - PSW key must be 0
  - Current primary must match the primary at the time that IWM4CON was invoked. Calling from a subspace is not supported.
5. This macro supports multiple versions. Some keywords are only supported by certain versions. Refer to the PLISTVER parameter description for further information.

---

## Input Register Information

Before issuing the IWMRPT macro, the caller must ensure that the following general purpose registers (GPRs) contain the specified information:

<b>Register</b>	<b>Contents</b>
<b>13</b>	The address of a 72-byte standard save area in the primary address space

Before issuing the IWMRPT macro, the caller does not have to place any information into any AR unless using it in register notation for a particular parameter, or using it as a base register.

---

## Output Register Information

When control returns to the caller, the GPRs contain:

<b>Register</b>	<b>Contents</b>
<b>0-1</b>	Used as work registers by the system
<b>2-13</b>	Unchanged
<b>14-15</b>	Used as work registers by the system.

When control returns to the caller, the ARs contain:

<b>Register</b>	<b>Contents</b>
<b>0-1</b>	Used as work registers by the system
<b>2-13</b>	Unchanged
<b>14-15</b>	Used as work registers by the system.

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

## Performance Implications

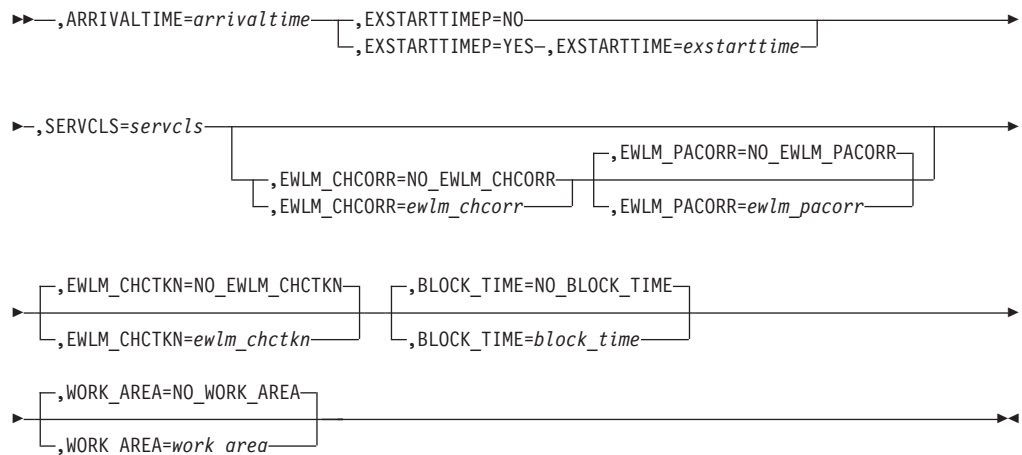
None.

## Syntax

### main diagram



### parameters-1



## Parameters

The parameters are explained as follows:

## IWMRPT Macro

### *name*

An optional symbol, starting in column 1, that is the name on the IWMRPT macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

### **,ARRIVALTIME=***arrivaltime*

When MONTKNI=NO is specified, a required input parameter, which contains the arrival time for the work unit in STCK format.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 64-bit field.

### **,BLOCK\_TIME=***block\_time*

### **,BLOCK\_TIME=NO\_BLOCK\_TIME**

When MONTKNI=NO is specified, an optional input parameter, which contains the duration where the work request has been blocked. The format of the field is STCK. A work request is blocked, when the transaction processing is waiting on an external transaction processing or some other event to complete. The default is NO\_BLOCK\_TIME indicates that no block time is passed.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

### **,CONNTKN=***conntkn*

A required input parameter, which is returned by IWM4CON.

If a monitoring token is passed (MONTKNI(YES)), AND this monitoring token was obtained using a connect token on IWM4MCRE, then the latter connect token is expected to be the same as that specified for IWMRPT.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32 bit field.

### **,ENDTIME=***endtime*

### **,ENDTIME=CURRENT**

An optional input parameter, which specifies the ending time for the transaction (typically, when the output is sent or available to be sent) in STCK format. The default is CURRENT. indicates that the current time should be used.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 64 bit field.

### **,EWLM\_CHCORR=***ewlm\_chcorr*

### **,EWLM\_CHCORR=NO\_EWLM\_CHCORR**

When MONTKNI=NO is specified, an optional input parameter, which contains the cross-platform Enterprise Workload Management (EWLM) correlator associated with the work request.

**Note:** If this correlator is not a valid ARM correlator, return code 8 and reason code `IwmRsnCodeInvalidEWLMCorr` is returned to the caller (see return code section below). If the correlator is valid, but cannot be understood by EWLM (no EWLM format), the correlator is silently ignored and the work request will not be reported to EWLM.

The EWLM\_CHCORR and EWLM\_CHCTKN parameters are mutually exclusive.

The default is NO\_EWLM\_CHCORR. It indicates that no EWLM correlator is passed.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a character field.

### **,EWLM\_CHCTKN=***ewlm\_chctkn*

**,EWLM\_CHCTKN=NO\_EWLM\_CHCTKN**

When MONTKNI=NO is specified, an optional input parameter, which contains the cross-platform Enterprise Workload Management (EWLM) correlator token associated with the work request. The EWLM\_CHCORR and EWLM\_CHCTKN parameters are mutually exclusive. The default is NO\_EWLM\_CHCTKN. It indicates that no EWLM correlator token is passed.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a character field.

**,EWLM\_PACORR=*ewlm\_pacorr*****,EWLM\_PACORR=NO\_EWLM\_PACORR**

When EWLM\_CHCORR=*ewlm\_chcorr* and MONTKNI=NO are specified, an optional input parameter, which contains the cross-platform Enterprise Workload Management (EWLM) parent correlator associated with the work request. The default is NO\_EWLM\_PACORR. It indicates that no EWLM parent correlator is passed.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a character field.

**,EXSTARTTIME=*exstarttime***

When EXSTARTTIMEP=YES and MONTKNI=NO are specified, a required input parameter, which contains the start execution time in STCK format. Note that this should only be used when IWMMNTFY was NOT used to pass the execution time for this work request.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 64 bit field.

**,EXSTARTTIMEP=NO****,EXSTARTTIMEP=YES**

When MONTKNI=NO is specified, a required parameter, which indicates whether the start execution time value is passed.

**,EXSTARTTIMEP=NO**

indicates that the start execution time value is not passed.

**,EXSTARTTIMEP=YES**

indicates that the start execution time value is passed. Note that this should only be used when IWMMNTFY was NOT used to pass the execution time for this work request.

**,MF=S****,MF=(L,*list addr*)****,MF=(L,*list addr,attr*)****,MF=(L,*list addr,OD*)****,MF=(E,*list addr*)****,MF=(E,*list addr,COMPLETE*)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

## IWMRPT Macro

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

*,list addr*

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

*,attr*

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

**,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

**,MONTKN=*montkn***

When MONTKNI=YES is specified, a required input parameter, which contains the delay monitoring token

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32 bit field.

**,MONTKNI=YES**

**,MONTKNI=NO**

A required parameter, which indicates whether a delay monitoring token is provided.

**,MONTKNI=YES**

indicates that a delay monitoring token is provided.

**,MONTKNI=NO**

indicates that no delay monitoring token is provided.

**,OK\_THRESHOLD=*ok\_threshold***

When STATUS=NORMAL\_LE\_VAL is specified, a required input parameter, which contains the threshold value at which the work request is considered to have ended normally.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

**,OK\_THRESHOLD=*ok\_threshold***

When STATUS=NORMAL\_GE\_VAL is specified, a required input parameter, which contains the threshold value at which the work request is considered to have ended normally.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

**,PLISTVER=IMPLIED\_VERSION**

**,PLISTVER=MAX**

**,PLISTVER=0**

**,PLISTVER=1**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When



using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form; in this way, MAX ensures that the parameter list does not overwrite nearby storage.

- **0**, which supports all parameters except those specifically referenced in higher versions.
- **1**, which supports both the following parameters and those from version 0:

BLOCK_TIME	EWLM_CHCTKN	WORK_AREA
EWLM_CHCORR	EWLM_PACORR	WORKREQ_STA

**To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX
- A decimal value of 0, or 1

**,PSWKEY=pswkey**

When PSWKEYP=VALUE and TRAXFRPT=NO are specified, a required input parameter, which contains the current PSW key. The low order 4 bits (bits 4-7) contain the key value. The high order 4 bits (bits 0-3) contain zeros.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8 bit field.

**,PSWKEYP=CURRENT**

**,PSWKEYP=VALUE**

When TRAXFRPT=NO is specified, an optional parameter, which describes how to determine the current PSW key. The default is PSWKEYP=CURRENT.

**,PSWKEYP=CURRENT**

indicates that the current PSW key should be determined.

**,PSWKEYP=VALUE**

indicates that the key is being passed explicitly via PSWKEY.

**,RETCODE=retcode**

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,RSNCODE=rsncode**

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,SERVCLS=servcls**

When MONTKNI=NO is specified, a required input parameter, which contains the service class token.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32 bit field.

**,STATUS=NORMAL**  
**,STATUS=ABNORMAL**  
**,STATUS=NORMAL\_LE\_VAL**  
**,STATUS=NORMAL\_GE\_VAL**

An optional parameter, which indicates whether the portion of the work request associated with the Report call has completed normally or not. The default is STATUS=NORMAL.

**,STATUS=NORMAL**  
 indicates that work request execution associated with the Report call has completed normally.

**,STATUS=ABNORMAL**  
 indicates that work request execution associated with the Report call has completed abnormally.

**,STATUS=NORMAL\_LE\_VAL**  
 indicates that work request execution associated with the Report call has completed normally PROVIDED the work completion code is below or at (<=) the threshold value given by OK\_THRESHOLD.

**,STATUS=NORMAL\_GE\_VAL**  
 indicates that work request execution associated with the Report call has completed normally PROVIDED the work completion code is above or at (>=) the threshold value given by OK\_THRESHOLD.

**,SYSEVPL=***sysevp/*  
 When TRAXFRPT=YES is specified, a required input parameter, which is the fully initialized SYSEVENT parameter list, as mapped by IHATRBPL.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 40-character field.

**TRAXFRPT=NO**  
**TRAXFRPT=YES**

An optional parameter, which indicated prior to z/OS R3 whether a SYSEVENT TRAXFRPT should be issued when the system was in compatibility mode. This has become irrelevant. However, for compatibility reasons TRAXFRPT can still be set but has no effect. The default is TRAXFRPT=NO.

**TRAXFRPT=NO**  
 indicates that no SYSEVENT TRAXFRPT should be issued.

**TRAXFRPT=YES**  
 indicated prior to z/OS R3 that a SYSEVENT TRAXFRPT should be issued when the system was in compatibility mode. This has become irrelevant. However, for compatibility reasons TRAXFRPT can still be set but has no effect.

**,WORK\_AREA=***work\_area*  
**,WORK\_AREA=NO\_WORK\_AREA**

When MONTKNI=NO is specified, an optional input parameter, which is used as a work area by WLM when MONTKNI(NO) is specified and either EWLM\_CHCORR or EWLM\_CHTKN is specified on the IWMRPT invocation (in these cases WORK\_AREA is required). The work area must begin on a doubleword boundary and must be accessible in the current PSW key when the macro is invoked. The default is NO\_WORK\_AREA. It indicates that no work area is passed.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 256-character field.

**,WORK\_COMPCD=***work\_compcd*

**,WORK\_COMPCD=NO\_WORK\_COMPCD**

When STATUS=NORMAL is specified, an optional input parameter, which contains the completion/return code for the work request execution associated with the Report call. The default is NO\_WORK\_COMPCD. indicates that NO completion/return code is passed.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

**,WORK\_COMPCD=***work\_compcd*

**,WORK\_COMPCD=NO\_WORK\_COMPCD**

When STATUS=ABNORMAL is specified, an optional input parameter, which contains the completion/return code for the work request execution associated with the Report call. The default is NO\_WORK\_COMPCD. indicates that NO completion/return code is passed.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

**,WORK\_COMPCD=***work\_compcd*

When STATUS=NORMAL\_LE\_VAL is specified, a required input parameter, which contains the completion/return code for the work request execution associated with the Report call.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

**,WORK\_COMPCD=***work\_compcd*

When STATUS=NORMAL\_GE\_VAL is specified, a required input parameter, which contains the completion/return code for the work request execution associated with the Report call.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

**,WORKREQ\_STA=***workreq\_sta*

**,WORKREQ\_STA=IWMEWLMARMSTATUSNONE**

An optional input parameter, which contains the completion status code of the work request. Available completion status codes (defined in macro IWMYCON) are: \* IwmEwlmArmStatusGood(0), \* IwmEwlmArmStatusAborted(1), \* IwmEwlmArmStatusFailed(2) or \* IwmEwlmArmStatusUnknown(3) The codes above correspond to status codes in the OpenGroup ARM 4.0 Standard (for the meaning of the status codes see the ARM 4.0 Standard at <http://www.opengroup.org/management/arm>). The default is IWMEWLMARMSTATUSNONE. indicates that work request completion status should be derived from the passed STATUS parameter value.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

---

## ABEND Codes

None.

## Return and Reason Codes

When the IWMRPT macro returns control to your program:

- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, if you coded RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 53. Return and Reason Codes for the IWMRPT Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<p><b>Equate Symbol:</b> <code>IwmRetCodeOk</code></p> <p><b>Meaning:</b> Successful completion.</p> <p><b>Action:</b> None required.</p>
4	—	<p><b>Equate Symbol:</b> <code>IwmRetCodeWarning</code></p> <p><b>Meaning:</b> Successful completion, unusual conditions noted.</p>
4	xxxx0401	<p><b>Equate Symbol:</b> <code>IwmRsnCodeNoWLM</code></p> <p><b>Meaning:</b> System does not support delay monitoring services. This return code is only set when the MVS release is prior to MVS/ESA SP5.1.0.</p> <p><b>Action:</b> None required.</p>
4	xxxx0405	<p><b>Equate Symbol:</b> <code>IwmRsnCodeGoalNoMonEnv</code></p> <p><b>Meaning:</b> System is in goal mode but the input monitoring token indicates no monitoring environment was established, hence MVS did not receive the information.</p> <p><b>Action:</b> None required.</p>
4	xxxx0409	<p><b>Equate Symbol:</b> <code>IwmRsnCodeNoConn</code></p> <p><b>Meaning:</b> Connect token does not reflect a successful Connect. The system did not receive the information (applies to both goal mode and compatibility mode). No SYSEVENT TRAXFRPT was issued.</p> <p><b>Action:</b> None required.</p>
8	—	<p><b>Equate Symbol:</b> <code>IwmRetCodeInvocError</code></p> <p><b>Meaning:</b> Invalid invocation environment or parameters.</p>
8	xxxx080C	<p><b>Equate Symbol:</b> <code>IwmRsnCodeMonEnvLacksData</code></p> <p><b>Meaning:</b> Input monitoring environment does not contain the necessary information.</p> <p><b>Action:</b> Ensure that the monitoring environment was established with the necessary information.</p>
8	xxxx080E	<p><b>Equate Symbol:</b> <code>IwmRsnCodeArrTimeGTEndTime</code></p> <p><b>Meaning:</b> Input arrival time later than end time.</p> <p><b>Action:</b> Check for possible storage overlay of the parameter list or variable.</p>

Table 53. Return and Reason Codes for the IWMRPT Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0820	<p><b>Equate Symbol:</b> IwmRsnCodeBadMonEnv</p> <p><b>Meaning:</b> Input monitoring environment does not pass short form validity checking.</p> <p><b>Action:</b> Check for possible storage overlay.</p>
8	xxxx0821	<p><b>Equate Symbol:</b> IwmRsnCodeBadConn</p> <p><b>Meaning:</b> Input connect token does not pass validity checking.</p> <p><b>Action:</b> Check for possible storage overlay.</p>
8	xxxx082D	<p><b>Equate Symbol:</b> IwmRsnCodeExStTimeGTEndTime</p> <p><b>Meaning:</b> Execution start time greater than execution end time</p> <p><b>Action:</b> Check for possible storage overlay of the parameter list or variable.</p>
8	xxxx0840	<p><b>Equate Symbol:</b> IwmRsnCodeServiceNotEnabled</p> <p><b>Meaning:</b> Caller's space connection is not enabled for this service.</p> <p><b>Action:</b> Avoid requesting this function under the input connection. IWM4CON options must be specified previously to enable this service.</p>
8	xxxx087E	<p><b>Equate Symbol:</b> IwmRsnCodeRoMonEnv</p> <p><b>Meaning:</b> Input monitoring environment is report-only checking.</p> <p><b>Action:</b> Avoid calling this function for report-only monitoring environments.</p>
8	xxxx0894	<p><b>Equate Symbol:</b> IwmRsnCodeInvalidEWLMCorr</p> <p><b>Meaning:</b> Passed correlator information (EWLM_CHCORR, EWLM_PACORR, or EWLM_CHCTKN) did not pass validity checking, that means: the architected ARM correlator length field in the first two Bytes of the correlator (token) is either less than 4 ('0004'x) or greater than 512 ('0200'x).</p> <p><b>Action:</b> Check the specification of the correlator information.</p>
8	xxxx0895	<p><b>Equate Symbol:</b> IwmRsnCodeEWLMSErvNotEnabled</p> <p><b>Meaning:</b> Service is not enabled because caller invoked the IWMCONN service with EWLM=NO.</p> <p><b>Action:</b> Specify the parameter EWLM_CHCORR, EWLM_PACORR, EWLM_CHCTKN, or WORKREQ_STA only when connected with EWLM=YES.</p>
8	xxxx0897	<p><b>Equate Symbol:</b> IwmRsnCodeTranStatusInvalid</p> <p><b>Meaning:</b> Passed work request completion status is not valid.</p> <p><b>Action:</b> Check the EWLM ARM interface specification for valid completion status values.</p>
C	—	<p><b>Equate Symbol:</b> IwmRetCodeEnvError</p> <p><b>Meaning:</b> Environmental error.</p>

## IWMRPT Macro

Table 53. Return and Reason Codes for the IWMRPT Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
C	xxxx0C02	<p><b>Equate Symbol:</b> IwmRsnCodeReportingSusp</p> <p><b>Meaning:</b> SYSEVENT TRAXFRPT invoked, but reporting is temporarily suspended for one of the following reasons:</p> <ul style="list-style-type: none"> <li>• RMF workload activity reporting is not active</li> <li>• There is no installation control specification (IEAICSxx parmlib member with RPGN specified for some subsystem other than TSO) in effect. No data reported but a later reissue could be successful.</li> </ul> <p><b>Action:</b> Invoke the function when the conditions are alleviated.</p>
C	xxxx0C03	<p><b>Equate Symbol:</b> IwmRsnCodeSyseventNoWorkElt</p> <p><b>Meaning:</b> SYSEVENT TRAXFRPT invoked, but no work element was available to save the input information.</p> <p><b>Action:</b> Invoke the function when the conditions are alleviated. This condition may be due to a common storage shortage condition.</p>
C	xxxx0C05	<p><b>Equate Symbol:</b> IwmRsnCodeRptNoWorkElt</p> <p><b>Meaning:</b> Report routine invoked, but no work element was available to save the input information.</p> <p><b>Action:</b> Invoke the function when the conditions are alleviated. This condition may be due to a common storage shortage condition.</p>
C	xxxx0C06	<p><b>Equate Symbol:</b> IwmRsnCodeNoEndTime</p> <p><b>Meaning:</b> No end time was supplied to the service and STCK gave a non-zero condition code.</p> <p><b>Action:</b> No action required.</p>

---

## Chapter 47. IWMRQRY – Collecting Address Space Delay Information

IWMRQRY is the interface reporting products should use to obtain address space related general execution delays. Enclave related information may optionally be requested.

The macro will complete the parameter list with caller specified data and invoke a stacking, space switching PC routine in the WLM address space. Address space related data collected will be aggregated on an address space basis, while Enclave related data (if requested) will be aggregated by Enclave.

If a user does not know the size of the answer area required by the service, he should code issue IWMRQRY with ANSLEN set to zero. The length of the answer area will be placed in QRYLEN.

The IWMRQRY macro is provided in list, execute, and standard form. The list form accepts no variable parameters and is used only to reserve space for the parameter list. The standard form is provided for use with routines which do not require reentrant code. The execute form is provided for use with the list format for reentrant routines.

The parameter list must be in the caller's primary address space, or in a dataspace accessible by the current unit of work's dispatchable unit access list (DUaI).

---

### Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Supervisor state or program key mask (PKM) allowing keys 0-7.
<b>Dispatchable unit mode:</b>	Task
<b>Cross memory mode:</b>	Any PASN, any HASN, any SASN
<b>AMODE:</b>	31-bit
<b>ASC mode:</b>	Primary or access register (AR) Any P,S,H.  If in Access Register ASC mode, specify SYSSTATE ASCENV=AR before invoking this macro.
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks may be held.
<b>Control parameters:</b>	The caller of IWMRQRY must provide storage for an answer area mapped by IWMWRQAA. This answer area may reside in the caller's primary address space, or in a dataspace accessible via the current unit of work's dispatchable unit access list (DUaI).

---

### Programming Requirements

1. The macro CVT must be included to use this macro.
2. The functions provided by this macro are only supported on MVS/ESA SP5.2.0 or later.

## IWMRQRY Macro

3. The ENCLAVES=ONE keyword is only supported on OS/390 R4 or later. If code with this keyword expansion is executed on lower OS/390 releases a return code of 8 and reason code of xxxx0828 (constant IwmRsnCodeBadVersion) will be returned indicating that the version is incorrect.
4. The macro IWMYCON must be included to use this macro.
5. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
6. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, IWMSNCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.

---

## Restrictions

The caller cannot have an EUT FRR established.

---

## Input Register Information

Before issuing the IWMRQRY macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

---

## Output Register Information

When control returns to the caller, the GPRs contain:

<b>Register</b>	<b>Contents</b>
<b>0</b>	Reason code if GR15 return code is non-zero
<b>1</b>	Used as work registers by the system
<b>2-13</b>	Unchanged
<b>14</b>	Used as a work register by the system
<b>15</b>	Return code

When control returns to the caller, the ARs contain:

<b>Register</b>	<b>Contents</b>
<b>0-1</b>	Used as work registers by the system
<b>2-13</b>	Unchanged
<b>14-15</b>	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

---

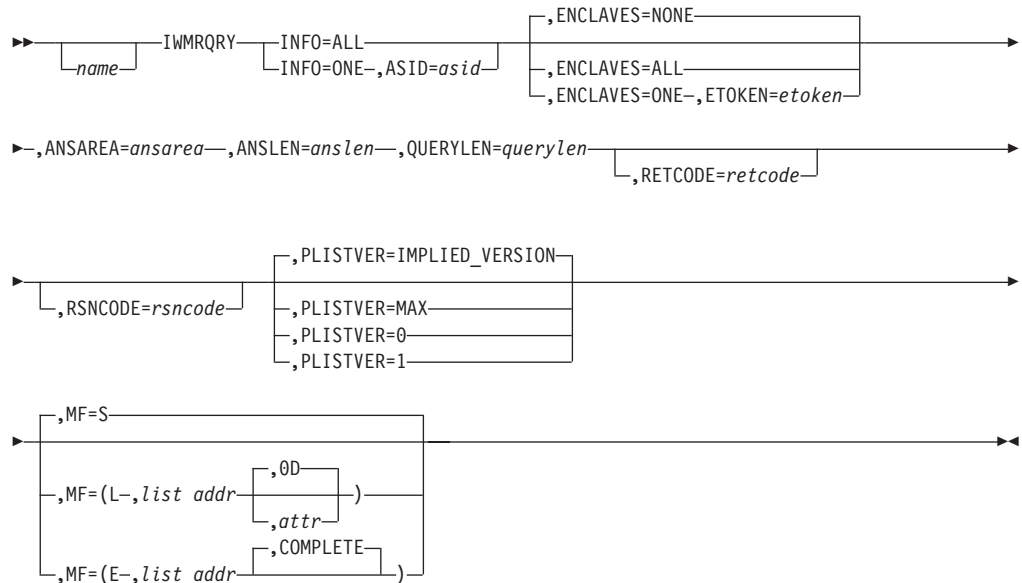
## Performance Implications

None.



## Syntax

### main diagram



## Parameters

The parameters are explained as follows:

### *name*

An optional symbol, starting in column 1, that is the name on the IWMRQRY macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

### **,ANSAREA=ansarea**

A required output parameter, variable specifying an area to contain the data returned by the query service. If the length of the output area is insufficient, no data is returned.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a character field.

### **,ANSLEN=anslen**

A required input parameter, which contains the length of the answer area.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

### **,ASID=asid**

When INFO=ONE is specified, a required input parameter, which contains the ASID of the address space to be queried.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

### **,ENCLAVES=NONE**

### **,ENCLAVES=ALL**

### **,ENCLAVES=ONE**

An optional parameter, which indicates whether Enclave information is requested. The default is ENCLAVES=NONE.

### **,ENCLAVES=NONE**

indicates that no Enclave information is requested.

### **,ENCLAVES=ALL**

indicates that information for all Enclaves should be returned.

### **,ENCLAVES=ONE**

indicates that Enclave information is requested for a particular Enclave.

This ENCLAVES=ONE keyword is only supported on OS/390 R4 or later.

### **,ETOKEN=*etoken***

When ENCLAVES=ONE is specified, a required input parameter, which contains the Enclave token of the Enclave to be queried.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

### **INFO=ALL**

### **INFO=ONE**

A required parameter, which indicates what information the query service is to return

### **INFO=ALL**

indicates that information for all address spaces should be returned.

### **INFO=ONE**

indicates that address space information is requested for a particular asid.

### **,MF=S**

### **,MF=(L,*list addr*)**

### **,MF=(L,*list addr*,*attr*)**

### **,MF=(L,*list addr*,0D)**

### **,MF=(E,*list addr*)**

### **,MF=(E,*list addr*,COMPLETE)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

### **,*list addr***

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

### **,*attr***

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

**,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

**,PLISTVER=IMPLIED\_VERSION****,PLISTVER=MAX****,PLISTVER=0****,PLISTVER=1**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form; in this way, MAX ensures that the parameter list does not overwrite nearby storage.

- **0**, if you use only the following parameters:

ANSAREA	ASID	INFO
ANSLEN	ENCLAVES	QUERYLEN

- **1**, if you use any of the following parameters, and parameters from version 0, or both:

ETOKEN

**To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX
- A decimal value of 0 or 1

**,QUERYLEN=*querylen***

A required output parameter, which contains the length of the storage area required by the IWMRQRY service. The length of the area may change between invocations.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

**,RETCODE=*retcode***

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,RSNCODE=*rsncode***

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

---

## ABEND Codes

None.

---

## Return and Reason Codes

When the IWMRQRY macro returns control to your program:

- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, if you coded RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 54. Return and Reason Codes for the IWMRQRY Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<p><b>Equate Symbol:</b> <code>IwmRetCodeOk</code></p> <p><b>Meaning:</b> Successful completion.</p> <p><b>Action:</b> None required.</p>
4	—	<p><b>Equate Symbol:</b> <code>IwmRetCodeWarning</code></p> <p><b>Meaning:</b> Successful completion, unusual conditions noted.</p>
4	xxxx0401	<p><b>Equate Symbol:</b> <code>IwmRsnCodeNoWLM</code></p> <p><b>Meaning:</b> System does not support workload activity query services. The query output area has not been modified. This return code is set only when the MVS release is prior to MVS/ESA SP5.1.0.</p> <p><b>Action:</b> None required.</p>
4	xxxx040A	<p><b>Equate Symbol:</b> <code>IwmRsnCodeOutputAreaTooSmall</code></p> <p><b>Meaning:</b> The output area supplied is too small to receive all the available information.</p> <p><b>Action:</b> None required. If necessary, reinvoke the service with an output area of sufficient size to receive all information.</p>
4	xxxx042C	<p><b>Equate Symbol:</b> <code>IwmRsnCodeEtokenNoMatch</code></p> <p><b>Meaning:</b> No Enclave information matching the input Enclave token was found. Enclave related information is not returned.</p> <p><b>Action:</b> None required.</p>
8	—	<p><b>Equate Symbol:</b> <code>IwmRetCodeInvocError</code></p> <p><b>Meaning:</b> Invalid invocation environment or parameters.</p>
8	xxxx0801	<p><b>Equate Symbol:</b> <code>IwmRsnCodeSrbMode</code></p> <p><b>Meaning:</b> Caller is in SRB mode.</p> <p><b>Action:</b> Avoid requesting this function while in SRB mode.</p>
8	xxxx0803	<p><b>Equate Symbol:</b> <code>IwmRsnCodeDisabled</code></p> <p><b>Meaning:</b> Caller is disabled.</p> <p><b>Action:</b> Avoid requesting this function while disabled.</p>

Table 54. Return and Reason Codes for the IWMRQRY Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0804	<b>Equate Symbol:</b> lwmRsnCodeLocked <b>Meaning:</b> Caller is locked. <b>Action:</b> Avoid requesting this function while locked.
8	xxxx080B	<b>Equate Symbol:</b> lwmRsnCodeBadPI <b>Meaning:</b> Error accessing parameter list. <b>Action:</b> Check for possible storage overlay.
8	xxxx0810	<b>Equate Symbol:</b> lwmRsnCodeEutFrr <b>Meaning:</b> Caller has EUT FRR established. <b>Action:</b> Avoid requesting this function with an EUT FRR set.
8	xxxx0812	<b>Equate Symbol:</b> lwmRsnCodeBadAscb <b>Meaning:</b> The ASID value specified on the ASID keyword is invalid. <b>Action:</b> Check for possible storage overlay.
8	xxxx0828	<b>Equate Symbol:</b> lwmRsnCodeBadVersion <b>Meaning:</b> Version number in parameter list is not valid or version length field is not valid. <b>Action:</b> Check for possible storage overlay of the parameter list.
8	xxxx0830	<b>Equate Symbol:</b> lwmRsnCodeBadAlet <b>Meaning:</b> Caller has passed an invalid ALET. <b>Action:</b> Check for possible storage overlay of the parameter list or variable.
C	—	<b>Equate Symbol:</b> lwmRetCodeEnvError <b>Meaning:</b> Environmental error.
C	xxxx0C01	<b>Equate Symbol:</b> lwmRsnCodeNoStg <b>Meaning:</b> Storage is not available for the request. <b>Action:</b> There is a storage shortage. The function may work successfully at a later time.
C	xxxx0C0A	<b>Equate Symbol:</b> lwmRsnCodeSuspended <b>Meaning:</b> Data sampling is suspended as a result of a component error. No data can be returned for this IWMRQRY invocation. <b>Action:</b> Reinvoke the function as it may be successful.
C	xxxx0C0B	<b>Equate Symbol:</b> lwmRsnCodeStateChanged <b>Meaning:</b> A state change (a policy activation) occurred while the data for the last sampling interval was being collected. No data is returned for this invocation of IWMRQRY. <b>Action:</b> The current sampling interval should be bypassed, future invocations of IWMRQRY for subsequent sampling intervals should begin returning data again.

## IWMRQRY Macro

Table 54. Return and Reason Codes for the IWMRQRY Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
10	—	<b>Equate Symbol:</b> lwmRetCodeCompError <b>Meaning:</b> Component error. <b>Action:</b> No action required. The function may be successful if invoked again.

---

## Chapter 48. IWMSEDES – Scheduling Environments Determine Execution Service

The IWMSEDES service determines if a scheduling environment is available on a specified system. A scheduling environment is a list of resource names and their required states. If all of the resources are in the required state, the scheduling environment is available. If any of the resources is not in the required state, the scheduling environment is not available.

The caller can use the IWMSEDES service to perform certain work only when a particular scheduling environment is available.

If the scheduling environment is available, the caller receives return code `IwmRetCodeOK`. If the scheduling environment is not available, the caller receives return code `IwmRetCodeWarning` and reason code `IwmRsnCodeSCHENVNotAvailable`.

---

### Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Supervisor state or program key mask (PKM) allowing keys 0-7.
<b>Dispatchable unit mode:</b>	Task
<b>Cross memory mode:</b>	Any PASN, any HASN, any SASN
<b>AMODE:</b>	31-bit
<b>ASC mode:</b>	Primary
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks may be held.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

---

### Programming Requirements

1. The macro `CVT` must be included to use this macro.
2. The macro `IWMYCON` must be included to use this macro.
3. The macro `IWMPB` must be in the library concatenation, since it is included by `IWMYCON`.
4. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, `IWMRSNCODE_MASK_CONST` defined in `IWMYCON`, may be used for this purpose.

---

### Restrictions

None.

## Input Register Information

Before issuing the IWMSEDES macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

## Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Reason code if GR15 return code is non-zero
1	Used as work registers by the system
2-13	Unchanged
14	Used as work registers by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

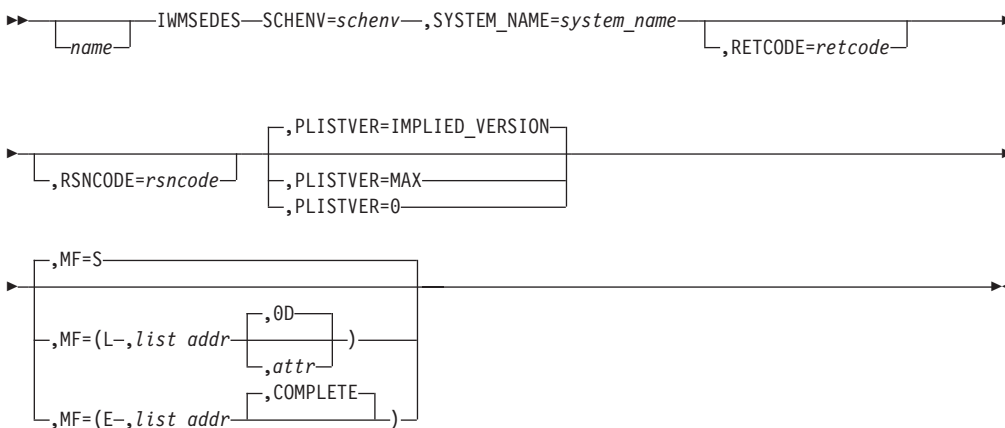
Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

## Performance Implications

None.

## Syntax

### main diagram



## Parameters

The parameters are explained as follows:



*name*

An optional symbol, starting in column 1, that is the name on the IWMSEDES macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

**,MF=S**

**,MF=(L,*list addr*)**

**,MF=(L,*list addr,attr*)**

**,MF=(L,*list addr,0D*)**

**,MF=(E,*list addr*)**

**,MF=(E,*list addr,COMPLETE*)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

*,list addr*

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

*,attr*

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

**,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

**,PLISTVER=IMPLIED\_VERSION**

**,PLISTVER=MAX**

**,PLISTVER=0**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that

## IWMSEDES Macro

the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form; in this way, MAX ensures that the parameter list does not overwrite nearby storage.

- 0, if you use the currently available parameters.

**To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX
- A decimal value of 0

**,RETCODE=***retcode*

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,RSNCODE=***rsncode*

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**SCHENV=***schenv*

A required input parameter, which contains the scheduling environment name to be checked.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 16-character field.

**,SYSTEM\_NAME=***system\_name*

A required input parameter, which contains the system name to be checked.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

---

## ABEND Codes

None.

---

## Return and Reason Codes

When the IWMSEDES macro returns control to your program:

- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, if you coded RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 55. Return and Reason Codes for the IWMSEDES Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	IwmRetCodeOk: Successful completion.
4	—	IwmRetCodeWarning: Successful completion, unusual conditions noted.

Table 55. Return and Reason Codes for the IWMSEDES Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
4	xxxx0425	<p><b>Equate Symbol:</b> lwmRsnCodeNoSCHENV:</p> <p><b>Meaning:</b> The system does not support scheduling environments services. This return code is set for releases prior to OS/390 Release 4.</p> <p><b>Action:</b> Avoid requesting this function on releases prior to OS/390 Release 4.</p>
4	xxxx0426	<p><b>Equate Symbol:</b> lwmRsnCodeSCHENVNotFound:</p> <p><b>Meaning:</b> The scheduling environment specified by SCHENV does not exist.</p> <p><b>Action:</b> Check the specification of the SCHENV parameter. If the SCHENV parameter is correct, check whether the scheduling environment is defined in the active service policy.</p>
4	xxxx0427	<p><b>Equate Symbol:</b> lwmRsnCodeSCHENVNotAvailable:</p> <p><b>Meaning:</b> The scheduling environment is not available on the specified system.</p> <p><b>Action:</b> Do not process work that depends upon the scheduling environment being available on the specified system.</p>
4	xxxx042A	<p><b>Equate Symbol:</b> lwmRsnCodeSCHENVNoSystem:</p> <p><b>Meaning:</b> The scheduling environment exists however the specified system does not exist.</p> <p><b>Action:</b> Do not process work that depends upon the scheduling environment being available on the specified system.</p>
8	—	lwmRetCodeInvocError: Invalid invocation environment or parameters
8	xxxx0801	<p><b>Equate Symbol:</b> lwmRsnCodeSrbMode:</p> <p><b>Meaning:</b> Caller is in SRB mode.</p> <p><b>Action:</b> Avoid requesting this function while in SRB mode.</p>
8	xxxx0803	<p><b>Equate Symbol:</b> lwmRsnCodeDisabled</p> <p><b>Meaning:</b> Caller is disabled.</p> <p><b>Action:</b> Avoid requesting this function while disabled.</p>
8	xxxx0804	<p><b>Equate Symbol:</b> lwmRsnCodeLocked</p> <p><b>Meaning:</b> Caller is locked.</p> <p><b>Action:</b> Avoid requesting this function while locked.</p>
8	xxxx080B	<p><b>Equate Symbol:</b> lwmRsnCodeBadPI</p> <p><b>Meaning:</b> Error accessing parameter list.</p> <p><b>Action:</b> Check for possible storage overlay.</p>
8	xxxx0824	<p><b>Equate Symbol:</b> lwmRsnCodeAmode24</p> <p><b>Meaning:</b> Caller in 24 bit addressing mode.</p> <p><b>Action:</b> Request this function in 31 bit addressing mode.</p>

## IWMSEDES Macro

Table 55. Return and Reason Codes for the IWMSEDES Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0825	<b>Equate Symbol:</b> IwmRsnCodeAscModeNotPrimary <b>Meaning:</b> Caller invoked service but was not in primary ASC mode. <b>Action:</b> Request this function in primary mode.
8	xxxx0827	<b>Equate Symbol:</b> IwmRsnCodeRsvdNot0 <b>Meaning:</b> Reserved field in parameter list was non-zero. <b>Action:</b> Request this function with reserved fields zero.
8	xxxx0828	<b>Equate Symbol:</b> IwmRsnCodeBadVersion <b>Meaning:</b> Version number in parameter list or version length field is not valid. <b>Action:</b> Request this function with corrent version number.
10	—	<b>Equate Symbol:</b> IwmRetCodeCompError <b>Meaning:</b> Component error. <b>Action:</b> Contact your system programmer.

---

## Example

To determine if the scheduling environment at location ENVNAME is available on the system name at location SYSNAME specify:

```
IWMSEDES SCHENV=ENVNAME,  
          SYSTEM_NAME=SYSNAME,  
          RETCODE=RETCODE,  
          RSNCODE=RSNCODE  
  
*  
* Storage areas  
*  
ENVNAME  DS   CL16      Scheduling environment name  
SYSNAME  DS   CL8       Name of system  
RETCODE  DS    1F       Return code  
RSNCODE  DS    1F       Reason code
```

---

## Chapter 49. IWMSEQR – Scheduling Environments Query Service

IWMSEQR returns information about the scheduling environments and resources that are defined in the active service policy. The information includes the current state of each scheduling environment and resource on the current system and on other systems in the sysplex.

The information is obtained without serialization. Data may not be available for all systems in the sysplex.

The information is returned in a work area that you specify. The work area must be located in the caller's primary address space. The format of the work area is mapped by the IWMSET macro. IWMSEQR checks if the specified work area length (ANSLEN) is large enough to receive the output. If so, IWMSEQR returns the information in the work area and returns the actual length of the information in the QUERYLEN parameter. If the storage provided is not large enough, the caller receives reason code 040A, and IWMSEQR returns the required amount of storage in the QUERYLEN parameter.

---

### Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Supervisor state or program key mask (PKM) allowing keys 0-7.
<b>Dispatchable unit mode:</b>	Task
<b>Cross memory mode:</b>	Any PASN, any HASN, any SASN
<b>AMODE:</b>	31-bit
<b>ASC mode:</b>	Primary
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks may be held.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

---

### Programming Requirements

1. The macro CVT must be included to use this macro.
2. The macro IWMYCON must be included to use this macro.
3. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
4. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, IWMSNCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.

---

### Restriction

The caller cannot have an EUT FRR established.

---

## Input Register Information

Before issuing the IWMSEQRY macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

---

## Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Reason code if GR15 return code is non-zero
1	Used as work registers by the system
2-13	Unchanged
14	Used as work registers by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

---

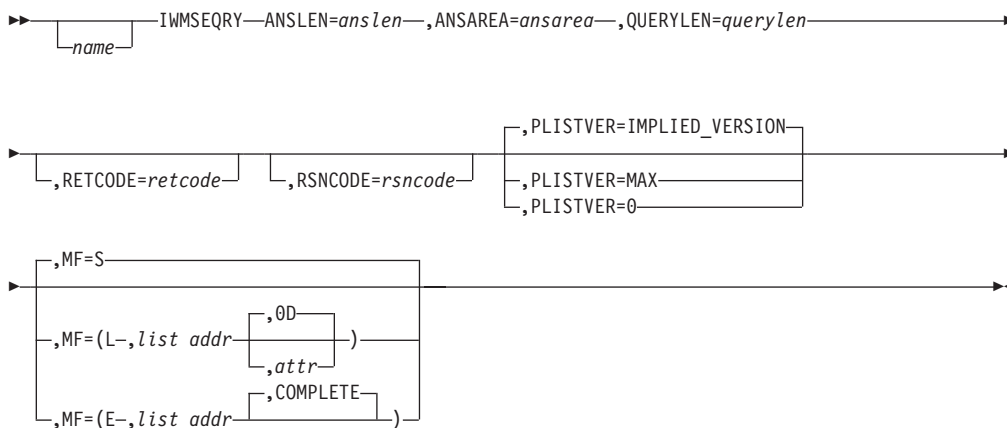
## Performance Implications

None.

---

## Syntax

### main diagram




---

## Parameters

The parameters are explained as follows:

*name*

An optional symbol, starting in column 1, that is the name on the IWMSEQRY macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

**,ANSAREA=ansarea**

A required input/output parameter, of an area to contain the data returned by IWMSEQRY. The area is mapped by macro IWMSET.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a field.

**ANSLEN=anslen**

A required input parameter, which contains the length of the area provided to contain the data returned by IWMSEQRY.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

**,MF=S****,MF=(L,list addr)****,MF=(L,list addr,attr)****,MF=(L,list addr,0D)****,MF=(E,list addr)****,MF=(E,list addr,COMPLETE)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

*,list addr*

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

*,attr*

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

**,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

**,PLISTVER=IMPLIED\_VERSION****,PLISTVER=MAX****,PLISTVER=0**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When

## IWMSEQRY Macro

using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form; in this way, MAX ensures that the parameter list does not overwrite nearby storage.

- **0**, if you use the currently available parameters.

**To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX
- A decimal value of 0

### **,QUERYLEN=***querylen*

A required output parameter, which contains the number of bytes needed to contain the scheduling environment information.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

### **,RETCODE=***retcode*

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

### **,RSNCODE=***rsncode*

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

---

## ABEND Codes

None.

---

## Return and Reason Codes

When the IWMSEQRY macro returns control to your program:

- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, if you coded RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 56. Return and Reason Codes for the IWMSEQRY Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	IwmRetCodeOk: Successful completion. All requested data returned.



Table 56. Return and Reason Codes for the IWMSEQRY Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
4	—	IwmRetCodeWarning: Successful completion, unusual conditions noted.
4	xxxx040A	<b>Equate Symbol:</b> IwmRsnCodeOutputAreaTooSmall: <b>Meaning:</b> The output area supplied is too small to receive all the available information. <b>Action:</b> Obtain a new output area using the length returned in the QUERYLEN parameter and invoke the service again.
4	xxxx0425	<b>Equate Symbol:</b> IwmRsnCodeNoSCHENV: <b>Meaning:</b> The system does not support scheduling environments services. This return code is set for releases prior to OS/390 Release 4. <b>Action:</b> Avoid requesting this function on releases prior to OS/390 Release 4.
4	xxxx0428	<b>Equate Symbol:</b> IwmRsnCodeNoSCHENVDefined <b>Meaning:</b> No scheduling environments or resources are defined in the active service policy. No data is returned in the output area. <b>Action:</b> Do not use the output area.
8	—	IwmRetCodeInvocError: Invalid invocation environment or parameters
8	xxxx0801	<b>Equate Symbol:</b> IwmRsnCodeSrbMode: <b>Meaning:</b> Caller is in SRB mode. <b>Action:</b> Avoid requesting this function while in SRB mode.
8	xxxx0803	<b>Equate Symbol:</b> IwmRsnCodeDisabled <b>Meaning:</b> Caller is disabled. <b>Action:</b> Avoid requesting this function while disabled.
8	xxxx0804	<b>Equate Symbol:</b> IwmRsnCodeLocked <b>Meaning:</b> Caller is locked. <b>Action:</b> Avoid requesting this function while locked.
8	xxxx080B	<b>Equate Symbol:</b> IwmRsnCodeBadPI <b>Meaning:</b> Error accessing parameter list. <b>Action:</b> Check for possible storage overlay.
8	xxxx0810	<b>Equate Symbol:</b> IwmRsnCodeEutFrr <b>Meaning:</b> Caller has an EUT FRR established. <b>Action:</b> Avoid requesting this function with an EUT FRR set.
8	xxxx0824	<b>Equate Symbol:</b> IwmRsnCodeAmode24 <b>Meaning:</b> Caller is in 24 bit addressing mode. <b>Action:</b> Request this function in 31 bit addressing mode.
8	xxxx0825	<b>Equate Symbol:</b> IwmRsnCodeAscModeNotPrimary <b>Meaning:</b> Caller invoked service but was not in primary ASC mode. <b>Action:</b> Request this function in primary mode.

## IWMSEQRY Macro

Table 56. Return and Reason Codes for the IWMSEQRY Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0827	<b>Equate Symbol:</b> IwmRsnCodeRsvdNot0 <b>Meaning:</b> Reserved field in parameter list was non-zero. <b>Action:</b> Request this function with reserved fields zero.
C	—	<b>Equate Symbol:</b> IwmRetCodeEnvError <b>Meaning:</b> Environmental error.
C	xxxx0C2C	<b>Equate Symbol:</b> IwmRsnCodeCannotAccessPolicy <b>Meaning:</b> The service cannot access the active policy possibly due to a policy activation in progress. <b>Action:</b> The caller can try the service again later, or return an error indication to its caller.
10	—	<b>Equate Symbol:</b> IwmRetCodeCompError <b>Meaning:</b> Component error. <b>Action:</b> Contact your system programmer.

---

## Example

To store scheduling environment and resource information into an area whose address is in register 5 and whose length is at location ANSLLEN:

```
IWMSEQRY ANSAREA=(R5),
          ANSLLEN=ANSLLEN,
          QUERYLEN=RQDLEN,
          RETCODE=RETCODE,
          RSNCODE=RSNCODE

*
* Storage areas
*
ANSLEN   DS   1F           Answer area length
RQDLEN   DS   1F           Required length
RETCODE  DS   1F           Return code
RSNCODE  DS   1F           Reason code
*
```

---

## Chapter 50. IWMSESET – Scheduling Environments Set Resource

IWMSESET allows the caller to modify the state of a resource. A resource is an abstract element that can represent an actual physical entity (such as a peripheral device), or an intangible quality (such as a certain time of day). A resource has an ON, OFF, or RESET state.

A resource is a component of a scheduling environment. A scheduling environment is a list of resource names and their required states. By modifying the state of a resource you can:

- change a scheduling environment such that the resources are in the required state thereby allowing work to be scheduled,
- change a scheduling environment such that the resources are not in the required state thereby preventing work from being scheduled.

---

### Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Supervisor state or program key mask (PKM) allowing keys 0-7.
<b>Dispatchable unit mode:</b>	Task
<b>Cross memory mode:</b>	Any PASN, any HASN, any SASN
<b>AMODE:</b>	31-bit
<b>ASC mode:</b>	Primary
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks may be held.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

---

### Programming Requirements

1. The macro CVT must be included to use this macro.
2. The macro IWMYCON must be included to use this macro.
3. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
4. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, IWMRSNCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.

---

### Restrictions

None.

---

### Input Register Information

Before issuing the IWMSESET macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

---

## Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Reason code if GR15 return code is non-zero
1	Used as work registers by the system
2-13	Unchanged
14	Used as work registers by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

---

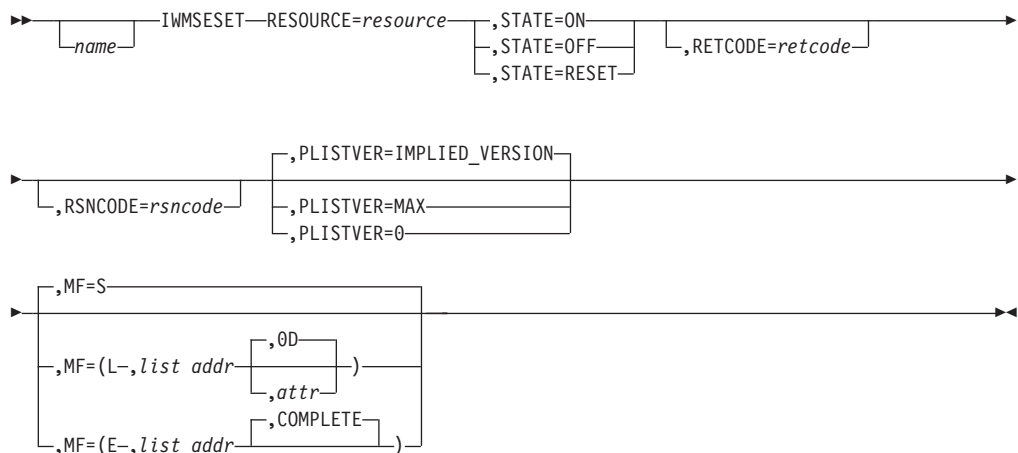
## Performance Implications

None.

---

## Syntax

### main diagram




---

## Parameters

The parameters are explained as follows:

### *name*

An optional symbol, starting in column 1, that is the name on the IWMSESET macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

### **,MF=S**

,MF=(L,*list addr*)  
 ,MF=(L,*list addr,attr*)  
 ,MF=(L,*list addr,0D*)  
 ,MF=(E,*list addr*)  
 ,MF=(E,*list addr,COMPLETE*)

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

,*list addr*

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

,*attr*

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

,**COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

,**PLISTVER=IMPLIED\_VERSION**

,**PLISTVER=MAX**

,**PLISTVER=0**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form; in this way, MAX ensures that the parameter list does not overwrite nearby storage.

- **0**, if you use the currently available parameters.

**To code:** Specify one of the following:

## IWMSESET Macro

- IMPLIED\_VERSION
- MAX
- A decimal value of 0

### **RESOURCE=resource**

A required input parameter, which contains the resource name to be modified.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 16-character field.

### **,RETCODE=retcode**

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

### **,RSNCODE=rsncode**

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

### **,STATE=ON**

### **,STATE=OFF**

### **,STATE=RESET**

A required parameter, which sets the state of the resource.

### **,STATE=ON**

sets the resource to the ON state.

### **,STATE=OFF**

sets the resource to the OFF state.

### **,STATE=RESET**

sets the resource to the RESET state.

---

## ABEND Codes

None.

---

## Return and Reason Codes

When the IWMSESET macro returns control to your program:

- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, if you coded RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 57. Return and Reason Codes for the IWMSESET Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	IwmRetCodeOk: Successful completion. All requested data returned.
4	—	IwmRetCodeWarning: Successful completion, unusual conditions noted.

Table 57. Return and Reason Codes for the IWMSESET Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
4	xxxx0425	<p><b>Equate Symbol:</b> lwmRsnCodeNoSCHENV:</p> <p><b>Meaning:</b> The system does not support scheduling environments services. This return code is set for releases prior to OS/390 Release 4.</p> <p><b>Action:</b> Avoid requesting this function on releases prior to OS/390 Release 4.</p>
4	xxxx0429	<p><b>Equate Symbol:</b> lwmRsnCodeResourceNotFound:</p> <p><b>Meaning:</b> The resource specified by RESOURCE does not exist.</p> <p><b>Action:</b> Check the specification of the RESOURCE parameter. If the RESOURCE parameter is correct, check whether the resource is defined in the active service policy.</p>
8	—	lwmRetCodeInvocError: Invalid invocation environment or parameters
8	xxxx0801	<p><b>Equate Symbol:</b> lwmRsnCodeSrbMode:</p> <p><b>Meaning:</b> Caller is in SRB mode.</p> <p><b>Action:</b> Avoid requesting this function while in SRB mode.</p>
8	xxxx0803	<p><b>Equate Symbol:</b> lwmRsnCodeDisabled</p> <p><b>Meaning:</b> Caller is disabled.</p> <p><b>Action:</b> Avoid requesting this function while disabled.</p>
8	xxxx0804	<p><b>Equate Symbol:</b> lwmRsnCodeLocked</p> <p><b>Meaning:</b> Caller is locked.</p> <p><b>Action:</b> Avoid requesting this function while locked.</p>
8	xxxx080B	<p><b>Equate Symbol:</b> lwmRsnCodeBadPI</p> <p><b>Meaning:</b> Error accessing parameter list.</p> <p><b>Action:</b> Check for possible storage overlay.</p>
8	xxxx0824	<p><b>Equate Symbol:</b> lwmRsnCodeAmode24</p> <p><b>Meaning:</b> Caller is in 24 bit addressing mode.</p> <p><b>Action:</b> Request this function in 31 bit addressing mode.</p>
8	xxxx0825	<p><b>Equate Symbol:</b> lwmRsnCodeAscModeNotPrimary</p> <p><b>Meaning:</b> Caller invoked service but was not in primary ASC mode.</p> <p><b>Action:</b> Request this function in primary mode.</p>
8	xxxx0827	<p><b>Equate Symbol:</b> lwmRsnCodeRsvdNot0</p> <p><b>Meaning:</b> Reserved field in parameter list was non-zero.</p> <p><b>Action:</b> Request this function with reserved fields zero.</p>
8	xxxx0828	<p><b>Equate Symbol:</b> lwmRsnCodeBadVersion</p> <p><b>Meaning:</b> Version number in parameter list or version length field is not valid.</p> <p><b>Action:</b> Request this function with correct version number.</p>

## IWMSESET Macro

Table 57. Return and Reason Codes for the IWMSESET Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
10	—	<b>Equate Symbol:</b> IwmRetCodeCompError <b>Meaning:</b> Component error. <b>Action:</b> Contact your system programmer.

---

## Example

To set the resource specified at location RESNAME to the ON state specify:

```
IWMSESET RESOURCE=RESNAME,  
          STATE=ON,  
          RETCODE=RETCODE,  
          RSNCODE=RSNCODE  
*  
* Storage areas  
*  
RESNAME  DS   CL16           Resource name  
RETCODE  DS   1F            Return code  
RSNCODE  DS   1F            Reason code  
*
```



---

## Chapter 51. IWMSEVAL – Scheduling Environments Validate Service

The IWMSEVAL service validates a scheduling environment name. The caller can validate a scheduling environment prior to associating it with a work item (such as a job or transaction).

If the scheduling environment is valid, the caller receives return code `IwmRetCodeOK`. If the scheduling environment is not valid, the caller receives return code `IwmRetCodeWarning` and reason code `IwmRsnCodeSCHENVNotFound`.

---

### Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Supervisor state or program key mask (PKM) allowing keys 0-7.
<b>Dispatchable unit mode:</b>	Task
<b>Cross memory mode:</b>	Any PASN, any HASN, any SASN
<b>AMODE:</b>	31-bit
<b>ASC mode:</b>	Primary
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks may be held.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

---

### Programming Requirements

1. The macro `CVT` must be included to use this macro.
2. The macro `IWMYCON` must be included to use this macro.
3. The macro `IWMPB` must be in the library concatenation, since it is included by `IWMYCON`.
4. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, `IWMRSNCODE_MASK_CONST` defined in `IWMYCON`, may be used for this purpose.

---

### Restrictions

None.

---

### Input Register Information

Before issuing the `IWMSEVAL` macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

---

## Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Reason code if GR15 return code is non-zero
1	Used as work registers by the system
2-13	Unchanged
14	Used as work registers by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

---

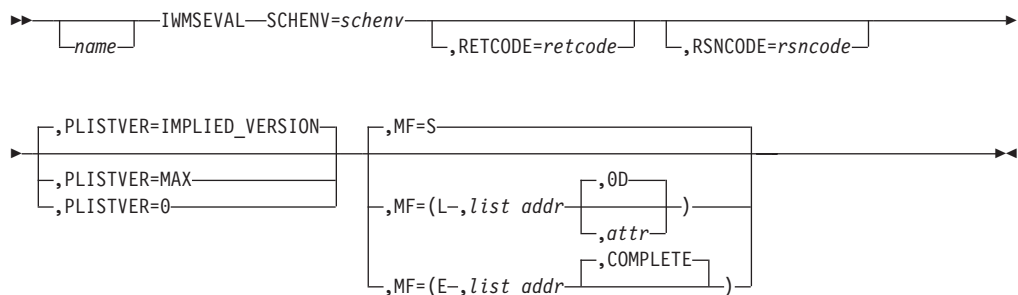
## Performance Implications

None.

---

## Syntax

### main diagram




---

## Parameters

The parameters are explained as follows:

### *name*

An optional symbol, starting in column 1, that is the name on the IWMSEVAL macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

### **,MF=S**

**,MF=(L,list addr)**

**,MF=(L,list addr,attr)**

**,MF=(L,list addr,0D)**

**,MF=(E,list addr)**

**,MF=(E,*list addr*,COMPLETE)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

**,*list addr***

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

**,*attr***

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

**,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

**,PLISTVER=IMPLIED\_VERSION****,PLISTVER=MAX****,PLISTVER=0**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form; in this way, MAX ensures that the parameter list does not overwrite nearby storage.

- **0**, if you use the currently available parameters.

**To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX
- A decimal value of 0

## IWMSEVAL Macro

### **,RETCODE=***retcode*

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

### **,RSNCODE=***rsncode*

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

### **SCHENV=***schenv*

A required input parameter, which contains the scheduling environment to be validated.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 16-character field.

---

## ABEND Codes

None.

---

## Return and Reason Codes

When the IWMSEVAL macro returns control to your program:

- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, if you coded RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 58. Return and Reason Codes for the IWMSEVAL Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<code>IwmRetCodeOk</code> : Successful completion.
4	—	<code>IwmRetCodeWarning</code> : Successful completion, unusual conditions noted.
4	xxxx0425	<b>Equate Symbol:</b> <code>IwmRsnCodeNoSCHENV</code> <b>Meaning:</b> The system does not support scheduling environments services. This return code is set for releases prior to OS/390 Release 4. <b>Action:</b> Avoid requesting this function on releases prior to OS/390 Release 4.
4	xxxx0426	<b>Equate Symbol:</b> <code>IwmRsnCodeSCHENVNotFound</code> <b>Meaning:</b> The scheduling environment specified by SCHENV does not exist. <b>Action:</b> Check the specification of the SCHENV parameter. If the SCHENV parameter is correct, check whether the scheduling environment is defined in the active service policy.
8	—	<code>IwmRetCodeInvocError</code> : Invalid invocation environment or parameters

Table 58. Return and Reason Codes for the IWMSEVAL Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0801	<b>Equate Symbol:</b> lwmRsnCodeSrbMode: <b>Meaning:</b> Caller is in SRB mode. <b>Action:</b> Avoid requesting this function while in SRB mode.
8	xxxx0803	<b>Equate Symbol:</b> lwmRsnCodeDisabled <b>Meaning:</b> Caller is disabled. <b>Action:</b> Avoid requesting this function while disabled.
8	xxxx0804	<b>Equate Symbol:</b> lwmRsnCodeLocked <b>Meaning:</b> Caller is locked. <b>Action:</b> Avoid requesting this function while locked.
8	xxxx080B	<b>Equate Symbol:</b> lwmRsnCodeBadPI <b>Meaning:</b> Error accessing parameter list. <b>Action:</b> Check for possible storage overlay.
8	xxxx0824	<b>Equate Symbol:</b> lwmRsnCodeAmode24 <b>Meaning:</b> Caller is in 24 bit addressing mode. <b>Action:</b> Request this function in 31 bit addressing mode.
8	xxxx0825	<b>Equate Symbol:</b> lwmRsnCodeAscModeNotPrimary <b>Meaning:</b> Caller invoked service but was not in primary ASC mode. <b>Action:</b> Request this function in primary mode.
8	xxxx0827	<b>Equate Symbol:</b> lwmRsnCodeRsvdNot0 <b>Meaning:</b> Reserved field in parameter list was non-zero. <b>Action:</b> Request this function with reserved fields zero.
8	xxxx0828	<b>Equate Symbol:</b> lwmRsnCodeBadVersion <b>Meaning:</b> Version number in parameter list or version length field is not valid. <b>Action:</b> Request this function with corrent version number.
10	—	<b>Equate Symbol:</b> lwmRetCodeCompError <b>Meaning:</b> Component error. <b>Action:</b> Contact your system programmer.

---

## Example

To validate the scheduling environment name at location ENVNAME specify:

```

IWMSEVAL SCHENV=ENVNAME,
          RETCODE=RETCODE,
          RSNCODE=RSNCODE
*
* Storage areas
*
ENVNAME  DS   CL16           Scheduling environment name
RETCODE  DS   1F             Return code
RSNCODE  DS   1F             Reason code

```



---

## Chapter 52. IWMSINF — WLM Server Manager Inform Service

The IWMSINF service should be used to obtain the number of server instances to be started from WLM. The caller must have previously connected to WLM using the IWM4CON service specifying SERVER\_MANAGER=YES, SERVER\_TYPE=QUEUE and MANAGE\_TASKS=YES.

The caller can use the service in the following ways:

1. **MODE=SUSPEND**

The calling task is suspended until WLM wants the caller to start additional server instances. The caller must re-invoke the service after it starts the server instances to wait for the next notification. The caller cannot rely upon asynchronous exits receiving control while the task is suspended.

2. **MODE=POST**

The calling task is not suspended by WLM. WLM returns the number of additional server instances to start now. WLM will post the caller's ECB when the number of server instances should be increased. After the ECB is posted the caller must re-invoke IWMSINF to obtain the number of server instances to start.

3. **MODE=INFORM**

The calling task is not suspended by WLM. WLM returns the number of additional server instances to start now. This form should only be used the first time the service is invoked immediately after connect because at this time the service will always return a value.

4. **MODE=ECBCANCEL**

The calling task is not suspended by WLM. This invocation should only be used to inform WLM that it should not post a caller's ECB any more. The form can be used for error recovery purposes in conjunction with MODE=POST.

WLM stops server instances by returning reason code `IwmRsnCodeStopTask` from IWM4SSL.

---

### Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Problem state. Any PSW key
<b>Dispatchable unit mode:</b>	Task
<b>Cross memory mode:</b>	PASN=HASN=SASN
<b>AMODE:</b>	31-bit
<b>ASC mode:</b>	Primary
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks may be held.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

---

### Programming Requirements

1. Make sure no EUT FRRs are established.
2. The macro CVT must be included to use this macro.
3. The macro IWMYCON must be included to use this macro.

## IWMSINF Macro

4. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
5. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, IWMSRNCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.

---

## Restrictions

1. This macro may not be used during task/address space termination.
2. Only a single invocation is allowed to be active for a given address space at any given time.
3. Before using this macro the caller must connect to WLM via IWM4CON Server\_Manager=YES, Server\_Type=Queue, Manage\_Tasks=Yes.

---

## Input Register Information

Before issuing the IWMSINF macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

---

## Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Reason code if GR15 return code is non-zero
1	Used as work registers by the system
2-13	Unchanged
14	Used as work registers by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

---

## Performance Implications

None.

---

## Syntax

main diagram





## Parameters

The parameters are explained as follows:

### *name*

An optional symbol, starting in column 1, that is the name on the IWMSINF macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

### **,ECB=ecb**

### **,ECB=NO\_ECB**

When **MODE=POST** is specified, an optional input parameter, to specify the ECB which should be posted if WLM wants to inform the caller that the number of server instances have changed. The caller re-invokes IWMSINF after the ECB was posted by WLM to obtain number of server instances to start. The default is **NO\_ECB**. indicates that no ECB has been specified by the Caller

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

### **,MF=S**

### **,MF=(L,list addr)**

### **,MF=(L,list addr,attr)**

### **,MF=(L,list addr,OD)**

### **,MF=(E,list addr)**

### **,MF=(E,list addr,COMPLETE)**

An optional input parameter that specifies the macro form.

Use **MF=S** to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. **MF=S** is the default.

Use **MF=L** to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The

## IWMSINF Macro

list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

*,list addr*

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

*,attr*

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

**,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

**MODE=SUSPEND**

**MODE=POST**

**MODE=INFORM**

**MODE=ECBCANCEL**

A required parameter that indicates how the caller uses the service

**MODE=SUSPEND**

indicates that the caller wants to get suspended by WLM to listen for additional server instances to start. WLM will resume the caller if the number of server instances should be increased.

**MODE=POST**

indicates that the caller wants to get posted if additional server instances should be started. If the caller gets posted it must re-invoke IWMSINF to obtain the value.

The caller is not suspended by WLM and WLM will also return the number of additional server instances on this call.

**MODE=INFORM**

indicates that the caller wants to obtain the number of server instances without being suspended by WLM. This form is useful during initialization after IWM4CON if the caller wants to use the POST form but is not able to provide an ECB yet. It is expected that the caller will provide an ECB on the next invocation of the service.

**MODE=ECBCANCEL**

indicates that a caller who passed an ECB to WLM wants to cancel the ECB address to avoid being posted in the future by WLM.

This form is useful during termination or recovery of server address spaces if a previous ECB address is no longer valid.

**,PLISTVER=IMPLIED\_VERSION**

**,PLISTVER=MAX**

**,PLISTVER=0**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an

optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form, when both are assembled with the same level of the system. In this way, MAX ensures that the parameter list does not overwrite nearby storage.

- **0**, if you use the currently available parameters.

**To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX
- A decimal value of 0

**,RETCODE=retcode**

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,RSNCODE=rsncode**

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,SRVINST\_TO\_STRT=svinst\_to\_strt**

When MODE=SUSPEND is specified, a required output parameter that returns the number of server instances to start by the caller

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

**,SRVINST\_TO\_STRT=svinst\_to\_strt**

When MODE=POST is specified, a required output parameter that returns the number of server instances to start by the caller

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

**,SRVINST\_TO\_STRT=svinst\_to\_strt**

When MODE=INFORM is specified, a required output parameter that returns the number of server instances to start by the caller

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

---

## ABEND Codes

None.

## Return and Reason Codes

When the IWMSINF macro returns control to your program:

- GPR 15 (and *retcode*, when you code RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, when you code RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 59. Return and Reason Codes for the IWMSINF Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> <code>IwmRetCodeOk</code> <b>Meaning:</b> Successful completion. <b>Action:</b> None required.
8	—	<b>Equate Symbol:</b> <code>IwmRetCodeInvocError</code> <b>Meaning:</b> Invalid invocation environment or parameters.
8	xxxx0801	<b>Equate Symbol:</b> <code>IwmRsnCodeSrbMode</code> <b>Meaning:</b> Caller is in SRB mode. <b>Action:</b> Avoid requesting this function while in SRB mode.
8	xxxx0803	<b>Equate Symbol:</b> <code>IwmRsnCodeDisabled</code> <b>Meaning:</b> Caller is disabled. <b>Action:</b> Avoid requesting this function while disabled.
8	xxxx0804	<b>Equate Symbol:</b> <code>IwmRsnCodeLocked</code> <b>Meaning:</b> Caller is locked. <b>Action:</b> Avoid requesting this function while locked.
8	xxxx080B	<b>Equate Symbol:</b> <code>IwmRsnCodeBadPI</code> <b>Meaning:</b> Error accessing parameter list. <b>Action:</b> Check for possible storage overlay.
8	xxxx0824	<b>Equate Symbol:</b> <code>IwmRsnCodeAmode24</code> <b>Meaning:</b> Caller invoked service but was in 24 bit addressing mode. <b>Action:</b> Request this function only when you are in 31 bit addressing mode.
8	xxxx0825	<b>Equate Symbol:</b> <code>IwmRsnCodeAscModeNotPrimary</code> <b>Meaning:</b> Caller invoked service but was not in primary ASC mode. <b>Action:</b> Avoid requesting this function in this environment.
8	xxxx0828	<b>Equate Symbol:</b> <code>IwmRsnCodeBadVersion</code> <b>Meaning:</b> Version number in parameter list is not valid. <b>Action:</b> Check for possible storage overlay of the parameter list.

Table 59. Return and Reason Codes for the IWM5INF Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0840	<p><b>Equate Symbol:</b> IwmRsnCodeServiceNotEnabled</p> <p><b>Meaning:</b> Caller's space connection is not enabled for this service</p> <p><b>Action:</b> Make sure that SERVER_MANAGER=YES, SERVER_TYPE=QUEUE and MANAGE_TASKS=YES is specified on the IWM4CON request to enable this service.</p>
8	xxxx0841	<p><b>Equate Symbol:</b> IwmRsnCodeXmemMode</p> <p><b>Meaning:</b> Caller is in cross-memory mode.</p> <p><b>Action:</b> Request this function only when you are not in cross-memory mode.</p>
8	xxxx0842	<p><b>Equate Symbol:</b> IwmRsnCodeNoWLMConnect</p> <p><b>Meaning:</b> Caller's space is not connected to WLM.</p> <p><b>Action:</b> Invoke the IWM4CON macro before invoking this macro.</p>
8	xxxx084D	<p><b>Equate Symbol:</b> IwmRsnCodeNotAuthConnect</p> <p><b>Meaning:</b> The caller must be supervisor state or have PSW key mask 0-7 authority to use the requested WLM service. This applies only if the caller uses the service with MODE=POST.</p> <p><b>Action:</b> Avoid requesting this function in this environment.</p>
8	xxxx087B	<p><b>Equate Symbol:</b> IwmRsnCodeUnexpectedCall</p> <p><b>Meaning:</b> The system didn't expected the caller to use this service</p> <p><b>Action:</b> Make sure that MANAGE_TASKS=YES is specified on the IWM4CON request.</p>
C	—	<p><b>Equate Symbol:</b> IwmRetCodeEnvError</p> <p><b>Meaning:</b> Environmental error.</p>
C	xxxx0C14	<p><b>Equate Symbol:</b> IwmRsnCodeNoWorkShutDown</p> <p><b>Meaning:</b> No work selected. Caller is to shutdown.</p> <p><b>Action:</b> Caller must disconnect by invoking the IWM4DIS macro.</p>
10	—	<p><b>Equate Symbol:</b> IwmRetCodeCompError</p> <p><b>Meaning:</b> Component error.</p> <p><b>Action:</b> Contact your system programmer.</p>

## Example

To obtain information about the number of server instances to start from WLM, specify the following:

```
IWM4CON WORK_MANAGER=YES,
        SERVER_MANAGER=YES,
        PARALLEL_EU=EUNITS,
        SERVER_TYPE=QUEUE,
        MANAGE_TASKS=YES,
        SERVER_LIMIT=MAXTASKS,
        CONNTKN=CTKN,CONNTKNKEY=PSWKEY,
```

## IWMSINF Macro

```
                RETCODE=RC,RSNCODE=RSN

IWMSINF MODE=SUSPEND,
        SRVINST_TO_STRT=NUMINST,
        RETCODE=RC,RSNCODE=RSN
*
* Storage areas
*
EUNITS  DS    F           Number of Tasks which will be started
*                               if the application environment is not managed.
MAXTASKS DS    F           Maximum Number of Tasks up to which
*                               WLM adjusts the number of server
*                               instances for the server AS
CTKN    DS    FL4         Connect Token
NUMINSTS DS    F           Number of server instances to start
RC      DS    F           Return code
RSN     DS    F           Reason code
```

---

## Chapter 53. IWMSRDNS – Get Sysplex Routing Location List

IWMSRDNS will return a list of location names for all registered servers which have been registered with a host name, known to the system on which the service is invoked. Servers which have deregistered, via IWMSRDRS, may still be present in the output list, due to the asynchronous nature of deregistration. Conversely, some registered servers may not appear for this same reason.

---

### Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Problem state with any PSW key.
<b>Dispatchable unit mode:</b>	Task
<b>Cross memory mode:</b>	Any PASN, any HASN, any SASN
<b>AMODE:</b>	31-bit
<b>ASC mode:</b>	Primary
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks may be held. FRRs may be established.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

---

### Programming Requirements

1. The macro CVT must be included to use this macro.
2. The macro IWMYCON must be included to use this macro.
3. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
4. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, IWMRSNCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.
5. All character data, unless otherwise specified, is assumed to be left justified and padded with blanks on the right, as needed, to occupy the specified number of bytes.

---

### Restrictions

This macro may only be used on OS/390 Version 2, Release 4 or equivalent.

---

### Input Register Information

Before issuing the IWMSRDNS macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

---

### Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
----------	----------

## IWMSRDNS Macro

0	Reason code if GR15 return code is non-zero
1	Used as work register by the system
2-13	Unchanged
14	Used as work register by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

---

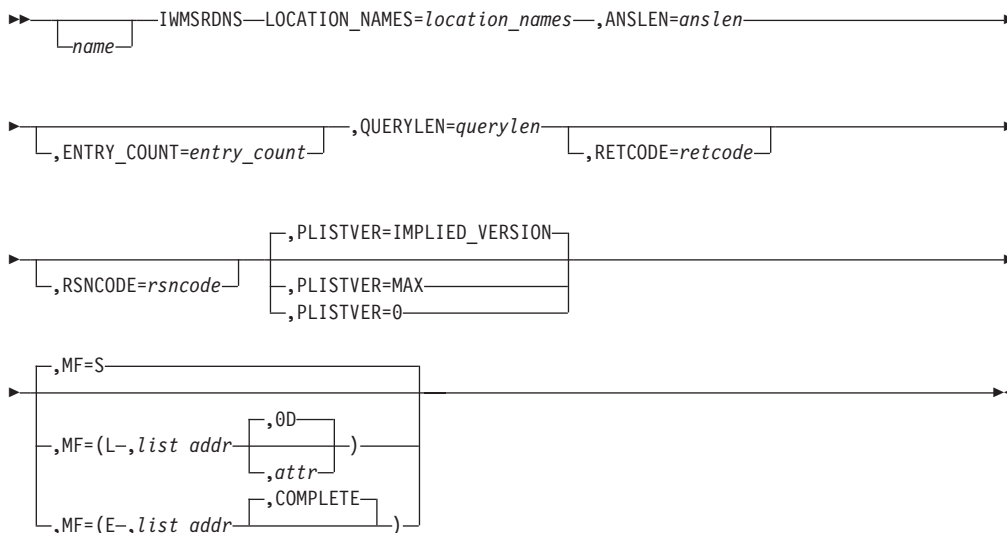
## Performance Implications

None.

---

## Syntax

### main diagram




---

## Parameters

The parameters are explained as follows:

### *name*

An optional symbol, starting in column 1, that is the name on the IWMSRDNS macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

### **,ANSLEN=anslen**

A required input parameter, which contains the length of the LOCATION\_NAMES in bytes.



**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

**,ENTRY\_COUNT=entry\_count**

An optional output parameter, which will hold the number of location entries returned by the service. This is the number of entries in the SYSL\_INFO array (see IWMWSYSL).

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

**LOCATION\_NAMES=location\_names**

A required input parameter, which specifies the name of the area to be filled in with the list of location names for the registered, active, LUs in the SYSPLEX registered with a host name.

The area must be large enough to contain at least 1 entry. The format of this area is mapped by IWMWSYSL.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a field.

**,MF=S**

**,MF=(L,list addr)**

**,MF=(L,list addr,attr)**

**,MF=(L,list addr,0D)**

**,MF=(E,list addr)**

**,MF=(E,list addr,COMPLETE)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

**,list addr**

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

**,attr**

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

**,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

**,PLISTVER=IMPLIED\_VERSION**

**,PLISTVER=MAX**

### **,PLISTVER=0**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form; in this way, MAX ensures that the parameter list does not overwrite nearby storage.

- **0**, if you use the currently available parameters.

**To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX
- A decimal value of 0

### **,QUERYLEN=*querylen***

A required output parameter, variable which contains the number of bytes needed for all data requested.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

### **,RETCODE=*retcode***

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

### **,RSNCODE=*rsncode***

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

---

## ABEND Codes

None.

---

## Return and Reason Codes

When the IWMSRDNS macro returns control to your program:

- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, if you coded RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 60. Return and Reason Codes for the IWMSRDNS Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> IwmRetCodeOk <b>Meaning:</b> Successful completion. <b>Action:</b> None required.
4	—	<b>Equate Symbol:</b> IwmRetCodeWarning <b>Meaning:</b> Successful completion, unusual conditions noted.
4	xxxx040A	<b>Equate Symbol:</b> IwmRsnCodeOutputAreaTooSmall <b>Meaning:</b> The output area supplied is too small to receive all the available information. <b>Action:</b> None required. If necessary, reinvoke the service with an output area of sufficient size to receive all information.
4	xxxx040B	<b>Equate Symbol:</b> IwmRsnCodeNoServersRegistered <b>Meaning:</b> No Servers have registered in the sysplex. <b>Action:</b> None required.
8	—	<b>Equate Symbol:</b> IwmRetCodeInvocError <b>Meaning:</b> Invalid invocation environment or parameters.
8	xxxx0803	<b>Equate Symbol:</b> IwmRsnCodeDisabled <b>Meaning:</b> Caller is disabled. <b>Action:</b> Avoid requesting this function while disabled.
8	xxxx0804	<b>Equate Symbol:</b> IwmRsnCodeLocked <b>Meaning:</b> Caller is locked. <b>Action:</b> Avoid requesting this function while locked.
8	xxxx080B	<b>Equate Symbol:</b> IwmRsnCodeBadPI <b>Meaning:</b> Error accessing parameter list. <b>Action:</b> Check for possible storage overlay.
8	xxxx0823	<b>Equate Symbol:</b> IwmRsnCodeDatoff <b>Meaning:</b> Caller invoked service while DATOFF <b>Action:</b> Avoid requesting this function in this environment.
8	xxxx0824	<b>Equate Symbol:</b> IwmRsnCodeAmode24 <b>Meaning:</b> Caller invoked service but was in 24 bit addressing mode. <b>Action:</b> Request this function only when you are in 31 bit addressing mode.
8	xxxx0825	<b>Equate Symbol:</b> IwmRsnCodeAscModeNotPrimary <b>Meaning:</b> Caller invoked service but was not DAT on Primary ASC mode. <b>Action:</b> Avoid requesting this function in this environment.
C	—	<b>Equate Symbol:</b> IwmRetCodeEnvError <b>Meaning:</b> Environmental error.

## IWMSRDNS Macro

Table 60. Return and Reason Codes for the IWMSRDNS Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
C	xxxx0C01	<b>Equate Symbol:</b> lwmRsnCodeNoStg <b>Meaning:</b> Storage is not available for the request. <b>Action:</b> There is a storage shortage. The function may work successfully at a later time.
10	—	<b>Equate Symbol:</b> lwmRetCodeCompError <b>Meaning:</b> Component error. <b>Action:</b> No action required. The function may be successful if invoked again.

---

## Example

To list all locations registered with a host name:

```
        IWMSRDNS LOCATION_NAMES=DATA,
              ANSLLEN=SIZE,
              ENTRY_COUNT=E,
              QUERYLEN=Q,
              RETCODE=RC,RSNCODE=RSN
*
* Storage areas
*
DATA    DS    CL200           Area to receive output
SIZEEQU EQU  *-DATA         Equate for size of Data
E       DS    F             Field to receive entry count
Q       DS    F             Field to receive query length
RC      DS    F             Return code
RSN     DS    F             Reason code
SIZE    DC    A(SIZEEQU)    Field to hold size
```

---

## Chapter 54. IWMSRDRS – Deregister a Server for Sysplex Routing

IWMSRDRS will deregister a server that had previously registered via IWMSRSRG, the Sysplex Router Registration macro, for sysplex workload balancing. Deregistration removes the specified server as a candidate from the Sysplex Routing Selection service, IWMSRSRS. Since the propagation of the deregistration to other systems is asynchronous, a newly deregistered triplet will continue to be eligible for selection by other systems for a period of time after return from the IWMSRDRS invocation. If the server was registered with a host name the caller must provide the host name in order to deregister the server.

### Important Note

With APAR OW33878 installed, new security checks are added to the module this service invokes (as well as to the register module invoked by IWMSRSRG). With this APAR installed, the security checks are invoked when the caller is in Problem state, and does *not* own any privileged keys (i.e. has none of the authorization key mask bits 0-7 turned on). In this case, the caller must satisfy the following three conditions to use this service:

- The caller must *not* be in cross memory mode
- The caller must *not* have an EUT FRR established
- The caller must be authorized in the BPX.WLMSEVER profile in the facility class.

---

## Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Problem state with any PSW key (see previous note).
<b>Dispatchable unit mode:</b>	Task or SRB
<b>Cross memory mode:</b>	Any PASN, any HASN, any SASN (see previous note).
<b>AMODE:</b>	31-bit
<b>ASC mode:</b>	Primary
<b>Interrupt status:</b>	Unlocked. FRRs may be established (see previous note).
<b>Locks:</b>	No locks may be held.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

---

## Programming Requirements

None.

---

## Restrictions

1. This macro may not be used prior to the completion of WLM address space initialization
2. All parameter areas must reside in current primary.

## Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Reason code if GR15 return code is non-zero
1	Used as a work register by the macro
14	Used as a work register by the macro
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0	Used as a work register by the macro
1	Used as a work register by the macro
14	Used as a work register by the macro
15	Used as a work register by the macro

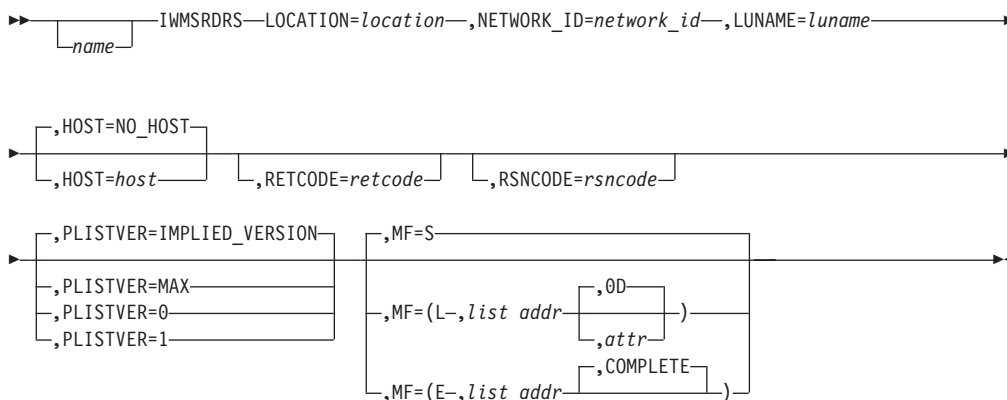
Some callers depend on register contents remaining the same before and after using a service. If the system changes the contents of registers on which the caller depends, the caller must save them before calling the service, and restore them after the system returns control.

## Performance Implications

None.

## Syntax

### main diagram



## Parameters

The parameters are explained as follows:

### *name*

An optional symbol, starting in column 1, that is the name on the IWMSRDRS macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

### ,HOST=*host*

### ,HOST=NO\_HOST

An optional input parameter, which contains the server HOST name associated

with the address space to be deregistered. The value should be padded on the right with blanks for any unused characters. The default is NO\_HOST, which indicates that a HOST name was not passed.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 64-character field.

**LOCATION=***location*

A required input parameter, which contains the server LOCATION associated with the registered address space.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 18-character field.

**,LUNAME=***luname*

A required input parameter, which contains the server Logical Unit name associated with the registered address space.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

**,MF=S**

**,MF=(L,***list addr***)**

**,MF=(L,***list addr***,***attr***)**

**,MF=(L,***list addr***,0D)**

**,MF=(E,***list addr***)**

**,MF=(E,***list addr***,COMPLETE)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

**,list addr**

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

**,attr**

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

**,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

**,NETWORK\_ID=***network\_id*

A required input parameter, which contains the Network ID associated with the registered address space.

## IWMSRDRS Macro

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

**,PLISTVER=IMPLIED\_VERSION**

**,PLISTVER=MAX**

**,PLISTVER=0**

**,PLISTVER=1**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form; in this way, MAX ensures that the parameter list does not overwrite nearby storage.

- **0**, if you use only the following parameters:

LOCATION	LUNAME	NETWORK_ID
----------	--------	------------

- **1**, if you use any of the following parameters, and parameters from version 0, or both:

HOST
------

**To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX
- A decimal value of 0 or 1

**,RETCODE=*retcode***

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,RSNCODE=*rsncode***

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

---

## ABEND Codes

None.

---

## Return and Reason Codes

When the IWMSRDRS macro returns control to your program:

- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.



- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, if you coded RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes. IBM support personnel may request the entire reason code, including the **xxxx** value.

*Table 61. Return and Reason Codes for the IWMSRDRS Macro*

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> <code>IwmRetCodeOk</code> <b>Meaning:</b> Successful completion.
4	—	<b>Equate Symbol:</b> <code>IwmRetCodeWarning</code> <b>Meaning:</b> Successful completion, unusual conditions noted.
4	xxxx0418	<b>Equate Symbol:</b> <code>IwmRsnCodeServerNotRegistered</code> <b>Meaning:</b> Server not registered
8	—	<b>Equate Symbol:</b> <code>IwmRetCodeInvocError</code> <b>Meaning:</b> Invalid invocation environment or parameters
8	xxxx0803	<b>Equate Symbol:</b> <code>IwmRsnCodeDisabled</code> <b>Meaning:</b> Caller is disabled.
8	xxxx0804	<b>Equate Symbol:</b> <code>IwmRsnCodeLocked</code> <b>Meaning:</b> Caller is locked.
8	xxxx080B	<b>Equate Symbol:</b> <code>IwmRsnCodeBadPI</code> <b>Meaning:</b> Error accessing parameter list
8	xxxx0823	<b>Equate Symbol:</b> <code>IwmRsnCodeDatoff</code> <b>Meaning:</b> Caller invoked service while DATOFF
8	xxxx0824	<b>Equate Symbol:</b> <code>IwmRsnCodeAmode24</code> <b>Meaning:</b> Caller invoked service but was in 24 bit addressing mode.
8	xxxx0825	<b>Equate Symbol:</b> <code>IwmRsnCodeAscModeNotPrimary</code> <b>Meaning:</b> Caller invoked service but was not DAT on Primary ASC mode.
8	xxxx0828	<b>Equate Symbol:</b> <code>IwmRsnCodeBadVersion</code> <b>Meaning:</b> Version number in parameter list or version length field is not valid. This reason code is only set when the MVS release is on MVS/ESA HBB6603 or later levels.
8	xxxx0829	<b>Equate Symbol:</b> <code>IwmRsnCodeBadOptions</code> <b>Meaning:</b> Parameter list omits required parameters or supplies mutually exclusive parameters or provides data associated with options not selected. Note that this reason code will only occur on calls to this service through the IWMDNDRG C language interface. <b>Action:</b> Check for possible storage overlay of the parameter list.
C	—	<b>Equate Symbol:</b> <code>IwmRetCodeEnvError</code> <b>Meaning:</b> Environmental error.

## IWMSRDRS Macro

Table 61. Return and Reason Codes for the IWMSRDRS Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
C	xxxx0C35	<b>Equate Symbol:</b> IwmRsnCodeNotSecAuthServReg  <b>Meaning:</b> The caller is not authorized by SAF to register/deregister a server.  <b>Action:</b> Only supervisor state or authorized key 8 callers can register or deregister servers. Key 8 callers must have RACF authority to the BPX.WLMSEVER profile in the facility class (BPX.WLMSEVER must be defined for authorized users to execute the DNS services). The RACROUTE command is invoked during the SAF check, which does not allow the caller to be in cross memory mode or have an FRR established.
10	—	<b>Equate Symbol:</b> IwmRetCodeCompError  <b>Meaning:</b> Component error (no reason codes)

---

## Chapter 55. IWMSRFSV – Finding a Sysplex Routing Server

IWMSRFSV will find a server associated with the specified application environment and return the associated server data which was passed at the time the server connected to WLM (via IWM4CON). The only eligible servers are those that have connected to WLM with a specification of IWM4CON SERVER\_MANAGER=YES,SERVER\_TYPE=ROUTING, and whose application environment matches the input value passed to IWMSRFSV, which implies that the server belongs to the same subsystem type as the caller. The server chosen is considered a best choice to run work in terms of a variety of system conditions which are monitored.

When no eligible servers are already started, and the service policy allows MVS to start a server, and certain other conditions apply, MVS will start a new server on behalf of the request. Circumstances such as this imply that the program calling this service may be suspended until the request can be resolved. When no eligible servers exist and none can be started the caller will receive a return code to reflect this.

---

### Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Supervisor state or program key mask (PKM) allowing keys 0-7.
<b>Dispatchable unit mode:</b>	Task
<b>Cross memory mode:</b>	PASN=HASN=SASN
<b>AMODE:</b>	31-bit
<b>ASC mode:</b>	Primary
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks may be held.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

---

### Programming Requirements

1. The macro IWMYCON must be included to use this macro.
2. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
3. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, IWMRSNCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.
4. All character inputs are assumed to be padded on the right with blanks, when needed to fill out the entire length.

---

### Restrictions

1. This macro may not be used during task/address space termination.
2. NO FRRs may be established.

## IWMSRFSV Macro

3. The Connect token from the input parameter list must be owned by the current home address space.
4. The address space from which this service is invoked must have previously connected to WLM, using IWM4CON Router=Yes. The input application environment must be associated in the current service policy with the subsystem type specified through IWM4CON.

---

## Input Register Information

Before issuing the IWMSRFSV macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

---

## Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Reason code if GR15 return code is non-zero
1	Used as work registers by the system
2-13	Unchanged
14	Used as work registers by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

---

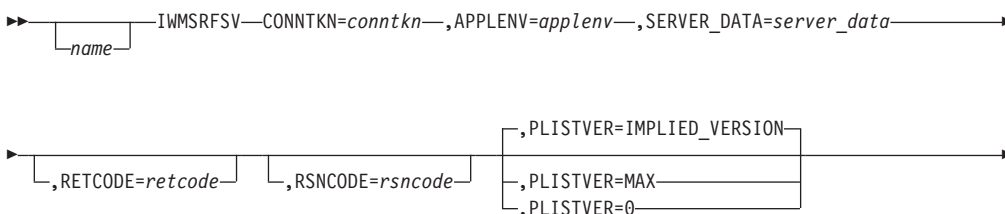
## Performance Implications

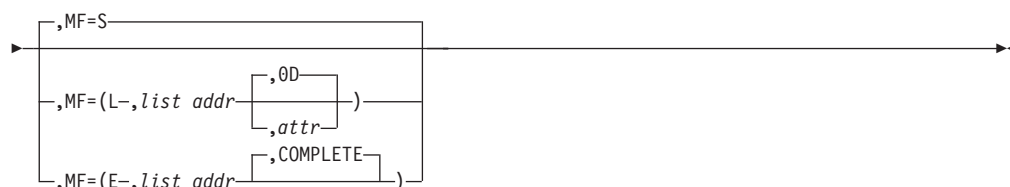
The task issuing this service may be suspended while a new server address space is being started, possibly on another MVS image.

---

## Syntax

### main diagram





## Parameters

The parameters are explained as follows:

### *name*

An optional symbol, starting in column 1, that is the name on the IWMSRFSV macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

### **,APPLENV=***applenv*

A required input parameter, which contains the application environment under which work requests are to be served.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32-character field.

### **CONNTKN=***conntkn*

A required input parameter, which contains the connect token for the current home space.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32 bit field.

### **,MF=S**

### **,MF=(L,***list addr***)**

### **,MF=(L,***list addr,attr***)**

### **,MF=(L,***list addr,OD***)**

### **,MF=(E,***list addr***)**

### **,MF=(E,***list addr,COMPLETE***)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

### *,list addr*

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

### *,attr*

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter

## IWMSRFSV Macro

list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

### **,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

### **,PLISTVER=IMPLIED\_VERSION**

### **,PLISTVER=MAX**

### **,PLISTVER=0**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form; in this way, MAX ensures that the parameter list does not overwrite nearby storage.

- **0**, if you use the currently available parameters.

**To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX
- A decimal value of 0

### **,RETCODE=*retcode***

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

### **,RSNCODE=*rsncode***

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

### **,SERVER\_DATA=*server\_data***

A required output parameter, which contains the data needed to uniquely identify the chosen server. The structure of this data is undefined to MVS, and is the same data passed when the server connected using IWM4CON SERVER\_MANAGER=YES, SERVER\_TYPE=ROUTING, SERVER\_DATA=...

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32-character field.

---

## ABEND Codes

None.

## Return and Reason Codes

When the IWMSRFSV macro returns control to your program:

- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, if you coded RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 62. Return and Reason Codes for the IWMSRFSV Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> <code>IwmRetCodeOk</code> <b>Meaning:</b> Successful completion. <b>Action:</b> None required.
8	—	<b>Equate Symbol:</b> <code>IwmRetCodeInvocError</code> <b>Meaning:</b> Invalid invocation environment or parameters.
8	xxxx0801	<b>Equate Symbol:</b> <code>IwmRsnCodeSRBMode</code> <b>Meaning:</b> Caller is in SRB mode. <b>Action:</b> Avoid requesting this function in SRB mode.
8	xxxx0803	<b>Equate Symbol:</b> <code>IwmRsnCodeDisabled</code> <b>Meaning:</b> Caller is disabled. <b>Action:</b> Avoid requesting this function while disabled.
8	xxxx0804	<b>Equate Symbol:</b> <code>IwmRsnCodeLocked</code> <b>Meaning:</b> Caller is locked. <b>Action:</b> Avoid requesting this function while locked.
8	xxxx080B	<b>Equate Symbol:</b> <code>IwmRsnCodeBadPI</code> <b>Meaning:</b> Error accessing parameter list. <b>Action:</b> Check for possible storage overlay.
8	xxxx0810	<b>Equate Symbol:</b> <code>IwmRsnCodeEutFrr</code> <b>Meaning:</b> Caller has EUT FRR established. <b>Action:</b> Avoid requesting this function with an EUT FRR.
8	xxxx0823	<b>Equate Symbol:</b> <code>IwmRsnCodeDatoff</code> <b>Meaning:</b> Caller invoked service while DATOFF <b>Action:</b> Avoid requesting this function in this environment.
8	xxxx0824	<b>Equate Symbol:</b> <code>IwmRsnCodeAmode24</code> <b>Meaning:</b> Caller invoked service but was in 24 bit addressing mode. <b>Action:</b> Request this function only when you are in 31 bit addressing mode.

Table 62. Return and Reason Codes for the IWMSRFSV Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0825	<p><b>Equate Symbol:</b> IwmRsnCodeAscModeNotPrimary</p> <p><b>Meaning:</b> Caller invoked service but was not DAT on Primary ASC mode.</p> <p><b>Action:</b> Avoid requesting this function in this environment.</p>
8	xxxx0826	<p><b>Equate Symbol:</b> IwmRsnCodeTaskTerm</p> <p><b>Meaning:</b> Caller invoked service while task termination is in progress for the current task.</p> <p><b>Action:</b> Avoid requesting this function in this environment.</p>
8	xxxx0828	<p><b>Equate Symbol:</b> IwmRsnCodeBadVersion</p> <p><b>Meaning:</b> Version number or version length field in parameter list is not valid.</p> <p><b>Action:</b> Check for possible overlay of the parameter list.</p>
8	xxxx083B	<p><b>Equate Symbol:</b> IwmRsnCodeHomeNotOwnConn</p> <p><b>Meaning:</b> Home address space does not own the connect token from the input parameter list.</p> <p><b>Action:</b> Invoke the function with the correct home address space.</p>
8	xxxx0840	<p><b>Equate Symbol:</b> IwmRsnCodeServiceNotEnabled</p> <p><b>Meaning:</b> Caller's space connection is not enabled for this service.</p> <p><b>Action:</b> Avoid requesting this function under the input connection. IWM4CON options must be specified previously to enable this service.</p>
8	xxxx0841	<p><b>Equate Symbol:</b> IwmRsnCodeXmemMode</p> <p><b>Meaning:</b> Caller invoked service but was in cross-memory mode.</p> <p><b>Action:</b> Avoid requesting this function in cross-memory mode.</p>
8	xxxx0842	<p><b>Equate Symbol:</b> IwmRsnCodeNoWLMConnect</p> <p><b>Meaning:</b> Caller's space is not connected to WLM.</p> <p><b>Action:</b> Issue IWM4CON with the necessary options prior to invoking this service.</p>
C	—	<p><b>Equate Symbol:</b> IwmRetCodeEnvError</p> <p><b>Meaning:</b> Environmental error.</p>
C	xxxx0C01	<p><b>Equate Symbol:</b> IwmRsnCodeNoStg</p> <p><b>Meaning:</b> Storage is not available for the request.</p> <p><b>Action:</b> There is a storage shortage. The function may work successfully at a later time.</p>



Table 62. Return and Reason Codes for the IWMSRFSV Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
C	xxxx0C1A	<p><b>Equate Symbol:</b> lwmRsnCodeApplNotDefined</p> <p><b>Meaning:</b> The application environment name is not defined in the active WLM policy.</p> <p><b>Action:</b> Check whether the correct application environment name is being used. If so, a service administrator must define the application environment in the WLM service definition.</p>
C	xxxx0C1B	<p><b>Equate Symbol:</b> lwmRsnCodeApplNotSST</p> <p><b>Meaning:</b> The application environment name is defined for use by a different subsystem type in the active WLM policy.</p> <p><b>Action:</b> Check whether the correct application environment name being used. If so, a service administrator must change the application environment in the WLM service definition to specify the correct subsystem type.</p>
C	xxxx0C1C	<p><b>Equate Symbol:</b> lwmRsnCodeServerNotStarted</p> <p><b>Meaning:</b> No server exists for the specified application environment and no server could be started.</p> <p><b>Action:</b> No action required. The function may be successful if invoked again.</p>
C	xxxx0C22	<p><b>Equate Symbol:</b> lwmRsnCodeApplEnvQuiesced</p> <p><b>Meaning:</b> The specified application environment has been quiesced, server cannot be started for the request.</p> <p><b>Action:</b> Restart the application environment and then retry the request.</p>
C	xxxx0C23	<p><b>Equate Symbol:</b> lwmRsnCodeIndLocalSystem</p> <p><b>Meaning:</b> Local system is not running with the current WLM policy, new server cannot be started for the request.</p> <p><b>Action:</b> Avoid requesting this function while the local system is not running with the current WLM policy.</p>
C	xxxx0C24	<p><b>Equate Symbol:</b> lwmRsnCodeProcNameBlank</p> <p><b>Meaning:</b> Server procname is blank, server cannot be started for the request.</p> <p><b>Action:</b> Check the server procname, fix it, and then retry the request.</p>
C	xxxx0C25	<p><b>Equate Symbol:</b> lwmRsnCodeApplEnvStopped</p> <p><b>Meaning:</b> WLM has given up trying to start a server because of failures. The associated application environment has been internally stopped.</p> <p><b>Action:</b> Restart the application environment and then retry the request.</p>
C	xxxx0C26	<p><b>Equate Symbol:</b> lwmRsnCodeRouterNotActive</p> <p><b>Meaning:</b> Either there is no router exists for the requested server or the router exists but not active. No server can be selected/started on this system.</p> <p><b>Action:</b> Re-connect the router for the requested application environment to WLM and then retry the request.</p>

## IWMSRFSV Macro

Table 62. Return and Reason Codes for the IWMSRFSV Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
C	xxxx0C27	<b>Equate Symbol:</b> IwmRsnCodeFsvReqInCompat <b>Meaning:</b> Reserved.
10	—	<b>Equate Symbol:</b> IwmRetCodeCompError <b>Meaning:</b> Component error. <b>Action:</b> No action required. The function may be successful if invoked again.

---

## Example

To determine the best server to which to route work, specify the following:

```
        IWMSRFSV CONNTKN=CTKN,                X
          APPLENV=AENAME,                    X
          SERVER_DATA=SVRDATA,                X
          RETCODE=RC,RSNCODE=RSN
*
* Storage areas
*
CTKN    DS    F           Contains the connect token for
*                               the current home space
AENAME  DS    CL32        Contains the application
*                               environment name
SVRDATA DS    CL32        Contains the output server data
RC      DS    F           Return code
RSN     DS    F           Reason code
```

---

## Chapter 56. IWMSRSRG – Register a Server for Sysplex Routing

The purpose of the IWMSRSRG service is to register a server that wishes to participate in sysplex workload balancing. The service allows the caller to identify an address space to be associated with a triplet, corresponding to location name, network id and LU name. This triplet is expected to be unique across all registered spaces in the sysplex, and, should be unique across all networks. The caller can additionally associate the triplet with a host name. Specifying a host name is optional, and if it is not coded it is set to blanks by the system. A list of eligible servers is made available to IWMSRSRS. These work requests include enclaves owned by the space as well as the address space's activity itself.

If this macro is issued to register a LOCATION.NETWORK\_ID.LUNAME that already exists on the issuing MVS image, the second registration will be ignored (IWMSRDRS should first be used to deregister the triplet). This condition will be identified through a unique return and reason code. Due to timing considerations, sysplex-wide uniqueness is not enforced, and so is the responsibility of the caller.

| The caller can additionally associate the server with a health indicator. This is a  
| value between 0 and 100, that indicates the percentage up to which the server is  
| capable of performing its normal work. This health indicator is used by the routing  
| service to modify the routing recommendations according to this indicator. It can be  
| modified by calling this service again with another HEALTH value. In this case the  
| return and reason code for already registered servers will not be returned. This is  
| the same as if the HEALTH value was set or modified by the IWM4HLTH service.

After a server registers by issuing this macro, the sysplex routing service IWMSRSRS can be issued to return a weighted list of registered servers in the sysplex to which work could be directed. Alternatively, IWMSRSRS can be used to obtain a complete list of servers associated with a given location when invoked on a system running OS/390 R3 or later. IWMSRSRS can be used to obtain the user data associated with each server when invoked on a system at OS/390 R3 or later, though servers resident on an earlier release will be associated with user data which is binary zeroes. Since the propagation of the registration to other systems is asynchronous, a newly registered triplet will not be immediately visible to other systems.

A server is automatically deregistered during job termination or memory termination.

---

### Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Problem state with any PSW key (see previous note)
<b>Dispatchable unit mode:</b>	Task or SRB
<b>Cross memory mode:</b>	Any PASN, any HASN, any SASN (see previous note)
<b>AMODE:</b>	31-bit
<b>ASC mode:</b>	Primary
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks may be held. FRRs may be established (see previous note).

**Control parameters:** Control parameters must be in the primary address space.

---

### Programming Requirements

1. The macro CVT must be included to use this macro.
2. The macro IWMYCON must be included to use this macro.
3. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
4. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, IWMSRNCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.
5. All character data, unless otherwise specified, is assumed to be left justified and padded with blanks on the right, as needed, to occupy the specified number of bytes.

---

### Restrictions

This macro supports multiple versions. Some keywords are only supported by certain versions. The PLISTVER=3 option can only be used on systems running z/OS V1R7 with OA14310 installed, and on systems running z/OS V1R8 or higher. Refer to the PLISTVER parameter description for further information.

---

### Input Register Information

Before issuing the IWMSRSRG macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

---

### Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Reason code if GR15 return code is non-zero
1	Used as work register by the system
2-13	Unchanged
14	Used as work register by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

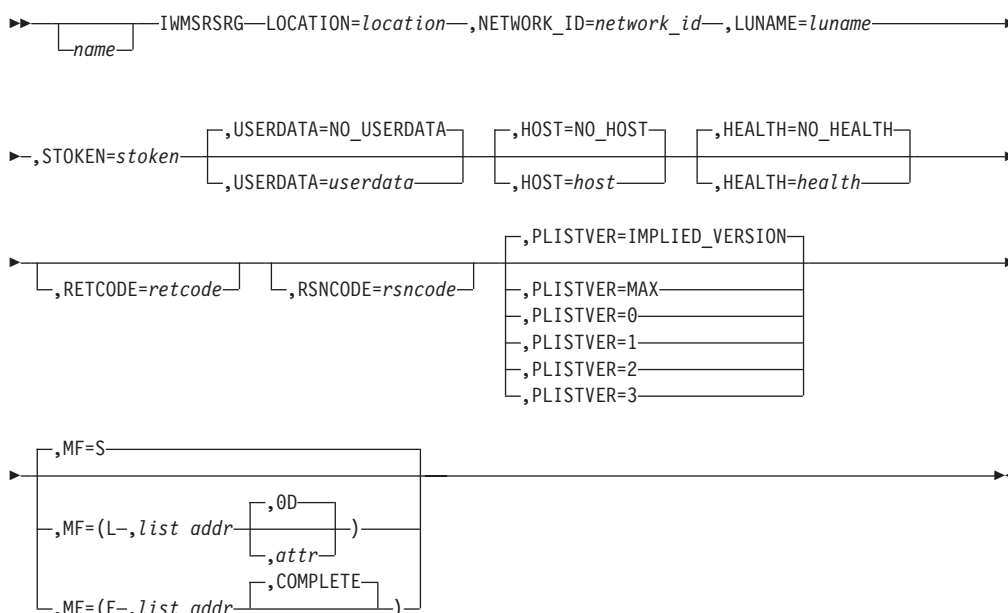
Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

## Performance Implications

None.

## Syntax

### main diagram



## Parameters

The parameters are explained as follows:

### *name*

An optional symbol, starting in column 1, that is the name on the IWMSRSRG macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

### **,HEALTH=health**

### **,HEALTH=NO\_HEALTH**

An optional input parameter, which contains the health factor associated with the address space. This is a value, that reflects, up to which percentage this address space is capable to handle requests. NO\_HEALTH indicates that no health indicator was passed. This is the default. In this case, a health value of 100 is assumed, when the server is registered.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

### **,HOST=host**

### **,HOST=NO\_HOST**

An optional input parameter, which contains the host name associated with the address space to be registered. The value should be padded on the right with blanks for any unused characters. NO\_HOST indicates that no host name was passed. This is the default.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 64-character field.

### **LOCATION=***location*

A required input parameter, which contains the server LOCATION associated with the address space to be registered. The value should be padded on the right with blanks for any unused characters.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 18-character field.

### **,LUNAME=***luname*

A required input parameter, which contains the server Logical Unit name associated with the address space to be registered.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

### **,MF=S**

#### **,MF=(L,***list addr***)**

#### **,MF=(L,***list addr,attr***)**

#### **,MF=(L,***list addr,0D***)**

#### **,MF=(E,***list addr***)**

#### **,MF=(E,***list addr,COMPLETE***)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

#### **,***list addr*

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

#### **,***attr*

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

### **,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

### **,NETWORK\_ID=***network\_id*

A required input parameter, which contains the Network ID associated with the address space to be registered.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

### **,PLISTVER=IMPLIED\_VERSION**

### **,PLISTVER=MAX**

**,PLISTVER=0**

**,PLISTVER=1**

**,PLISTVER=2**

**,PLISTVER=3**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form; in this way, MAX ensures that the parameter list does not overwrite nearby storage.

- **0**, which supports all parameters except those specifically referenced in higher versions.
- **1**, which supports the following parameter and those from version 0:

USERDATA

- **2**, which supports the following parameter and those from version 0 and 1:

HOST

- **3**, which supports the following parameter and those from version 0,1 and 2:

HEALTH

**Note:** The PLISTVER=3 option can only be used on systems running z/OS V1R7 with OA14310 installed, and on systems running z/OS V1R8 or higher.

**To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX
- A decimal value of 0, 1, 2, or 3

**,RETCODE=retcode**

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,RSNCODE=rsncode**

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

## IWMSRSRG Macro

### ,STOKEN=*stoken*

A required input parameter, which contains the space token of the server to be registered.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 64-bit field.

### ,USERDATA=*userdata*

### ,USERDATA=NO\_USERDATA

An optional input parameter, which contains data meaningful to the user of this service. This user data is available to callers of the IWMSRSRS service when invoked on a system running OS/390 R3 or later.

Userdata may only be passed via IWMSRSRG when running OS/390 R3 or higher - when passed to lower level systems the data will be ignored and no return code set to reflect this condition. The format is undefined to MVS. The default is NO\_USERDATA, which indicates that no user data was passed.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 64-character field.

## ABEND Codes

None.

## Return and Reason Codes

When the IWMSRSRG macro returns control to your program:

- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, if you coded RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 63. Return and Reason Codes for the IWMSRSRG Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> IwmRetCodeOk <b>Meaning:</b> Successful completion. <b>Action:</b> None required.
4	—	<b>Equate Symbol:</b> IwmRetCodeWarning <b>Meaning:</b> Successful completion, unusual conditions noted.
4	xxxx0419	<b>Equate Symbol:</b> IwmRsnCodeServerAlreadyReg <b>Meaning:</b> Server already registered. <b>Action:</b> None required.
8	—	<b>Equate Symbol:</b> IwmRetCodeInvocError <b>Meaning:</b> Invalid invocation environment or parameters.
8	xxxx0803	<b>Equate Symbol:</b> IwmRsnCodeDisabled <b>Meaning:</b> Caller is disabled. <b>Action:</b> Avoid requesting this function while disabled.



Table 63. Return and Reason Codes for the IWMSRSRG Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0804	<b>Equate Symbol:</b> lwmRsnCodeLocked <b>Meaning:</b> Caller is locked. <b>Action:</b> Avoid requesting this function while locked.
8	xxxx0807	<b>Equate Symbol:</b> lwmRsnCodeBadSTOKEN <b>Meaning:</b> Bad STOKEN passed. <b>Action:</b> Check for possible storage overlay.
8	xxxx080B	<b>Equate Symbol:</b> lwmRsnCodeBadPI <b>Meaning:</b> Error accessing parameter list. <b>Action:</b> Check for possible storage overlay.
8	xxxx0823	<b>Equate Symbol:</b> lwmRsnCodeDatoff <b>Meaning:</b> Caller invoked service while DATOFF <b>Action:</b> Avoid requesting this function in this environment.
8	xxxx0824	<b>Equate Symbol:</b> lwmRsnCodeAmode24 <b>Meaning:</b> Caller invoked service but was in 24-bit addressing mode. <b>Action:</b> Request this function only when you are in 31 bit addressing mode.
8	xxxx0825	<b>Equate Symbol:</b> lwmRsnCodeAscModeNotPrimary <b>Meaning:</b> Caller invoked service but was not DAT on Primary ASC mode. <b>Action:</b> Avoid requesting this function in this environment.
8	xxxx0828	<b>Equate Symbol:</b> lwmRsnCodeBadVersion <b>Meaning:</b> Version number in parameter list or version length field is not valid. This reason code is only set when the MVS release is on MVS/ESA HBB6603 or higher levels. <b>Action:</b> Check for possible storage overlay of the parameter list.
8	xxxx08A2	<b>Equate Symbol:</b> lwmRsnCodeBadHealth <b>Meaning:</b> Health Value out of range <b>Action:</b> Check for possible storage overlay.
C	—	<b>Equate Symbol:</b> lwmRetCodeEnvError <b>Meaning:</b> Environmental error.
C	xxxx0C01	<b>Equate Symbol:</b> lwmRsnCodeNoStg <b>Meaning:</b> Storage is not available for the request. <b>Action:</b> There is a storage shortage. The function may work successfully at a later time.
10	—	lwmRetCodeCompError: Component error (no reason codes)

## Example

To register an instance of a given location, specify:

## IWMSRSRG Macro

```
IWMSRSRG LOCATION=LOC,NETWORK_ID=NET,LUNAME=LU,  
          STOKEN=STKN,HOST=HST,HEALTH=HLTH,  
          USERDATA=DATA,RETCODE=RC,RSNCODE=RSN
```

```
*  
* Storage areas  
*  
LOC      DS    CL18      Contains the Location  
*                               associated with the server  
*                               instance  
NET      DS    CL8       Contains the Network id  
*                               associated with the server  
*                               instance  
LU       DS    CL8       Contains the LU name  
*                               associated with the server  
*                               instance  
STKN     DS    CL8       Contains the STOKEN  
*                               associated with the server  
*                               instance  
DATA     DS    CL64      Contains the user data  
*                               associated with the server  
*                               instance  
HST      DS    CL64      Contains the Host name  
*                               associated with the server  
*                               instance  
HLTH     DS    F         Contains the Health indicator  
*                               associated with the server  
*                               instance  
RC       DS    F         Return code  
RSN      DS    F         Reason code
```

---

## Chapter 57. IWMSRSRS – Determine Where to Route Work

IWMSRSRS provides three functions: SELECT, QUERY and SPECIFIC. All return a list of registered servers known to the system on which the service is invoked. Servers which have deregistered, via IWMSRDRS, may still be present in the output list, due to the asynchronous nature of deregistration. Conversely, some registered servers may not appear for this same reason.

When either the SELECT or the SPECIFIC function is chosen, IWMSRSRS will return a list of servers in the sysplex which are associated with the input Location name along with a relative weighting for each server. These servers are identified by their Network id and LU name, which were previously registered using the Sysplex Router Register macro, IWMSRSRG. Note that some servers may not appear in the output list due to balancing decisions, so this service should not be used as a general query service to find all currently registered servers for the input location.

Next to each server in the list will be a weight which tells the caller the relative number of requests to send to each entry. For example, the caller might send the indicated number of requests to each LU in the list before routing to the next LU in the list.

Server	Weight
-----	-----
NETIDA.LUNAME1	4
NETIDB.LUNAME2	7
NETIDC.LUNAME3	1
NETIDD.LUNAME43	4
NETID4.LUNAME2	2

The requestor could then choose to send the first 4 requests to NETIDA.LUNAME1, the next 7 requests to NETIDB.LUNAME2, the next request to NETIDC.LUNAME3, and so forth. When the list is exhausted, the requestor could invoke this macro again and get a whole new list or could rotate through the list again. It is expected that the requestor would invoke this macro frequently to get current system views for work balancing. For example, it would be appropriate for the caller to invoke this service approximately every 1 to 3 minutes, so that the list will remain current with changing system conditions and server availability.

The Query function is only available on systems running OS/390 R3 or higher. When the Query function is requested, IWMSRSRS will return the list of all servers in the sysplex which are associated with the input Location name along with a fixed weight of one for each server. The format of the output is the same as for SELECT.

The Sysplex Routing Service (IWMSRSRS) can be used to obtain the userdata associated with each server when invoked on a system at OS/390 R3 or later, though servers resident on an earlier release will be associated with userdata which is binary zeroes.

The SPECIFIC function is available on systems running z/OS R7 or higher. When the SPECIFIC function is issued, IWMSRSRS returns a list of servers in the sysplex which are associated with the input Location name along with a relative weighting for each server. The weights in the Select function consider available capacity values and the number of servers on the same system.

## IWMSRSRS Macro

With the SPECIFIC function the following three additional factors are taken into account:

- The performance index that indicates the achievement of the WLM-defined goals of the server, that is its related work. A server that achieves its goal is preferred over one that does not achieve its goal.
- If the server owns independent enclaves those also take the delays into account that the work is subject to, due to the queue times of the owned enclaves. A server with less average queue times for its enclaves is preferred over one with higher queue times.
- The health factor of this server. It is dependent on the health indicator which was reported to WLM for this server by the IWM4HLTH service or by IWMSRSRG. If no health indicator was reported, this factor is also neutral. This factor is only taken into account on systems running z/OS V1R7 with OA14310 installed, and on systems running z/OS V1R8 or higher.

---

## Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Problem state. Any PSW key
<b>Dispatchable unit mode:</b>	Task or SRB
<b>Cross memory mode:</b>	Any PASN, any HASN, any SASN
<b>AMODE:</b>	31-bit
<b>ASC mode:</b>	Primary
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks may be held. FRRs may be established.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

---

## Programming Requirements

1. The macro CVT must be included to use this macro.
2. The macro IWMYCON must be included to use this macro.
3. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
4. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, IWMSRNCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.
5. All character data, unless otherwise specified, is assumed to be left justified and padded with blanks on the right, as needed, to occupy the specified number of bytes.

---

## Restrictions

This macro may only be used with MVS/ESA SP5.2 or higher versions. Extended data may only be returned when running on OS/390 R3 or higher versions, and server instances residing on prior releases will be assigned user data values consisting of all binary zeroes.

## Input Register Information

Before issuing the IWMSRSRS macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

## Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Reason code if GR15 return code is non-zero
1	Used as work register by the system
2-13	Unchanged
14	Used as work register by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

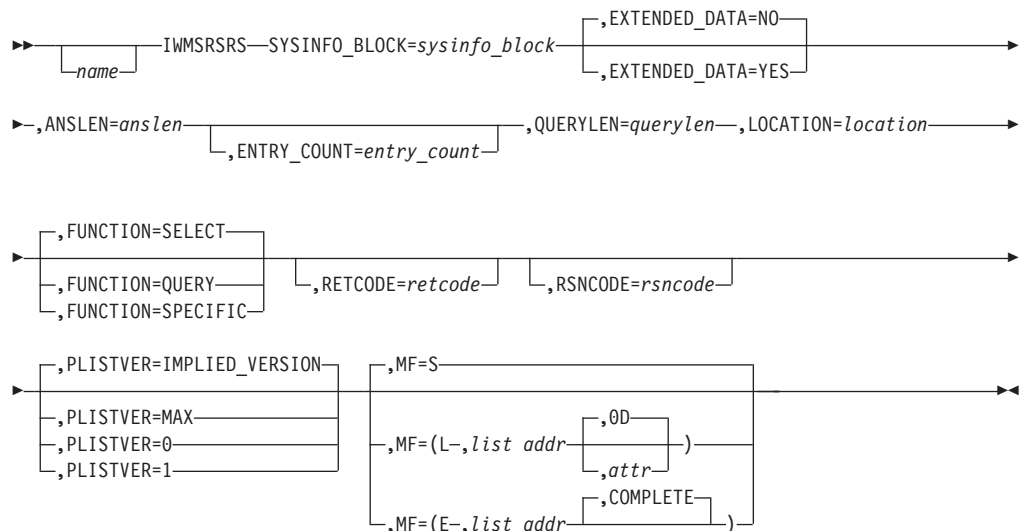
Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

## Performance Implications

None.

## Syntax

### main diagram



---

## Parameters

The parameters are explained as follows:

*name*

An optional symbol, starting in column 1, that is the name on the IWMSRSRS macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

**,ANSLEN=anslen**

A required input parameter, which contains the length of the SYSINFO\_BLOCK in bytes.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

**,ENTRY\_COUNT=entry\_count**

An optional output parameter, which will hold the number of server entries returned by the service. This is the number of entries in the SYSR\_INFO array (see IWMWSYSR). ENTRY\_COUNT may only be specified when running on OS/390 R3 or later. On earlier releases this request is ignored and the variable is zeroed. No return code is given that this has happened.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

**,EXTENDED\_DATA=NO**

**,EXTENDED\_DATA=YES**

An optional parameter, which describes whether the format of the output area named by SYSINFO\_BLOCK includes the extended section or not (see IWMWSYSR). The default is EXTENDED\_DATA=NO.

**,EXTENDED\_DATA=NO**

indicates that the format of the output area named by SYSINFO\_BLOCK includes only the standard information mapped by the SYSR, which consists of an array of entries described by SYSR\_INFO.

**,EXTENDED\_DATA=YES**

indicates that the format of the output area given by SYSINFO\_BLOCK includes first the standard information mapped by the SYSR, which consists of an array of entries described by SYSR\_INFO, followed immediately by the header for the extension section and any other data described by the header.

EXTENDED\_DATA is only available to callers of the IWMSRSRS service when invoked on a system running OS/390 R3 or later. The EXTENDED\_DATA returned by OS/390 R3 includes the header and associated userdata. Data associated with entries on MVS images not supporting a particular type of data will be set as binary zeroes - for example servers on MVS SP5.2 will have userdata which is all zeroes. No special indication will be given for this condition. In particular no special return or reason codes are given.

On releases prior to OS/390 R3, EXTENDED\_DATA(YES) is processed as though EXTENDED\_DATA(NO) was specified. No return code is given that this has happened.

**,FUNCTION=SELECT**

**,FUNCTION=QUERY**

**,FUNCTION=SPECIFIC**

An optional parameter, which describes which set of servers are of interest to the caller. The default is FUNCTION=SELECT.

**,FUNCTION=SELECT**

indicates that the servers best suited to receive work are to be returned.

**,FUNCTION=QUERY**

indicates that all servers associated with the input LOCATION are to be returned. QUERY may only be specified when running on OS/390 R3 or later. On earlier releases this request is processed as a SELECT request. In this case no return code is issued.

**,FUNCTION=SPECIFIC**

indicates that the servers best suited to receive work are to be returned together with their weights. The weight calculation differs from the weights that are returned by the SELECT function, in that each weight takes also server-specific information for each server into account the Performance Index (PI) of the server and the Queue time of enclaves, if the server is an enclave owner. SPECIFIC may only be specified when running z/OS V1R7 or higher. On earlier releases this request is processed as a SELECT request. In this case no return code is issued.

**,LOCATION=*location***

A required input parameter, which contains the LOCATION associated with the registered address spaces which are candidates to receive work.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 18-character field.

**,MF=S**

**,MF=(L,*list addr*)**

**,MF=(L,*list addr,attr*)**

**,MF=(L,*list addr,0D*)**

**,MF=(E,*list addr*)**

**,MF=(E,*list addr,COMPLETE*)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

**,*list addr***

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

**,*attr***

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter

list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

**,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

**,PLISTVER=IMPLIED\_VERSION**

**,PLISTVER=MAX**

**,PLISTVER=0**

**,PLISTVER=1**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form; in this way, MAX ensures that the parameter list does not overwrite nearby storage.

- **0**, if you use only the following parameters:

ANSLEN	QUERYLEN
LOCATION	SYSINFO_BLOCK

- **1**, if you use any of the following parameters, and parameters from version 0, or both:

ENTRY_COUNT	EXTENDED_DATA	FUNCTION
-------------	---------------	----------

**To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX
- A decimal value of 0 or 1

**,QUERYLEN=*querylen***

A required output parameter, variable which contains the number of bytes needed for all data requested, taking into account the format specified via the EXTENDED\_DATA keyword.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

**,RETCODE=*retcode***

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,RSNCODE=*rsncode***

An optional output parameter into which the reason code is to be copied from GPR 0.



**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**SYSINFO\_BLOCK=***sysinfo\_block*

A required input parameter, which specifies the name of the area to be filled in with the system information for the registered, active, LUs in the SYSPLEX associated with the input location.

The area must be large enough to contain at least 1 entry. The format of this area is mapped by IWMWSYSR. The EXTENDED\_DATA keyword describes the desired format. The FUNCTION keyword describes which servers are candidates for inclusion.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a field.

## ABEND Codes

None.

## Return and Reason Codes

When the IWMSRSRS macro returns control to your program:

- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, if you coded RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 64. Return and Reason Codes for the IWMSRSRS Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> <i>IwmRetCodeOk</i> <b>Meaning:</b> Successful completion. <b>Action:</b> None required.
4	—	<b>Equate Symbol:</b> <i>IwmRetCodeWarning</i> <b>Meaning:</b> Successful completion, unusual conditions noted.
4	xxxx040A	<b>Equate Symbol:</b> <i>IwmRsnCodeOutputAreaTooSmall</i> <b>Meaning:</b> The output area supplied is too small to receive all the available information. <b>Action:</b> None required. If necessary, reinvoke the service with an output area of sufficient size to receive all information.
4	xxxx040B	<b>Equate Symbol:</b> <i>IwmRsnCodeNoServersRegistered</i> <b>Meaning:</b> No Servers have registered in the sysplex. <b>Action:</b> None required.
8	—	<b>Equate Symbol:</b> <i>IwmRetCodeInvocError</i> <b>Meaning:</b> Invalid invocation environment or parameters.
8	xxxx0803	<b>Equate Symbol:</b> <i>IwmRsnCodeDisabled</i> <b>Meaning:</b> Caller is disabled. <b>Action:</b> Avoid requesting this function while disabled.

Table 64. Return and Reason Codes for the IWMSRSRS Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0804	<b>Equate Symbol:</b> IwmRsnCodeLocked <b>Meaning:</b> Caller is locked. <b>Action:</b> Avoid requesting this function while locked.
8	xxxx080B	<b>Equate Symbol:</b> IwmRsnCodeBadPI <b>Meaning:</b> Error accessing parameter list. <b>Action:</b> Check for possible storage overlay.
8	xxxx0823	<b>Equate Symbol:</b> IwmRsnCodeDatoff <b>Meaning:</b> Caller invoked service while DATOFF <b>Action:</b> Avoid requesting this function in this environment.
8	xxxx0824	<b>Equate Symbol:</b> IwmRsnCodeAmode24 <b>Meaning:</b> Caller invoked service but was in 24 bit addressing mode. <b>Action:</b> Request this function only when you are in 31 bit addressing mode.
8	xxxx0825	<b>Equate Symbol:</b> IwmRsnCodeAscModeNotPrimary <b>Meaning:</b> Caller invoked service but was not DAT on Primary ASC mode. <b>Action:</b> Avoid requesting this function in this environment.
8	xxxx0828	<b>Equate Symbol:</b> IwmRsnCodeWrongSysLevels <b>Meaning:</b> There are servers registered in the sysplex, associated with the input Location name, but with a z/OS level prior to V1R7 for the SPECIFIC function in the routing service. <b>Action:</b> Either deregister the servers with the downlevel software, or use IWMSRSRS with the SELECT function instead of SPECIFIC.
8	xxxx89D	<b>Equate Symbol:</b> IwmRsnCodeBadOptions <b>Meaning:</b> Parameter list omits required parameters or supplies mutually exclusive parameters or provides data associated with options not selected. Note that this reason code will only occur on calls to this service through the IWMDNSRV C language interface. <b>Action:</b> Check for possible storage overlay of the parameter list.
C	—	<b>Equate Symbol:</b> IwmRetCodeEnvError <b>Meaning:</b> Environmental error.
C	xxxx0C01	<b>Equate Symbol:</b> IwmRsnCodeNoStg <b>Meaning:</b> Storage is not available for the request. <b>Action:</b> There is a storage shortage. The function may work successfully at a later time.

Table 64. Return and Reason Codes for the IWMSRSRS Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
10	—	<p><b>Equate Symbol:</b> IwmRetCodeCompError</p> <p><b>Meaning:</b> Component error.</p> <p><b>Action:</b> No action required. The function may be successful if invoked again.</p>

## Example

To register an instance of a given location, specify the following:

```

        IWMSRSRS SYSINFO_BLOCK=DATA,
                EXTENDED_DATA=YES,
                ANSLLEN=SIZE,
                ENTRY_COUNT=E,
                QUERYLEN=Q,
                LOCATION=LOC,
                FUNCTION=QUERY,RETCODE=RC,RSNCODE=RSN
*
* Storage areas
*
DATA    DS    CL200      Area to receive output
SIZEEQU EQU   *-DATA     Equate for size of Data
E       DS    F         Field to receive entry count
Q       DS    F         Field to receive query length
LOC     DS    CL18      Contains the Location
*                               associated with the server
*                               instance
RC      DS    F         Return code
RSN     DS    F         Reason code
SIZE    DC    A(SIZEEQU) Field to hold size
    
```



---

## Chapter 58. IWMUEXPT – WLM Undo Export Service

The IWMUEXPT macro undoes an earlier request to export an enclave via the IWMEXPT macro.

The caller is expected to invoke IWMUEXPT after all importing systems have invoked IWMUIMPT. If IWMUEXPT is invoked while other systems have imported the enclave, WLM loses the ability to manage the multisystem enclave as a unit. Also the enclave owner's SMF 30 record will not contain all of the CPU time used by the enclaves on the other systems.

---

### Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Supervisor state or program key mask (PKM) allowing keys 0-7.
<b>Dispatchable unit mode:</b>	Task or SRB
<b>Cross memory mode:</b>	Any HASN, any SASN. PASN must be the address space which connected to WLM.
<b>AMODE:</b>	31-bit
<b>ASC mode:</b>	Primary
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks may be held.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

---

### Programming Requirements

1. The macro CVT must be included to use this macro.
2. The macro IWMYCON must be included to use this macro.
3. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
4. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, IWMRSNCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.

---

### Restrictions

None.

---

### Input Register Information

Before issuing the IWMUEXPT macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

---

### Output Register Information

When control returns to the caller, the GPRs contain:

## IWMUEXPT Macro

Register	Contents
0	Reason code if GR15 return code is non-zero
1	Used as work registers by the system
2-13	Unchanged
14	Used as work registers by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

---

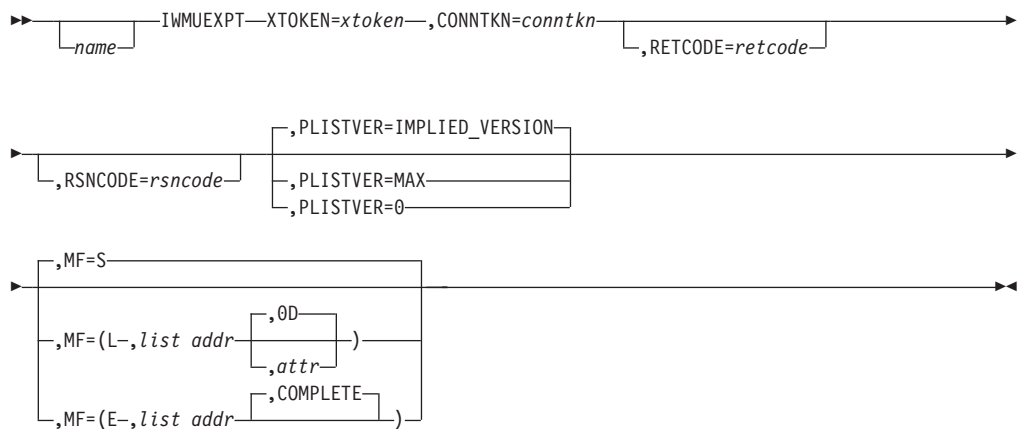
## Performance Implications

None.

---

## Syntax

### main diagram



---

## Parameters

The parameters are explained as follows:

### *name*

An optional symbol, starting in column 1, that is the name on the IWMUEXPT macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

### **,CONNTKN=conntkn**

A required input parameter that contains the connect token for the primary address space's connection to WLM.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32 bit field.

**,MF=S**  
**,MF=(L,list addr)**  
**,MF=(L,list addr,attr)**  
**,MF=(L,list addr,OD)**  
**,MF=(E,list addr)**  
**,MF=(E,list addr,COMPLETE)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

*,list addr*

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

*,attr*

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

**,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

**,PLISTVER=IMPLIED\_VERSION**

**,PLISTVER=MAX**

**,PLISTVER=0**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form, when both are assembled with the same level of the system. In this way, MAX ensures that the parameter list does not overwrite nearby storage.

## IWMUEXPT Macro

- 0, if you use the currently available parameters.

**To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX
- A decimal value of 0

**,RETCODE=retcode**

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,RSNCODE=rsncode**

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**XTOKEN=xtoken**

A required input parameter that contains the export token which identifies the exported enclave.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32-character field.

---

## ABEND Codes

None.

---

## Return and Reason Codes

When the IWMUEXPT macro returns control to your program:

- GPR 15 (and *retcode*, when you code RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, when you code RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 65. Return and Reason Codes for the IWMUEXPT Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> IwmRetCodeOk <b>Meaning:</b> Successful completion. <b>Action:</b> None required.
4	—	<b>Equate Symbol:</b> IwmRetCodeWarning <b>Meaning:</b> Successful completion, unusual conditions noted.
4	xxxx0432	<b>Equate Symbol:</b> IwmRsnCodeUnknownExportToken <b>Meaning:</b> No enclave matching the export token was found. The enclave may have been unexported or deleted, or the WLM coupling facility structure may have been lost. <b>Action:</b> None.
8	—	<b>Equate Symbol:</b> IwmRetCodeInvocError <b>Meaning:</b> Invalid invocation environment or parameters.



Table 65. Return and Reason Codes for the IWMUEXPT Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0803	<b>Equate Symbol:</b> IwmRsnCodeDisabled <b>Meaning:</b> Caller is disabled. <b>Action:</b> Avoid requesting this function while disabled.
8	xxxx0804	<b>Equate Symbol:</b> IwmRsnCodeLocked <b>Meaning:</b> Caller is locked. <b>Action:</b> Avoid requesting this function while locked.
8	xxxx080B	<b>Equate Symbol:</b> IwmRsnCodeBadPI <b>Meaning:</b> Error accessing parameter list. <b>Action:</b> Check for possible storage overlay.
8	xxxx0821	<b>Equate Symbol:</b> IwmRsnCodeBadConn <b>Meaning:</b> Input connect token does not pass validity checking. <b>Action:</b> Make sure the primary address space connected to WLM using the IWM4CON service. Make sure the connect token returned by IWM4CON is passed correctly.
8	xxxx0824	<b>Equate Symbol:</b> IwmRsnCodeAmode24 <b>Meaning:</b> Caller invoked service but was in 24 bit addressing mode. <b>Action:</b> Request this function only when you are in 31 bit addressing mode.
8	xxxx0825	<b>Equate Symbol:</b> IwmRsnCodeAscModeNotPrimary <b>Meaning:</b> Caller invoked service but was not in primary ASC mode. <b>Action:</b> Avoid requesting this function in this environment.
8	xxxx0827	<b>Equate Symbol:</b> IwmRsnCodeRsvdNot0 <b>Meaning:</b> Reserved field in parameter list was non-zero. <b>Action:</b> Check for use of keywords that are not supported by the MVS release on which the program is running.
8	xxxx0828	<b>Equate Symbol:</b> IwmRsnCodeBadVersion <b>Meaning:</b> Version number in parameter list is not valid. <b>Action:</b> Check for possible storage overlay of the parameter list.
8	xxxx0840	<b>Equate Symbol:</b> IwmRsnCodeServiceNotEnabled <b>Meaning:</b> Caller's connection is not enabled for this service. <b>Action:</b> Make sure that EXPTIMPT=YES is specified on the IWM4CON macro invocation.
8	xxxx0842	<b>Equate Symbol:</b> IwmRsnCodeNoWLMConnect <b>Meaning:</b> Caller's primary address space is not connected to WLM. <b>Action:</b> Invoke the IWM4CON macro before invoking this macro.

## IWMUEXPT Macro

Table 65. Return and Reason Codes for the IWMUEXPT Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0870	<b>Equate Symbol:</b> IwmRsnCodeBadExportToken <b>Meaning:</b> The export token is not validly formatted. <b>Action:</b> Check for possible storage overlay of the export token.
8	xxxx0871	<b>Equate Symbol:</b> IwmRsnCodeDidNotExportOrImport <b>Meaning:</b> The primary address space did not export the enclave. <b>Action:</b> Invoke IWMUEXPT from the correct address space.
C	—	<b>Equate Symbol:</b> IwmRetCodeEnvError <b>Meaning:</b> Environmental error.
10	—	<b>Equate Symbol:</b> IwmRetCodeCompError <b>Meaning:</b> Component error. <b>Action:</b> Contact your system programmer.

---

## Example

```
                IWMUEXPT XTOKEN=EXPORTT,CONNTKN=CONNECTT
*
* Storage areas
*
EXPORTT DS      CL32      Export token
CONNECTT DS     CL4       Connect token
```

---

## Chapter 59. IWMUIMPT – WLM Undo Import Service

The IWMUIMPT macro undoes an earlier request to import an enclave via the IWMIMPT macro.

---

### Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Supervisor state or program key mask (PKM) allowing keys 0-7.
<b>Dispatchable unit mode:</b>	Task or SRB
<b>Cross memory mode:</b>	Any HASN, any SASN. PASN must be the address space which connected to WLM.
<b>AMODE:</b>	31-bit
<b>ASC mode:</b>	Primary
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks may be held.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

---

### Programming Requirements

1. The macro CVT must be included to use this macro.
2. The macro IWMYCON must be included to use this macro.
3. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
4. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, IWMRSNCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.

---

### Restrictions

None.

---

### Input Register Information

Before issuing the IWMUIMPT macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

---

### Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Reason code if GR15 return code is non-zero
1	Used as work registers by the system
2-13	Unchanged
14	Used as work registers by the system

15 Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

---

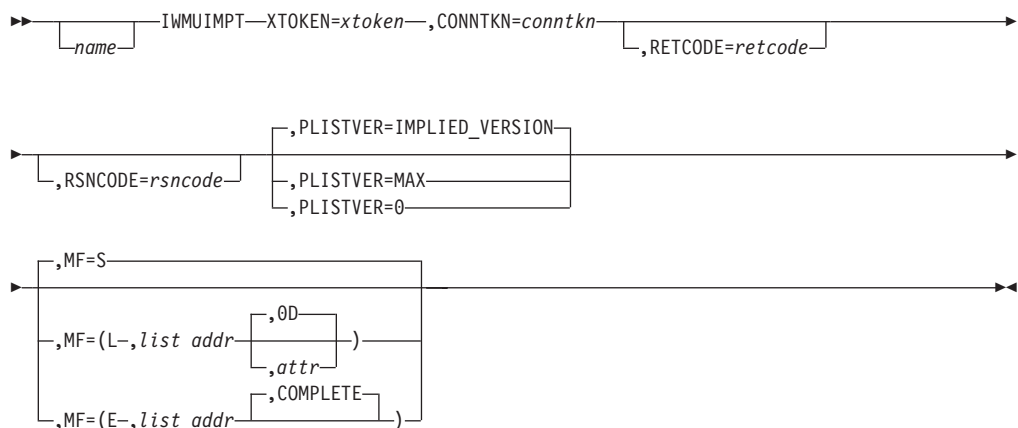
## Performance Implications

None.

---

## Syntax

### main diagram



---

## Parameters

The parameters are explained as follows:

### *name*

An optional symbol, starting in column 1, that is the name on the IWMUIPMT macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

### **,CONNTKN=conntkn**

A required input parameter that contains the connect token for the primary address space's connection to WLM.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32 bit field.

### **,MF=S**

**,MF=(L,list addr)**

**,MF=(L,list addr,attr)**

**,MF=(L,list addr,OD)**

**,MF=(E,*list addr*)**

**,MF=(E,*list addr*,**COMPLETE**)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

**,*list addr***

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

**,*attr***

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

**,**COMPLETE****

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

**,PLISTVER=IMPLIED\_VERSION**

**,PLISTVER=MAX**

**,PLISTVER=0**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form, when both are assembled with the same level of the system. In this way, MAX ensures that the parameter list does not overwrite nearby storage.

- **0**, if you use the currently available parameters.

**To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX

- A decimal value of 0

**,RETCODE=retcode**

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,RSNCODE=rsncode**

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**XTOKEN=xtoken**

A required input parameter that contains the export token which identifies the exported enclave.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32-character field.

## ABEND Codes

None.

## Return and Reason Codes

When the IWMUIIPT macro returns control to your program:

- GPR 15 (and *retcode*, when you code RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, when you code RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 66. Return and Reason Codes for the IWMUIIPT Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> <code>IwmRetCodeOk</code> <b>Meaning:</b> Successful completion. <b>Action:</b> None required.
4	—	<b>Equate Symbol:</b> <code>IwmRetCodeWarning</code> <b>Meaning:</b> Successful completion, unusual conditions noted.
4	xxxx0432	<b>Equate Symbol:</b> <code>IwmRsnCodeUnknownExportToken</code> <b>Meaning:</b> No enclave matching the export token was found. The enclave may have been unexported or deleted, or the WLM coupling facility structure may have been lost. <b>Action:</b> None.
8	—	<b>Equate Symbol:</b> <code>IwmRetCodeInvocError</code> <b>Meaning:</b> Invalid invocation environment or parameters.
8	xxxx0803	<b>Equate Symbol:</b> <code>IwmRsnCodeDisabled</code> <b>Meaning:</b> Caller is disabled. <b>Action:</b> Avoid requesting this function while disabled.

Table 66. Return and Reason Codes for the IWMUIMPT Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0804	<b>Equate Symbol:</b> lwmRsnCodeLocked <b>Meaning:</b> Caller is locked. <b>Action:</b> Avoid requesting this function while locked.
8	xxxx080B	<b>Equate Symbol:</b> lwmRsnCodeBadPI <b>Meaning:</b> Error accessing parameter list. <b>Action:</b> Check for possible storage overlay.
8	xxxx0821	<b>Equate Symbol:</b> lwmRsnCodeBadConn <b>Meaning:</b> Input connect token does not pass validity checking. <b>Action:</b> Make sure the primary address space connected to WLM using the IWM4CON service. Make sure the connect token returned by IWM4CON is passed correctly.
8	xxxx0824	<b>Equate Symbol:</b> lwmRsnCodeAmode24 <b>Meaning:</b> Caller invoked service but was in 24 bit addressing mode. <b>Action:</b> Request this function only when you are in 31 bit addressing mode.
8	xxxx0825	<b>Equate Symbol:</b> lwmRsnCodeAscModeNotPrimary <b>Meaning:</b> Caller invoked service but was not in primary ASC mode. <b>Action:</b> Avoid requesting this function in this environment.
8	xxxx0827	<b>Equate Symbol:</b> lwmRsnCodeRsvdNot0 <b>Meaning:</b> Reserved field in parameter list was non-zero. <b>Action:</b> Check for use of keywords that are not supported by the MVS release on which the program is running.
8	xxxx0828	<b>Equate Symbol:</b> lwmRsnCodeBadVersion <b>Meaning:</b> Version number in parameter list is not valid. <b>Action:</b> Check for possible storage overlay of the parameter list.
8	xxxx0840	<b>Equate Symbol:</b> lwmRsnCodeServiceNotEnabled <b>Meaning:</b> Caller's connection is not enabled for this service. <b>Action:</b> Make sure that EXPTIMPT=YES is specified, or used by default, on the IWM4CON macro invocation.
8	xxxx0842	<b>Equate Symbol:</b> lwmRsnCodeNoWLMConnect <b>Meaning:</b> Caller's primary address space is not connected to WLM. <b>Action:</b> Invoke the IWM4CON macro before invoking this macro.
8	xxxx0870	<b>Equate Symbol:</b> lwmRsnCodeBadExportToken <b>Meaning:</b> The export token is not validly formatted. <b>Action:</b> Check for possible storage overlay of the export token.

Table 66. Return and Reason Codes for the IWMUIMPT Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0871	<p><b>Equate Symbol:</b> IwmRsnCodeDidNotExportOrImport</p> <p><b>Meaning:</b> The primary address space did not import the enclave.</p> <p><b>Action:</b> Invoke IWMUIMPT from the correct address space.</p>
C	—	<p><b>Equate Symbol:</b> IwmRetCodeEnvError</p> <p><b>Meaning:</b> Environmental error.</p>
10	—	<p><b>Equate Symbol:</b> IwmRetCodeCompError</p> <p><b>Meaning:</b> Component error.</p> <p><b>Action:</b> Contact your system programmer.</p>

## Example

```

                IWMUIMPT XTOKEN=EXPORTT,CONNTKN=CONNECTT
*
* Storage areas
*
EXPORTT DS    CL32      Export token
CONNECTT DS   CL4       Connect token

```



---

## Chapter 60. IWMWMCON – Modify Connect Information

The purpose of this service is to modify a particular connection to WLM with respect to the associated subsystem type and/or name for Work Manager services, described below, and so could replace the use of the pair of services Disconnect (IWM4DIS) and Connect (IWM4CON) for the new values of subsystem type and/or name. This change only affects Work Manager related services, and does not affect the subsystem identify for Queue Manager or Server Manager services. For this reason, the caller must be connected to the WLM work management services, i.e. IWM4CON WORK\_MANAGER=YES must be specified. Queue Manager and/or Server Manager may also be specified at Connect, but are not affected by use of IWMWMCON. The PSW key and topology list associated with the connect may not be changed via this service.

Use of this service needs to be coordinated with the use of other Work Manager services which depend on the Connect token and the associated subsystem related information to ensure that the desired results are obtained. Among these services are Classify (IWMCLSFY), Report (IWMRPT), Notify (IWMMNTFY) where it is an optional input, and Enclave Create (IWM4ECRE). Note that use of IWMWMCON is not appropriate prior to creation of Enclaves with TYPE(Dependent) or TYPE(Montkn).

---

### Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Supervisor state or program key mask (PKM) allowing keys 0-7.
<b>Dispatchable unit mode:</b>	Task or SRB
<b>Cross memory mode:</b>	Any P,S. Current Home address space must be the same as Home when the corresponding Connect was invoked.
<b>AMODE:</b>	31-bit
<b>ASC mode:</b>	Primary
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks may be held.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

---

### Programming Requirements

1. The macro CVT must be included to use this macro.
2. The macro IWMYCON must be included to use this macro.
3. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
4. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, IWMSRNCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.

### Restrictions

1. If the key specified on IWM4CON was a user key (8-F), then the following must ALL be true:
  - Caller must be in non-cross-memory mode (P=S=H). This implies that the current primary must match the primary at the time that IWM4CON was invoked. Running in a subspace is not supported.
  - Must be in task mode (not SRB)
  - Current task must match the task at the time that IWM4CON was invoked.
2. It is the caller's responsibility to serialize use of this service with use of IWMCLSFY and other services using the connect token. Failure to do so may result in classification to a service and/or report class which is other than the intended one.
3. This service should not be invoked while in a RTM termination routine (resource manager) for the task owning the connect token since MVS will have its own resource cleanup routine and unpredictable results would occur. It is legitimate to use this service while in a recovery routine, however, or in mainline processing.

---

### Input Register Information

Before issuing the IWMWMCON macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

---

### Output Register Information

When control returns to the caller, the GPRs contain:

<b>Register</b>	<b>Contents</b>
<b>0</b>	Reason code if GR15 return code is non-zero
<b>1</b>	Used as work registers by the system
<b>2-13</b>	Unchanged
<b>14</b>	Used as work registers by the system
<b>15</b>	Return code

When control returns to the caller, the ARs contain:

<b>Register</b>	<b>Contents</b>
<b>0-1</b>	Used as work registers by the system
<b>2-13</b>	Unchanged
<b>14-15</b>	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

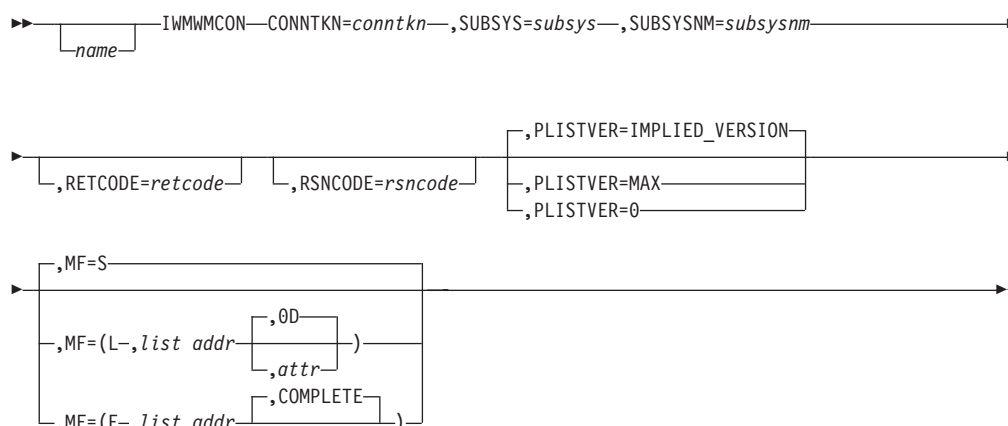
---

### Performance Implications

None.

## Syntax

### main diagram



## Parameters

The parameters are explained as follows:

### *name*

An optional symbol, starting in column 1, that is the name on the IWMWMCON macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

### CONNTKN=*conntkn*

A required input parameter, which contains the connect token for the environment to be modified.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32 bit field.

### ,MF=S

,MF=(L,*list addr*)

,MF=(L,*list addr*,*attr*)

,MF=(L,*list addr*,**OD**)

,MF=(E,*list addr*)

,MF=(E,*list addr*,**COMPLETE**)

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

## IWMWMCON Macro

**,list** *addr*

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

**,attr**

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

**,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

**,PLISTVER=IMPLIED\_VERSION**

**,PLISTVER=MAX**

**,PLISTVER=0**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form; in this way, MAX ensures that the parameter list does not overwrite nearby storage.

- **0**, if you use the currently available parameters.

**To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX
- A decimal value of 0

**,RETCODE=retcode**

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,RSNCODE=rsncode**

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,SUBSYS=subsys**

A required input parameter, which contains the generic subsystem type (e.g. IMS, CICS, etc.).

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 4-character field.

**,SUBSYSNM=***subsysnm*

A required input parameter, which contains the name of the specific subsystem instance

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

## ABEND Codes

None.

## Return and Reason Codes

When the IWMWMCON macro returns control to your program:

- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, if you coded RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 67. Return and Reason Codes for the IWMWMCON Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> <code>IwmRetCodeOk</code> <b>Meaning:</b> Successful completion. <b>Action:</b> None required.
4	—	<b>Equate Symbol:</b> <code>IwmRetCodeWarning</code> <b>Meaning:</b> Successful completion, unusual conditions noted.
4	xxxx0409	<b>Equate Symbol:</b> <code>IwmRsnCodeNoConn</code> <b>Meaning:</b> Input connection token does not reflect an active connection to WLM. <b>Action:</b> None required.
8	—	<b>Equate Symbol:</b> <code>IwmRetCodeInvocError</code> <b>Meaning:</b> Invalid invocation environment or parameters.
8	xxxx0802	<b>Equate Symbol:</b> <code>IwmRsnCodeXmemUserKeyTkn</code> <b>Meaning:</b> Caller is in cross-memory mode while the token was requested in user key. <b>Action:</b> Avoid requesting this function while in cross-memory mode.
8	xxxx0803	<b>Equate Symbol:</b> <code>IwmRsnCodeDisabled</code> <b>Meaning:</b> Caller is disabled. <b>Action:</b> Avoid requesting this function while disabled.
8	xxxx0804	<b>Equate Symbol:</b> <code>IwmRsnCodeLocked</code> <b>Meaning:</b> Caller is locked. <b>Action:</b> Avoid requesting this function while locked.

## IWMWMCON Macro

Table 67. Return and Reason Codes for the IWMWMCON Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0809	<p><b>Equate Symbol:</b> IwmRsnCodeSrbUserKeyTkn</p> <p><b>Meaning:</b> Caller is in SRB mode, while the token was obtained in a user key (8-F).</p> <p><b>Action:</b> Avoid requesting this function in SRB mode for tokens associated with user key.</p>
8	xxxx080A	<p><b>Equate Symbol:</b> IwmRsnCodeTcbNotOwnerUserKeyTkn</p> <p><b>Meaning:</b> Current task is not the owner, while the token was obtained in a user key (8-F).</p> <p><b>Action:</b> Avoid requesting this function under a task other than the owner for a token associated with user key.</p>
8	xxxx080B	<p><b>Equate Symbol:</b> IwmRsnCodeBadPI</p> <p><b>Meaning:</b> Error accessing parameter list.</p> <p><b>Action:</b> Check for possible storage overlay.</p>
8	xxxx0823	<p><b>Equate Symbol:</b> IwmRsnCodeDatoff</p> <p><b>Meaning:</b> Caller invoked service while DATOFF</p> <p><b>Action:</b> Avoid requesting this function in this environment.</p>
8	xxxx0824	<p><b>Equate Symbol:</b> IwmRsnCodeAmode24</p> <p><b>Meaning:</b> Caller invoked service but was in 24 bit addressing mode.</p> <p><b>Action:</b> Request this function only when you are in 31 bit addressing mode.</p>
8	xxxx0825	<p><b>Equate Symbol:</b> IwmRsnCodeAscModeNotPrimary</p> <p><b>Meaning:</b> Caller invoked service but was not DAT on Primary ASC mode.</p> <p><b>Action:</b> Avoid requesting this function in this environment.</p>
8	xxxx0826	<p><b>Equate Symbol:</b> IwmRsnCodeTaskTerm</p> <p><b>Meaning:</b> Caller invoked service while task termination is in progress for the task associated with the owner.</p> <p><b>Action:</b> Avoid requesting this function in this environment.</p>
8	xxxx0827	<p><b>Equate Symbol:</b> IwmRsnCodeRsvdNot0</p> <p><b>Meaning:</b> Reserved field in parameter list was non-zero.</p> <p><b>Action:</b> Check for possible storage overlay of the parameter list.</p>
8	xxxx0828	<p><b>Equate Symbol:</b> IwmRsnCodeBadVersion</p> <p><b>Meaning:</b> Version number in parameter list is not valid.</p> <p><b>Action:</b> Check for possible storage overlay of the parameter list.</p>
8	xxxx082F	<p><b>Equate Symbol:</b> IwmRsnCodeWrongHome</p> <p><b>Meaning:</b> Caller invoked the service from the wrong home address space.</p> <p><b>Action:</b> Invoke the function with the correct home address space.</p>

Table 67. Return and Reason Codes for the IWMWMCON Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0840	<p><b>Equate Symbol:</b> IwmRsnCodeServiceNotEnabled</p> <p><b>Meaning:</b> Caller's space connection is not enabled for this service. The caller is not connected to the WLM work management services.</p> <p><b>Action:</b> Avoid requesting this function under the input connection. IWM4CON options must be specified previously to enable this service. Check the specification of the WORK_MANAGER keyword on the IWM4CON macro invocation.</p>
10	—	<p><b>Equate Symbol:</b> IwmRetCodeCompError</p> <p><b>Meaning:</b> Component error.</p> <p><b>Action:</b> No action required. The function may be successful if invoked again.</p>

---

## Examples

To override the subsystem name and subsystem type provided on a previous call to IWM4CON, specify:

```
IWMWMCON  SUBSYS=GENSUB, SUBSYSNM=SUBNAME,
           CONNTKN=CONNTOKEN, RETCODE=RC, RSNCODE=RSN,
```

Where the following are declared:

GENSUB	DS	CL4	Generic subsystem type
SUBNAME	DS	CL8	Subsystem name
CONNTOKEN	DS	FL4	Connect token





---

## Chapter 61. IWMWQRY – Query Service

IWMWQRY provides information to help the subsystem work manager make work routing and scheduling decisions. IWMWQRY allows the caller to obtain the service class goals and importance for a service class by performance period.

The caller must provide an area of storage in the ANSAREA=*ansarea* and the length of that area in the ANSLEN=*anslen* for IWMWQRY to place the service class goal and importance information. IWMWQRY returns the actual length of the information in the QUERYLEN=*querylen* parameter.

The answer area is mapped by the IWMSVPOL and IWMSVPCD data areas. The data areas are described in *z/OS MVS Data Areas, Vol 3 (IVT-RCWK)*

The first time the caller invokes this macro, you should specify QUERYLEN. The service returns the length required for the service class information

The information returned is not serialized upon return to the caller, and so may be out of date due to a change in service policy.

---

### Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Supervisor state, or problem state. Any PSW key.
<b>Dispatchable unit mode:</b>	Task
<b>Cross memory mode:</b>	Any PASN, any HASN, any SASN
<b>AMODE:</b>	31-bit
<b>ASC mode:</b>	Primary or Access Register (AR) mode.
<b>Interrupt status:</b>	Enabled
<b>Locks:</b>	Unlocked
<b>Control parameters:</b>	Control parameters must be in the primary address space.

---

### Programming Requirements

- Make sure no EUT FRRs are established.
- If you are in AR mode, you must specify SYSSTATE ASCENV=AR before invoking IWMWQRY.
- You must include the CVT and the IWMYCON mapping macros in the calling program.

---

### Restrictions

None.

---

### Input Register Information

Before issuing the IWMWQRY macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

---

## Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Reason code if the return code in GPR 15 is not 0, otherwise, used as a work register by the system.
1	Used as a work register by the system.
2 - 13	Unchanged
14	Used as a work register by the system.
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0 - 1	Used as a work register by the system.
2 - 13	Unchanged
14 - 15	Used as a work register by the system.

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

---

## Performance Implications

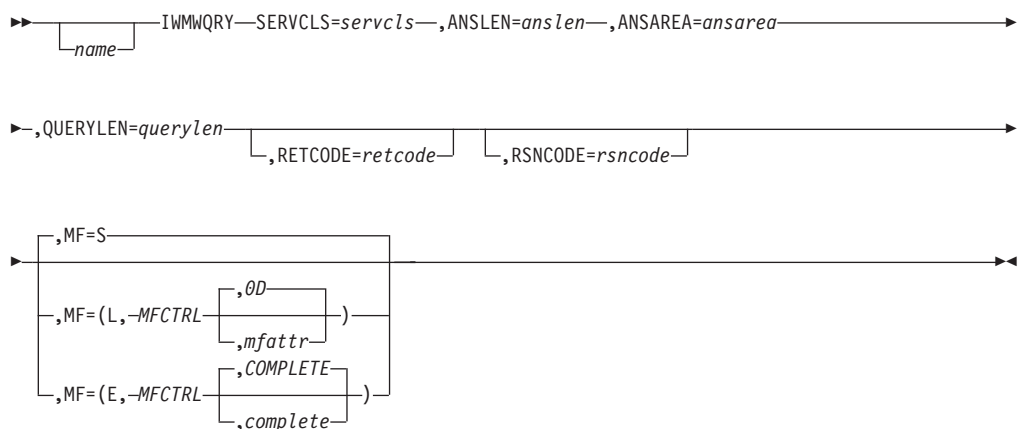
None.

---

## Syntax

The syntax of the IWMWQRY macro is as follows:

### main diagram




---

## Parameters

The parameters are explained as follows:

### **SERVCLS=***servcls*

Required input parameter that specifies the token representing the service class, and if there is one, the report class.

**To code:** Specify the RS-type name or address (using a register from 2 to 12) of a 32 bit field containing the service class token.

**,ANSLEN=anslen**

Required input parameter containing the length of the area provided to hold the data returned by IWMWQRY.

**To code:** Specify the RS-type name or address (using a register from 2 to 12) of a 32 bit field containing the length of the area provided.

**,ANSAREA=ansarea**

Required output parameter that specifies the area provided to contain the data being returned by IWMWQRY.

The area is mapped by the IWMSVPCD mapping macro and the service class period definition section in the IWMSVPOL mapping macro. The IWMSVPCD part of the answer area contains the offset to the class data, the size of the class data, and the size of each period entry. The IWMSVPOL part of the answer area contains the service class definition section, and the service class period definition of the service class.

**To code:** Specify the RS-type name or address (using a register from 2 to 12) of a character field specifying an area to contain the data returned by the query service.

**,QUERYLEN=querylen**

Required output parameter that specifies the number of bytes needed to contain the service class information.

**To code:** Specify the RS-type name or address (using a register from 2 to 12) of a fullword field to contain the required number of bytes.

**,RETCODE=retcode addr**

Optional output parameter that specifies where the system is to store the return code. The return code is also in GPR 15.

**To code:** Specify the name (RS-type) or address (using a register from 2 to 12) of a fullword to contain the return code.

**,RSNCODE=rsncode addr**

Optional output parameter that specifies where the system is to store the reason code. The reason code is also in GPR 0.

**To code:** Specify the name (RS-type) or address (using a register from 2 to 12) of a fullword to contain the reason code (if any).

**,MF=S**

**,MF=(L,mfctrl,mfattr)**

**,MF=(E,mfctrl,COMPLETE)**

Use MF=S to specify the standard form, which places parameters into an inline parameter list and invokes the IWM4CON macro service.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require re-entrant code. The list form defines an area of storage that the execute form uses to store the parameters.

Use MF=E to specify the execute form of the macro. Use the execute form with the list form of the macro for applications that require re-entrant code. The execute form stores the parameters into the storage area defined by the list form and generates the macro invocation to transfer control to the service.

## IWMWQRY Macro

*,mfctrl*

Use this output parameter to specify the name of the storage area to contain the parameters.

**To code:** Specify the name (RS-type) or address (using a register from 2 to 12) of the storage area containing the parameter list.

*,mfattr*

Use this input parameter to specify the name of a 1 to 60 character storage area that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *,mfattr* the system provides a value of 0D, which forces the parameter on a doubleword boundary.

**,COMPLETE**

Use this input parameter to specify that the system check for required parameters and supply defaults for omitted optional parameter.

---

## ABEND Codes

None.

---

## Return and Reason Codes

When IWMWQRY macro returns control to your program, GPR 15 contains a return code. When the return code is non-zero, then GPR 0 contains a reason code.

Hexadecimal Return Code	Hexadecimal Reason Code	Meaning
00		<b>Meaning:</b> Warning.
04	0401	<b>Meaning:</b> Warning. The system does not support work manager services. The Query output area has not been modified. This return code is only set when the MVS release is prior to MVS/ESA SP 5.1.0.
04	040A	<b>Meaning:</b> Warning. The output area supplied is too small to receive all the available information.
04	0410	<b>Meaning:</b> Warning. The input service class token does not reflect a service class in the current service policy.
08	0801	<b>Meaning:</b> Program error. The caller is in SRB mode.
08	0803	<b>Meaning:</b> Program error. The caller is disabled.
08	0804	<b>Meaning:</b> Program error. The caller is locked.
08	080D	<b>Meaning:</b> Program error. Input service class is not valid.
08	0810	<b>Meaning:</b> Program error. Caller has EUT FRR established.
08	0830	<b>Meaning:</b> Program error. Caller has passed an invalid ALET.

---

## Example

For information related to the service class represented by the service class token in the SERVCLS field, specify:

```
IWMWQRY  ANSAREA=ANSAREA,ANSLN=ANSLN,SERVCLS=SERVCLS
         QUERYLEN=QUERYLEN,RETCODE=RCODE,
         RSNCODE=RSN,MF=(E,MFWQRY)
```

---

## Chapter 62. IWMWQWRK – Query Work Service

A work manager can use IWMWQWRK to help identify where its transactions may be executing. A caller can issue this for a work manager address space that is having trouble executing transactions, and wants to find out where transactions are abending.

A caller can narrow in on where the problem is occurring by using the LU 6.2 token information contained in monitoring environments.

With this service, a caller can get:

- A list of LU 6.2 tokens

Specify SEARCH\_BY=CONNTKN, and get a list of LU 6.2 tokens for all work requests reflected in monitoring environments owned by the current home address space. To do this, the caller should provide the connect token of the address space in the CONNTKN parameter.

- A list of ASIDs and/or STOKENs

Specify SEARCH\_BY=LU62TKN and provide a list of LU 6.2 tokens, and IWMWQWRK returns a list of ASIDs and/or STOKENs representing owners of monitoring environments initialized (via IWM4MINI) with an LU 6.2 token in the list.

The list of LU 6.2 tokens must have been obtained on a previous call to IWMWQWRK.

Monitoring environments owned by the home address space which are related (by IWMMRELA) to other monitoring environments are not searched.

Optionally, to narrow the search, a caller can also provide the subsystem type or the subsystem instance in the SUBSYS and SUBSYSNM parameters.

---

### Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Supervisor state, or problem state. Any PSW key.
<b>Dispatchable unit mode:</b>	Task
<b>Cross memory mode:</b>	PASN = HASN = SASN, unless all monitoring environments owned by the current home address space were created only in system keys. If the current home address space owns any monitoring environments created in a user key (8-F), then PASN = HASN = SASN.
<b>AMODE:</b>	31-bit
<b>ASC mode:</b>	Primary
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	Unlocked
<b>Control parameters:</b>	Control parameters must be in the primary address space.

---

### Programming Requirements

You must include the IWMYCON mapping macro in the calling program.

---

## Restrictions

None.

---

## Input Register Information

Before issuing the IWMWQWRK macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

---

## Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Reason code if the return code in GPR 15 is not 0, otherwise, used as a work register by the system.
1	Used as a work register by the system.
2 - 13	Unchanged
14	Used as a work register by the system.
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0 - 1	Used as a work register by the system.
2 - 13	Unchanged
14 - 15	Used as a work register by the system.

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

---

## Performance Implications

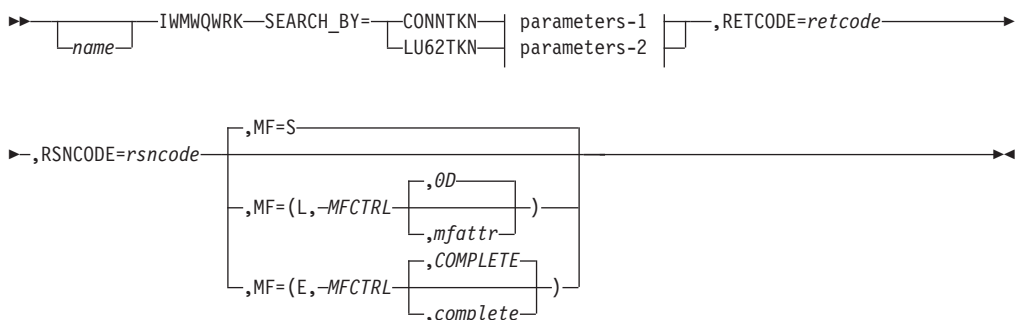
None.

---

## Syntax

The syntax of the IWMWQWRK macro is as follows:

### main diagram



**parameters-1**

```

▶—,CONNTKN=conntkn—,LU62_LIST=NO_LU62_LIST—,LU62_LIST=lu62_list—,LU62_LISTSIZE=lu62_listsize—
▶—,QUERYLEN=querylen—

```

**parameters-2**

```

▶—,SUBSYS=NO_SUBSYS—,SUBSYS=subsys—,SUBSYSNM=NO_SUBSYSNM—,SUBSYSNM=subsysnm—
▶—,LU62_LIST=lu62_list—,LU62_LISTSIZE=lu62_listsize—

```

```

▶—,ASID_LIST=NO_ASID_LIST—,ASID_LIST=asid_list— parameters-3
▶—,STOKEN_LIST=NO_STOKEN_LIST—,STOKEN_LIST=stoken_list— parameters-4 —,SKIP_REGIONS=skip_regions—

```

**parameters-3**

```

▶—,ASID_LISTSIZE=asid_listsize—,ASID_SIZENEED=asid_sizeneed—

```

**parameters-4**

```

▶—,STKN_LISTSIZE=stkn_listsize—,STKN_SIZENEED=stkn_sizeneed—

```

---

## Parameters

The parameters are explained as follows:

**SEARCH\_BY=CONNTKN****SEARCH\_BY=LU62TKN**

Required input parameter that specifies the type of search for the information about work.

Use SEARCH\_BY=CONNTKN to indicate that the information about work associated with the input connect token should be returned. The information returned is a list of LU 6.2 tokens, contained in LU62\_LIST, and a return code.

Use SEARCH\_BY=LU62TKN to indicate that information about work associated with the input list of LU 6.2 tokens (which the caller must provide in the LU62\_LIST=*lu62\_list*) should be returned.

You define the input list with the LU62\_LIST and LU62\_LISTSIZE keywords. The list must be the output of a prior call to IWMWQWRK specifying SEARCH\_BY=CONNTKN.

The information returned is a return code indicating whether work reflected in any monitoring environment owned by the current home address space is associated with ANY LU 6.2 token in the list of token provided in the LU62\_LIST.

**CONNTKN=*conntkn***

Required input parameter for SEARCH\_BY=CONNTKN that specifies the

## IWMWQWRK Macro

connect token returned by IWM4CON. The connect token (CONNTKN) must be owned by the current home address space.

**To code:** Specify the RS-type name or address in register (2)-(12), of a 32 bit field containing the connect token.

### **QUERYLEN=***querylen*

Required output parameter for SEARCH\_BY=CONNTKN that specifies the number of bytes needed to contain the information for the input connect token (CONNTKN=*conntkn*).

**To code:** Specify the RS-type name or address in register (2)-(12), of a fullword containing the number of bytes to contain the information.

### **SUBSYS=***subsys*

#### **SUBSYS=NO\_SUBSYS**

Optional input parameter for SEARCH\_BY=LU62TKN that specifies the generic subsystem type (ie CICS, IMS) This keyword helps narrow the search for the matching LU 6.2 token further.

**To code:** Specify the RS-type name or address in register (2)-(12), of a 4 character field containing the generic subsystem type.

### **SUBSYSNM=***subsysnm*

#### **SUBSYSNM=NO\_SUBSYSNM**

Optional input parameter for SEARCH\_BY=LU62TKN that specifies the subsystem instance name. This keyword helps narrow the search for the matching LU 6.2 token further.

**To code:** Specify the RS-type name or address in register (2)-(12), of an 8 character field containing the subsystem instance name.

### **LU62\_LIST=***lu62\_list*

#### **LU62\_LIST=NO\_LU62\_LIST**

Optional input/output parameter specifying the area for the list of LU 6.2 tokens. To specify an input LU62\_LIST, you must have previously invoked this macro to receive an output LU62\_LIST, and provide the list on a subsequent invocation.

**To code:** Specify the RS-type name or address in register (2)-(12), of a character field specifying the area for the LU 6.2 token list.

### **,LU62\_LISTSIZE=***lu62\_listsize*

Required input parameter for LU62\_LIST=*lu62\_list* specifying the length of the area provided to contain the data returned by IWMWQWRK.

**To code:** Specify the RS-type name or address in register (2)-(12), of a fullword field containing the length of the LU 6.2 token list.

### **ASID\_LIST=***asid\_list*

#### **ASID\_LIST=NO\_ASID\_LIST**

Optional input/output parameter that specifies an area for the list of ASIDs. Each entry (ASID) is 2 bytes. Only ASIDs for regions known to be involved in some work request are returned. Regions which could not be interrogated are reflected only in the SKIP\_REGIONS parameter.

ASID\_LIST=NO\_ASID\_LIST indicates that no list area is provided.

**To code:** Specify the RS-type name or address in register (2)-(12), of a character field specifying the area for the list of ASIDs.

### **ASID\_LISTSIZE=***asid\_listsize*

Required input parameter for ASID\_LIST=*asid\_list* that specifies the length of the area provided for the ASIDs returned by IWMWQWRK.



**To code:** Specify the RS-type name, or address in register (2)-(12), of a fullword containing the length of the area.

**ASID\_SIZENEED**=*asid\_sizeneed*

Required output parameter for ASID\_LIST=*asid\_list* that specifies the number of bytes needed for the output list of ASIDs.

**To code:** Specify the RS-type name or address in register (2)-(12), of a fullword to contain the number of bytes needed for the ASID list.

**STOKEN\_LIST**=*stoken\_list*

**STOKEN\_LIST=NO\_STOKEN\_LIST area is provided.**

Option input/output parameter for that specifies the area for the list of STOKENS. Only STOKENS for regions known to be involved in some work request are returned. Regions which could not be interrogated are only reflected in the SKIP\_REGIONS variable.

**To code:** Specify the name (RS-type), or address in register (2)-(12), of a character field specifying an area for the list of STOKENS.

**STKN\_LISTSIZE**=*stkn\_listsize*

Required input parameter for STOKEN\_LIST=*stoken\_list* that contains the length of the area provided for the STOKEN list being returned by IWMWQWRK.

**To code:** Specify the name (RS-type), or address in register (2)-(12), of a fullword containing the length of the area.

**STKN\_SIZENEED**=*stkn\_sizeneed*

Required input parameter for STOKEN\_LIST=*stoken\_list* that specifies the number of bytes needed for the output list of STOKENS.

**To code:** Specify the RS-type name or address in register (2)-(12), of a fullword containing the number of bytes.

**SKIP\_REGIONS**=*skip\_regions*

Optional output parameter for STOKEN\_LIST=*stoken\_list* that contains the number of address spaces skipped and therefore not included in the output list of ASIDs/STOKENS. If the caller wants to re-invoke IWMWQWRK when SKIP\_REGIONS is non-zero, it may be desirable to ensure that  $ASID\_SIZENEED + 2 * SKIP\_REGIONS \leq ASID\_LISTSIZE$ , AND that  $STKN\_SIZENEED + 8 * SKIP\_REGIONS \leq STKN\_LISTSIZE$  when these output areas are passed.

**,RETCODE**=*retcode addr*

Optional output parameter that specifies where the system is to store the return code. The return code is also in GPR 15.

**To code:** Specify the name (RS-type) or address (using a register from 2 to 12) of a fullword to contain the return code.

**,RSNCODE**=*rsncode addr*

Optional output parameter that specifies where the system is to store the reason code. The reason code is also in GPR 0.

**To code:** Specify the name (RS-type) or address (using a register from 2 to 12) of a fullword to contain the reason code (if any).

**,MF=S**

**,MF=(L,mfctrl,mfattr)**

**,MF=(E,mfctrl,COMPLETE)**

Use MF=S to specify the standard form, which places parameters into an inline parameter list and invokes the IWM4CON macro service.

## IWMWQWRK Macro

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require re-entrant code. The list form defines an area of storage that the execute form uses to store the parameters.

Use MF=E to specify the execute form of the macro. Use the execute form with the list form of the macro for applications that require re-entrant code. The execute form stores the parameters into the storage area defined by the list form and generates the macro invocation to transfer control to the service.

*,mfctrl*

Use this output parameter to specify the name of the storage area to contain the parameters.

**To code:** Specify the name (RS-type) or address (using a register from 2 to 12) of the storage area containing the parameter list.

*,mfattr*

Use this input parameter to specify the name of a 1 to 60 character storage area that can contain any value that is valid on an assembler DS pseudo-op. You can use this parameter to force boundary alignment of the parameter list. If you do not code *,mfattr* the system provides a value of 0D, which forces the parameter on a doubleword boundary.

**,COMPLETE**

Use this input parameter to specify that the system check for required parameters and supply defaults for omitted optional parameter.

---

## ABEND Codes

None.

---

## Return and Reason Codes

When IWMWQWRK macro returns control to your program, GPR 15 contains a return code. When the return code is non-zero, then GPR 0 contains a reason code.

Hexadecimal Return Code	Hexadecimal Reason Code	Meaning
00		<b>Meaning:</b> Successful completion.
04	0401	<b>Meaning:</b> Warning. System does not support work manager services. The query output area has not been modified. This return code is set only when the MVS release is prior to MVS/ESA SP 5.1.
04	0408	<b>Meaning:</b> Warning. No work matching the input search arguments was found.
04	0409	<b>Meaning:</b> Warning. Input connection token does not reflect an active connection to WLM.
04	040A	<b>Meaning:</b> Warning. The output area supplied is too small to receive all the available information.
08	0802	<b>Meaning:</b> Program error. Caller is in cross memory mode while some monitoring environments were in user key.
08	0803	<b>Meaning:</b> Program error. Caller is disabled.
08	0804	<b>Meaning:</b> Program error. Caller is locked.

Hexadecimal Return Code	Hexadecimal Reason Code	Meaning
08	080B	<b>Meaning:</b> Program error. Error accessing parameter list.
08	0810	<b>Meaning:</b> Program error. Caller has EUT FRR established.
08	0823	<b>Meaning:</b> Program error. Caller invoked service while dynamic address translation was disabled.
08	0824	<b>Meaning:</b> Program error. Caller invoked service but was in 24-bit addressing mode.
08	0825	<b>Meaning:</b> Program error. Caller invoked service but was not in primary ASC mode.
08	0827	<b>Meaning:</b> Program error. Reserved field in parameter list was non-zero.
08	0828	<b>Meaning:</b> Program error. Version number in parameter list is not valid.
08	0829	<b>Meaning:</b> Program error. Parameter list omits required parameters or supplies mutually exclusive parameters or provides data associated with options not selected.

---

## Example

To identify where the transactions associated with the input connect token are executing, specify:

```
IWMWQWRK SEARCH_BY=CONNTKN,CONNTKN=(R7),
          LU62_LIST=LIST1,LU62_LISTSIZE=SIZE1,
          QUERYLEN=(R9),RETCODE=RCODE,RSNCODE=RSN
```

LIST1 is a field containing the area for the list. SIZE1 is a field containing the length of the list.



---

## Chapter 63. IWMWSYSQ – Querying System Information

The purpose of this service is to query information about the systems in the sysplex that are in goal mode. The Query System Information service, IWMWSYSQ, returns a list of systems running in goal mode and information related to available CPU capacity and resource constraints.

The caller of IWMWSYSQ must provide storage to contain all of the system information. This storage area must reside in the caller's primary address space.

It is possible that the storage required by IWMWSYSQ may change such that multiple IWMWSYSQ calls are required to obtain data. IWMWSYSQ users should take this into account when obtaining the amount of storage that the IWMWSYSQ service can use.

If the caller does not provide enough storage to contain all of the system information, this service will return a return/reason code pair indicating that the input SYSINFO\_BLOCK is too small. Output data about the amount of storage required (QUERYLEN) will be set to reflect the required SYSINFO\_BLOCK size. However, no system capacity information is returned.

Applications that schedule work across multiple systems in an MVS sysplex can use this service to

- Locate the "best" (fastest or most idle) system in a sysplex for scheduling specific work
- Avoid scheduling additional work to systems already critically overloaded
- Factor WLM business importance level information into scheduling decisions

The output of this service is a data area mapped by the IWMWSYSI macro, that provides a point-in-time snapshot of each system WLM is managing in goal mode within the sysplex. A scheduling application can interpret and use this information to schedule one or more types of work to systems with specific operating characteristics. Some examples of operating characteristics you can identify with IWMWSYSQ are

- FASTEST CP SPEED - use the IWMWSYSI data area to identify the system having the fastest single CP speed.
- MULTI-PROCESSING CAPABILITY - use the IWMWSYSI data area to identify the number of online CPs on each available system.
- IDLE CAPACITY - use the IWMWSYSI data area to identify the system with the greatest idle capacity.

If a scheduling application can identify the IMPORTANCE LEVEL of the work it schedules the application can use IWMWSYSI to select the most appropriate system. IWMWSYSI provides a vector containing the amount of capacity consumed at each importance level on each system. Thus, if an application is scheduling importance level 3 work, it can use IWMWSYSI to identify the system that has the most capacity that is either idle or is handling importance level 4 or lower work.

An important use of a scheduling application is to avoid placing additional work on systems experiencing contention. IWMWSYSI provides an indicator for each system that, if on, signifies that the system should be avoided for scheduling additional work. This contention indicator is set if a auxilliary storage, fixed storage, or SQA

## IWMWSYSQ Macro

shortage exists. Also, if work to be scheduled may consume large quantities of CSA, you can use IWMWSYSI to determine the amount of CSA and ECSA that is available on each system.

Note that:

- Multiple applications may simultaneously use the same IWMWSYSQ information to make work scheduling decisions. These multiple applications will have no direct cooperation and will compete for the available systems. It is recommended that before an application schedules a large amount of work it activate a small quantity of work, wait for a built-in delay, and then use IWMWSYSQ to determine the effect of the added work before scheduling the additional work.
- Field SYSI\_CPU\_UP in macro IWMWSYSI returns the speed of an individual CP on the system, adjusted to compensate for MP effects. However, the CP speed of an LPAR is influenced by the CEC LPAR configuration. Variables such as the number of LPARs, the CP mode (shared or dedicated), capping controls, and the logical to physical CP ratio will all influence the actual CP speed. SYS\_CPU\_UP is not adjusted for such LPAR configuration effects and therefore the actual performance may differ. See the PR/SM manual for more details.

---

## Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Problem state. Any PSW key
<b>Dispatchable unit mode:</b>	Task
<b>Cross memory mode:</b>	Any PASN, any HASN, any SASN
<b>AMODE:</b>	31-bit
<b>ASC mode:</b>	Primary or access register (AR)
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks may be held.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

---

## Programming Requirements

1. The macro CVT must be included to use this macro.
2. The macro IWMYCON must be included to use this macro.
3. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
4. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, IWMSRNCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.

---

## Restrictions

None.

## Input Register Information

Before issuing the IWMWSYSQ macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

## Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Reason code if GR15 return code is non-zero
1	Used as work register by the system
2-13	Unchanged
14	Used as work registers by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

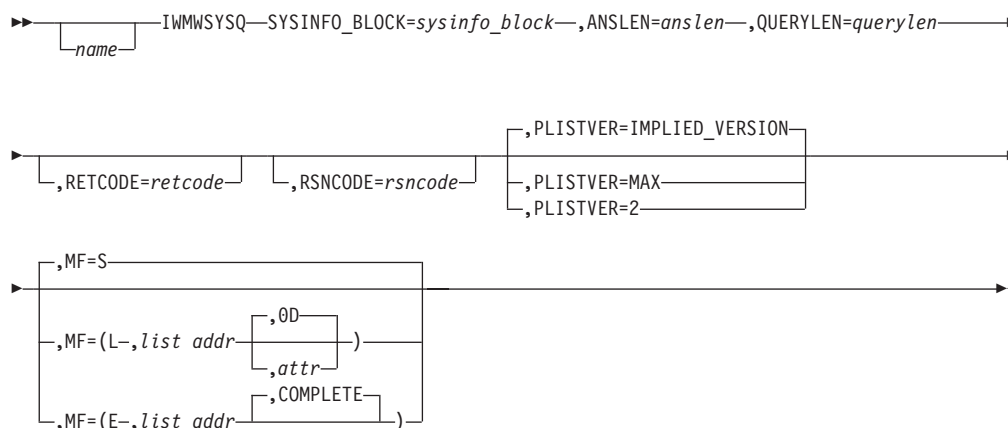
Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

## Performance Implications

None.

## Syntax

### main diagram



## Parameters

The parameters are explained as follows:

## IWMWSYSQ Macro

*name*

An optional symbol, starting in column 1, that is the name on the IWMWSYSQ macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

**,ANSLEN=anslen**

A required input parameter, which contains the length of the SYSINFO\_BLOCK in bytes.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

**,MF=S**

**,MF=(L,list addr)**

**,MF=(L,list addr,attr)**

**,MF=(L,list addr,0D)**

**,MF=(E,list addr)**

**,MF=(E,list addr,COMPLETE)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

*,list addr*

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

*,attr*

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

**,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

**,PLISTVER=IMPLIED\_VERSION**

**,PLISTVER=MAX**

**,PLISTVER=2**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.



- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify `PLISTVER=MAX` on the list form of the macro. Specifying `MAX` ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form; in this way, `MAX` ensures that the parameter list does not overwrite nearby storage.

- **2**, if you use the currently available parameters.

**To code:** Specify one of the following:

- `IMPLIED_VERSION`
- `MAX`
- A decimal value of 2

#### **,QUERYLEN=***querylen*

A required output parameter, variable which contains the output area size that must be provided by the caller to contain all of the active systems in the sysplex that are in goal mode (i.e. the amount of data returned by the IWMWSYSQ service).

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

#### **,RETCODE=***retcode*

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

#### **,RSNCODE=***rsncode*

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

#### **SYSINFO\_BLOCK=***sysinfo\_block*

A required input parameter that is to contain the address of an output area to contain information provided by this service. The format of this area is mapped by IWMWSYSI and should only be considered valid upon return code zero from this service.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a pointer field.

---

## ABEND Codes

None.

---

## Return and Reason Codes

When the IWMWSYSQ macro returns control to your program:

- GPR 15 (and *retcode*, if you coded `RETCODE`) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, if you coded `RSNCODE`) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 68. Return and Reason Codes for the IWMWSYSQ Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> IwmRetCodeOk <b>Meaning:</b> Successful completion. <b>Action:</b> None required.
4	—	<b>Equate Symbol:</b> IwmRetCodeWarning <b>Meaning:</b> Successful completion, unusual conditions noted.
4	xxxx040A	<b>Equate Symbol:</b> IwmRsnCodeOutputAreaTooSmall <b>Meaning:</b> The output area supplied is too small to receive all the available information. <b>Action:</b> None required. If necessary, reinvoke the service with an output area of sufficient size (returned in QUERYLEN) to receive all information.
4	xxxx0420	<b>Equate Symbol:</b> IwmRsnCodeSysInfoIncomplete <b>Meaning:</b> System capacity data for one or more systems running in goal mode is unavailable when the IWMWSYSQ service is invoked. <b>Action:</b> None required. If necessary, wait a few minutes and reinvoke the service to receive all information.
8	—	<b>Equate Symbol:</b> IwmRetCodeInvocError <b>Meaning:</b> Invalid invocation environment or parameters.
8	xxxx0801	<b>Equate Symbol:</b> IwmRsnCodeSrbMode <b>Meaning:</b> Caller is in SRB mode. <b>Action:</b> Avoid requesting this function while in SRB mode.
8	xxxx0803	<b>Equate Symbol:</b> IwmRsnCodeDisabled <b>Meaning:</b> Caller is disabled. <b>Action:</b> Avoid requesting this function while disabled.
8	xxxx0804	<b>Equate Symbol:</b> IwmRsnCodeLocked <b>Meaning:</b> Caller is locked. <b>Action:</b> Avoid requesting this function while locked.
8	xxxx080B	<b>Equate Symbol:</b> IwmRsnCodeBadPI <b>Meaning:</b> Error accessing parameter list. <b>Action:</b> Check for possible storage overlay.
8	xxxx0823	<b>Equate Symbol:</b> IwmRsnCodeDatoff <b>Meaning:</b> Caller invoked service while DATOFF <b>Action:</b> Avoid requesting this function in this environment.
8	xxxx0824	<b>Equate Symbol:</b> IwmRsnCodeAmode24 <b>Meaning:</b> Caller invoked service but was in 24 bit addressing mode. <b>Action:</b> Request this function only when you are in 31 bit addressing mode.

Table 68. Return and Reason Codes for the IWMWSYSQ Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0828	<p><b>Equate Symbol:</b> IwmRsnCodeBadVersion</p> <p><b>Meaning:</b> Version number in parameter list is not valid or the length specified is incorrect.</p> <p><b>Action:</b> Check for possible storage overlay of the parameter list.</p>
8	xxxx083C	<p><b>Equate Symbol:</b> IwmRsnCodeMissingAcro</p> <p><b>Meaning:</b> Required parameter list acronym (eye catcher) not found or a zero SYSINFO_BLOCK pointer is found to be associated with a non-zero ANSLEN.</p> <p><b>Action:</b> Check for possible storage overlay of the parameter list after ensuring that the acronym was correctly set.</p>
C	—	<p><b>Equate Symbol:</b> IwmRetCodeEnvError</p> <p><b>Meaning:</b> Environmental error.</p>
C	xxxx0C12	<p><b>Equate Symbol:</b> IwmRsnNoGoalModeSystems</p> <p><b>Meaning:</b> There are no goal mode systems in the sysplex</p> <p><b>Action:</b> No action required.</p>
10	—	<p><b>Equate Symbol:</b> IwmRetCodeCompError</p> <p><b>Meaning:</b> Component error.</p> <p><b>Action:</b> No action required. The function may be successful if invoked again.</p>

## Example

To query system information, specify:

```
IWMWSYSQ SYSINFO_BLOCK=SYSINFO,ANSLEN=ANSLEN,          X
          QUERYLEN=QUERYLEN,RETCODE=RC,RSNCODE=RSN
```

Where the following are declared:

```
SYSINFO DS   F           SYSINFO_BLOCK address
ANSLEN  DS   F           Length of the SYSINFO_BLOCK area
QUERYLEN DS  F           Query length
RC      DS   F           Return code
RSN     DS   F           Reason code
```



---

## Chapter 64. IWM4AEDF — Defining Dynamic Application Environments to Workload Management

The IWM4AEDF service defines dynamic application environments to WLM. The service can be used by queue manager address spaces to add new application environments after they connected to WLM and to delete the dynamic application environments before they disconnect from WLM.

Furthermore, the service can be used to define the method how server spaces should be resumed for static and dynamic application environments.

Before using this service, the caller must connect to WLM using the IWM4CON service, specifying `Work_Manager=Yes`, and `Queue_Manager=Yes`.

A queueing manager must not insert requests for a dynamic and static application environment with the same application environment name concurrently.

### Information

Note that this service was previously called IWMAEDEF for 31-bit address space only (see “IWMAEDEF — Defining Dynamic Application Environments to Workload Management” on page 765).

---

## Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Supervisor state or program key mask (PKM) allowing keys 0-7.
<b>Dispatchable unit mode:</b>	Task or SRB
<b>Cross memory mode:</b>	Any HASN, any SASN. PASN must be the address space which connected to WLM (i.e. the address space that was home when IWM4CON was issued for <code>Queue_Manager=Yes</code> , or <code>Server_Manager=Yes</code> with <code>Server_Type=Queue</code> ).
<b>AMODE:</b>	31- or 64-bit
<b>ASC mode:</b>	Primary
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks may be held.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

---

## Programming Requirements

1. The macro CVT must be included to use this macro.
2. The macro IWMYCON must be included to use this macro.
3. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
4. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, `IWMRSNCODE_MASK_CONST` defined in IWMYCON, may be used for this purpose.

## IWM4AEDF Macro

5. All character data, unless otherwise specified, is assumed to be left justified and padded with blanks on the right, as needed, to occupy the specified number of bytes.

---

## Restrictions

None.

---

## Input Register Information

Before issuing the IWM4AEDF macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

---

## Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Reason code if GR15 return code is non-zero
1	Used as work registers by the system
2-13	Unchanged
14	Used as work registers by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

---

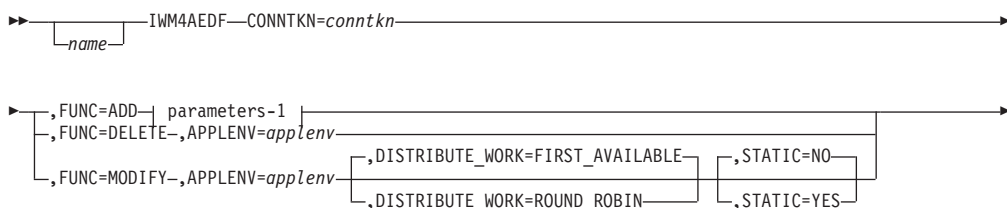
## Performance Implications

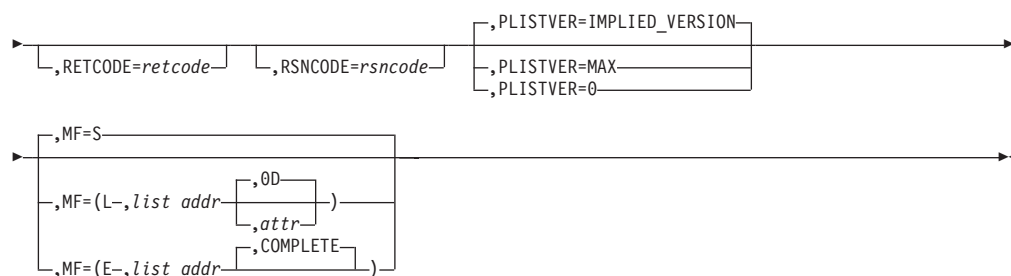
None.

---

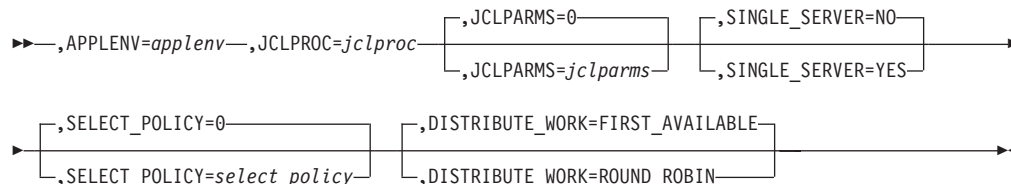
## Syntax

### main diagram





### parameters-1



## Parameters

The parameters are explained as follows:

### *name*

An optional symbol, starting in column 1, that is the name on the IWM4AEDF macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

### **,APPLENV=applenv**

When `FUNC=ADD` is specified, a required input parameter, which contains the name of the static or dynamic application environment.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32-character field.

### **CONNTKN=conntkn**

A required input parameter, which contains the connect token returned by the IWM4CON macro.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32 bit field.

### **,DISTRIBUTE\_WORK=FIRST\_AVAILABLE**

### **,DISTRIBUTE\_WORK=ROUND\_ROBIN**

When `FUNC=ADD` is specified, an optional parameter that controls how Workload Management resumes bound server spaces that are waiting for work. The default is `DISTRIBUTE_WORK=FIRST_AVAILABLE`.

### **,DISTRIBUTE\_WORK=FIRST\_AVAILABLE**

Workload Management wakes up the server space that has been suspended first (default).

### **,DISTRIBUTE\_WORK=ROUND\_ROBIN**

Workload Management wakes up the server space that has the smallest number of affinities. If there are several server spaces with the same number of affinities, workload management will start the server space with the smallest number of active server tasks.

### **,FUNC=ADD**

**,FUNC=DELETE**

**,FUNC=MODIFY**

A required parameter that indicates how the caller uses the service

**,FUNC=ADD**

indicates that the caller wants to add a dynamic application environment to WLM.

**,FUNC=DELETE**

indicates that the caller wants to delete its interest in the dynamic application environment.

**,FUNC=MODIFY**

indicates that the caller wants to redefine the method how server spaces should be resumed for static and dynamic application environments.

**,JCLPARMS=*jclparms***

**,JCLPARMS=0**

When **FUNC=ADD** is specified, an optional input parameter, which contains the parameters which are passed to the start procedure of the server manager address spaces by WLM. The default is 0.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 115-character field.

**,JCLPROC=*jclproc***

When **FUNC=ADD** is specified, a required input parameter, which contains the name of the start procedure which is used by WLM to start server manager address spaces for the application environment.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

**,MF=S**

**,MF=(L,*list addr*)**

**,MF=(L,*list addr,attr*)**

**,MF=(L,*list addr,OD*)**

**,MF=(E,*list addr*)**

**,MF=(E,*list addr,COMPLETE*)**

An optional input parameter that specifies the macro form.

Use **MF=S** to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. **MF=S** is the default.

Use **MF=L** to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the **PLISTVER** parameter may be coded with the list form of the macro.

Use **MF=E** to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

**,*list addr***

The name of a storage area to contain the parameters. For **MF=S** and **MF=E**, this can be an RS-type address or an address in register (1)-(12).



*,attr*

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

**,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

**,PLISTVER=IMPLIED\_VERSION**

**,PLISTVER=MAX**

**,PLISTVER=0**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form, when both are assembled with the same level of the system. In this way, MAX ensures that the parameter list does not overwrite nearby storage.

- **0**, if you use the currently available parameters.

**To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX
- A decimal value of 0

**,RETCODE=*retcode***

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,RSNCODE=*rsncode***

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,SELECT\_POLICY=*select\_policy***

**,SELECT\_POLICY=0**

When FUNC=ADD is specified, an optional input parameter, which tells WLM how to select work if work requests are directly routed to the server address space. Only 0,1 and 2 are valid select policies. 0 is the default which is also selected if an invalid policy is specified.

The select policy options 0,1 and 2 have the following meaning:

- 0 Default, the oldest request on either the service class or server address space queue is selected first.

## IWM4AEDF Macro

- 1 The request on the server address space queue (if present) is selected first independently of the times the requests have been inserted.
- 2 The request on the service class queue is always selected first.

The default is 0.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an one-byte field.

**,SINGLE\_SERVER=NO**

**,SINGLE\_SERVER=YES**

When FUNC=ADD is specified, an optional parameter indicating whether one or multiple server spaces should be started for the application environment. The default is SINGLE\_SERVER=NO.

**,SINGLE\_SERVER=NO**

Multiple server spaces should be started for the application environment (default).

**,SINGLE\_SERVER=YES**

Only one server space should be started for the application environment.

**,STATIC=NO**

**,STATIC=YES**

When FUNC=MODIFY is specified, an optional parameter that controls whether a static or dynamic application environment should be updated. The default is STATIC=NO.

**,STATIC=NO**

indicates that the caller wants to modify a dynamic application environment (default).

**,STATIC=YES**

indicates that the caller wants to modify a static application environment.

---

## ABEND Codes

None.

---

## Return and Reason Codes

When the IWM4AEDF macro returns control to your program:

- GPR 15 (and *retcode*, when you code RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, when you code RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 69. Return and Reason Codes for the IWM4AEDF Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> IwmRetCodeOk <b>Meaning:</b> Successful completion. <b>Action:</b> None required.
8	—	<b>Equate Symbol:</b> IwmRetCodeInvocError <b>Meaning:</b> Invalid invocation environment or parameters.

Table 69. Return and Reason Codes for the IWM4AEDF Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0803	<b>Equate Symbol:</b> IwmRsnCodeDisabled <b>Meaning:</b> Caller is disabled. <b>Action:</b> Avoid requesting this function while disabled.
8	xxxx0804	<b>Equate Symbol:</b> IwmRsnCodeLocked <b>Meaning:</b> Caller is locked. <b>Action:</b> Avoid requesting this function while locked.
8	xxxx080B	<b>Equate Symbol:</b> IwmRsnCodeBadPI <b>Meaning:</b> Error accessing parameter list. <b>Action:</b> Check for possible storage overlay.
8	xxxx0821	<b>Equate Symbol:</b> IwmRsnCodeBadConn <b>Meaning:</b> Input connect token does not pass validity checking. <b>Action:</b> Make sure to use the connect token returned by the IWM4CON service requesting Queue_Manager=Yes, or Server_Manager=Yes with Server_Type=Queuee.
8	xxxx0823	<b>Equate Symbol:</b> IwmRsnCodeDatoff <b>Meaning:</b> Caller invoked service while DATOFF <b>Action:</b> Avoid requesting this function in this environment.
8	xxxx0824	<b>Equate Symbol:</b> IwmRsnCodeAmode24 <b>Meaning:</b> Caller invoked service but was in 24 bit addressing mode. <b>Action:</b> Request this function only when you are in 31 bit addressing mode.
8	xxxx0825	<b>Equate Symbol:</b> IwmRsnCodeAscModeNotPrimary <b>Meaning:</b> Caller invoked service but was not in primary ASC mode. <b>Action:</b> Avoid requesting this function in this environment.
8	xxxx0827	<b>Equate Symbol:</b> IwmRsnCodeRsvdNot0 <b>Meaning:</b> Reserved field in parameter list was non-zero. <b>Action:</b> Check for use of keywords that are not supported by the MVS release on which the program is running.
8	xxxx0828	<b>Equate Symbol:</b> IwmRsnCodeBadVersion <b>Meaning:</b> Version number in parameter list is not valid. <b>Action:</b> Check for possible storage overlay of the parameter list.
8	xxxx083F	<b>Equate Symbol:</b> IwmRsnCodePrimaryNotOwnConn <b>Meaning:</b> Primary address space does not own the passed connect token. <b>Action:</b> Ensure that the primary address space has previously connected to WLM using the IWM4CON macro. Ensure that the connect token returned by the IWM4CON macro is passed to the IWM4AEDF macro.

## IWM4AEDF Macro

Table 69. Return and Reason Codes for the IWM4AEDF Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0840	<p><b>Equate Symbol:</b> IwmRsnCodeServiceNotEnabled</p> <p><b>Meaning:</b> Caller's space connection is not enabled for this service</p> <p><b>Action:</b> Make sure that Queue_Manager=Yes, or Server_Manager=Yes with Server_Type=Queue is specified on the IWM4CON request to enable this service.</p>
8	xxxx0842	<p><b>Equate Symbol:</b> IwmRsnCodeNoWLMConnect</p> <p><b>Meaning:</b> Caller's space disconnected from WLM during processing of the request.</p> <p><b>Action:</b> None.</p>
8	xxxx0890	<p><b>Equate Symbol:</b> IwmRsnCodeApplEnvExists</p> <p><b>Meaning:</b> The caller tried to add an application environment that has already been defined. subsystem type.</p> <p><b>Action:</b> Check whether the correct application environment name is being used. Make sure that a unique application environment name is used when adding application environments.</p>
8	xxxx0891	<p><b>Equate Symbol:</b> IwmRsnCodeApplEnvNotFound</p> <p><b>Meaning:</b> The caller tried to delete or modify an application environment that does not exist.</p> <p><b>Action:</b> Check whether the correct application environment name is being used.</p>
C	—	<p><b>Equate Symbol:</b> IwmRetCodeEnvError</p> <p><b>Meaning:</b> Environmental error.</p>
C	xxxx0C01	<p><b>Equate Symbol:</b> IwmRsnCodeNoStg</p> <p><b>Meaning:</b> Storage is not available for the request.</p> <p><b>Action:</b> Contact your system programmer. There is a common storage shortage.</p>
10	—	<p><b>Equate Symbol:</b> IwmRetCodeCompError</p> <p><b>Meaning:</b> Component error.</p> <p><b>Action:</b> Contact your system programmer.</p>

## Example

To add a dynamic application environment, specify the following:

```

IWM4AEDF CONNTKN=CONNTOKEN,           X
          FUNC=ADD,                     X
          APPLENV=APPLENV               X
          JCLPROC=JCLPROC               X
          RETCODE=RC,RSNCODE=RSN
*
* Storage areas
*
CONNTOKEN DS  FL4           Contains the connect token
*                               associated with the use of WLM
*                               Queuing services as returned by
*                               IWM4CON
APPLENV DS    CL32          Contains the application

```

*			environment name
JCLPROC	DS	CL8	Contains the name of the
*			start procedure
RC	DS	F	Return code
RSN	DS	F	Reason code



---

## Chapter 65. IWM4CON – Connecting to Workload Management

The purpose of this service is to connect a calling address space to WLM. This service returns a token which is needed to invoke other services. This service can be used to:

- Request that WLM Work Management services be available to the connecting address space and optionally to pass topology information to WLM.
- Request that WLM Work Queuing services be available to the connecting address space.
- Request that WLM Work Execution services be available to the connecting address space.
- Request that WLM Work Balancing services be available to the connecting address space.
- Request that WLM export and import services be available to the connecting address space.

Note that:

- The space which is connected is the current home address space.
- Only a single connection is allowed to be active for a given address space at any given time.
- For each connected task/space, WLM will establish a dynamic resource manager (RESMGR) to be associated with the current task/space. When it receives control, it will free any accumulated resources and delete any enclaves associated with the connect token. This implies that the resource manager will logically perform the disconnect function and the connect token can no longer be passed to WLM services.

### Information

Note that this service was previously called IWMCONN for 31-bit address space only (see “IWMCONN – Connecting to Workload Management” on page 773).

---

## Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	For WORK_MANAGER=YES or ROUTER=YES, QUEUE_MANAGER=YES or EXPTIMPT=YES, supervisor state or program key mask (PKM) allowing keys 0-7.  For SERVER_MANAGER=YES, problem state with any PSW key.
<b>Dispatchable unit mode:</b>	Task
<b>Cross memory mode:</b>	Non-XMEM when input key is a user key or SERVER_MANAGER = YES, otherwise XMEM, any P,S,H.
<b>AMODE:</b>	31- or 64-bit
<b>ASC mode:</b>	Primary
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks may be held.

**Control parameters:** Control parameters must be in the primary address space.

---

### Programming Requirements

1. Make sure no EUT FRRs are established.
2. The macro CVT must be included to use this macro.
3. The macro IWMYCON must be included to use this macro.
4. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
5. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, IWMRSNCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.
6. All character data, unless otherwise specified, is assumed to be left justified and padded with blanks on the right, as needed, to occupy the specified number of bytes.

---

### Restrictions

1. This macro may only be used on MVS/ESA (version 3 or higher), NOT versions 1 or 2 of MVS.
2. This macro may not be used during task/address space termination.
3. Only a single connection is allowed to be active for a given address space at any given time.
4. Specification of both Queue\_Manager=Yes, and Server\_Manager=Yes requires that Server\_Type=Queue. Specification of Server\_Type=Routing is rejected.
5. Specification of both Router=Yes, and Server\_Manager=Yes requires that Server\_Type=Routing. Specification of Server\_Type=Queue is rejected.
6. If the callers recovery routine should get control as a result of requesting this service, the function cannot be guaranteed to be complete. It is possible that a token has been saved in the parameter list where the connect token would reside upon successful completion. This token may be passed to IWM4DIS to prevent the address space from being disabled from future IWM4CON requests, but the token should not be used for other services. IWM4DIS in these circumstances may give a warning return code indicating that no connection was established, however.
7. If the key specified on IWM4CON is a user key (8-F) or SERVER\_MANAGER=YES was specified, then the caller must be in non-cross-memory mode (P=S=H)
8. While not a restriction for IWM4CON, it should be noted that when the key specified is a user key (8-F), the connect token may only be passed to IWMCLSFY, IWMRPT, or IWMMNTFY services, when the current primary matches primary at the time IWM4CON is invoked.

---

### Input Register Information

Before issuing the IWM4CON macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.



---

## Output Register Information

When control returns to the caller, the GPRs contain:

<b>Register</b>	<b>Contents</b>
<b>0</b>	Reason code if GR15 return code is non-zero
<b>1</b>	Used as work registers by the system
<b>2-13</b>	Unchanged
<b>14</b>	Used as work registers by the system
<b>15</b>	Return code

When control returns to the caller, the ARs contain:

<b>Register</b>	<b>Contents</b>
<b>0-1</b>	Used as work registers by the system
<b>2-13</b>	Unchanged
<b>14-15</b>	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

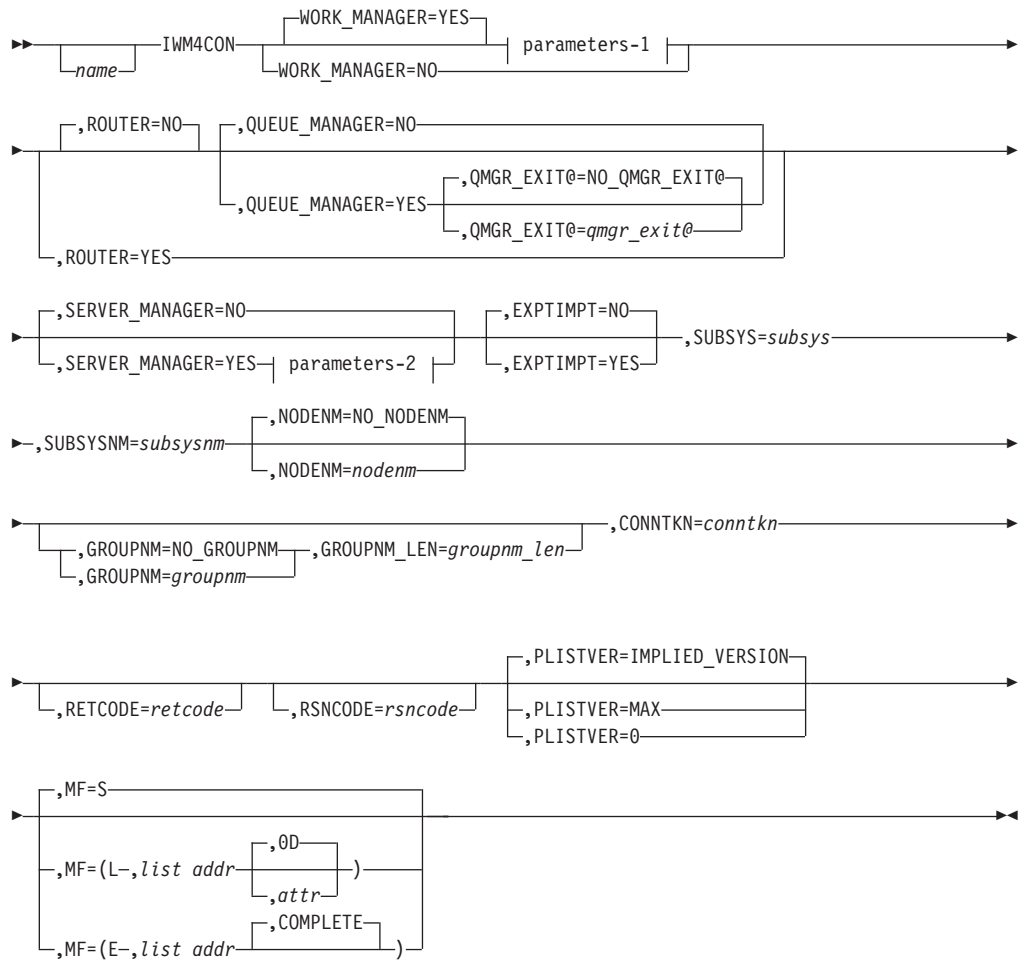
---

## Performance Implications

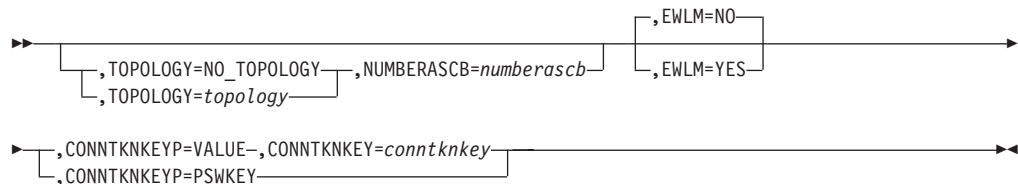
None.

# Syntax

## main diagram

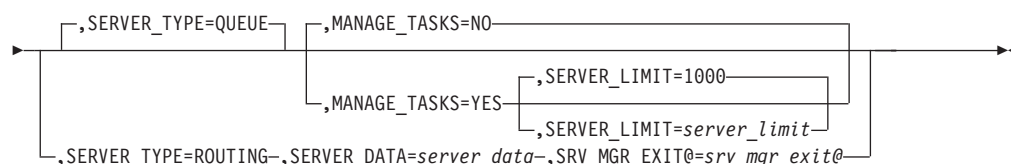


## parameters-1



## parameters-2





## Parameters

The parameters are explained as follows:

### *name*

An optional symbol, starting in column 1, that is the name on the IWM4CON macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

### **,APPLENV=applenv**

When SERVER\_MANAGER=YES is specified, a required input parameter, which contains the application environment under which work requests are served.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32-character field.

### **,CONNTKN=conntkn**

A required output parameter, which will receive the connect token.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32 bit field.

### **,CONNTKNKEY=conntknkey**

When CONNTKNKEYP=VALUE and WORK\_MANAGER=YES are specified, a required input parameter, which contains the key for which the various branch entry services using the CONNTKN returned by IWM4CON must have PSW update authority. These other services include Classify (IWMCLSFY), Report (IWMRPT), Notify (IWMNTFY). Create (IWM4MCRE) is a PC interface and hence is excluded. The low order 4 bits (bits 4-7) contain the key value. The high order 4 bits (bits 0-3) must be zeros.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8 bit field.

### **,CONNTKNKEYP=VALUE**

### **,CONNTKNKEYP=PSWKEY**

When WORK\_MANAGER=YES is specified, a required parameter, which describes how the input key should be obtained.

### **,CONNTKNKEYP=VALUE**

indicates that the key is being passed explicitly via CONNTKNKEY.

### **,CONNTKNKEYP=PSWKEY**

indicates that the current PSW key should be used.

### **,DYNAMIC=NO**

### **,DYNAMIC=YES**

When SERVER\_MANAGER=YES is specified, an optional parameter indicating whether the server manager connects to a dynamic or static application environment. The default is DYNAMIC=NO.

### **,DYNAMIC=NO**

The server manager connects to a static application environment. This is the default.

### **,DYNAMIC=YES**

The server manager connects to a dynamic application environment.

### **,EWLM=NO**

### **,EWLM=YES**

When WORK\_MANAGER=YES is specified, an optional parameter, which indicates if this work manager intends to participate in cross-platform enterprise workload management (EWLM). The default is EWLM=NO.

### **,EWLM=NO**

The work manager interacts only with WLM and no interaction with EWLM takes place. This is the default.

### **,EWLM=YES**

The work manager participates in cross-platform enterprise workload management and interacts with EWLM.

### **,EXPTIMPT=NO**

### **,EXPTIMPT=YES**

An optional parameter indicating whether the space needs access to the export and import services (IWMEXPT, IWMUEXPT, IWMIMPT, IWMUIMPT). The default is EXPTIMPT=NO.

### **,EXPTIMPT=NO**

The connecting address space will not use the export and import services.

### **,EXPTIMPT=YES**

The connecting address space will use the export and import services.

### **,GROUPNM=*groupnm***

### **,GROUPNM=NO\_GROUPNM**

An optional input parameter, which contains the name of an application group, for example, a group of similar or cooperating subsystem instances. A group name can be up to 255 characters long. Provision of a data area initialized to all blanks is equivalent to specification of NO\_GROUPNM. The default is NO\_GROUPNM. This indicates that no group name is passed.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a character field.

### **,GROUPNM\_LEN=*groupnm\_len***

When GROUPNM=*groupnm* is specified, a required input parameter, which contains the length of the group name. A group name can be up to 255 characters long.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

### **,MANAGE\_TASKS=NO**

### **,MANAGE\_TASKS=YES**

When SERVER\_TYPE=QUEUE and SERVER\_MANAGER=YES are specified, an optional parameter indicating that WLM will manage the server instances (tasks), selecting work from a work queue.

If YES is specified, the caller must use service IWMSINF to obtain the number of server instances to start from WLM.

The meaning of PARALLEL\_EU changes in this case. PARALLEL\_EU is only used to determine the number of tasks to start if the application environment cannot be managed by WLM. Otherwise PARALLEL\_EU can be used to limit the number of server tasks to start initially.

The server can define the SERVER\_LIMIT parameter to specify a limit for the number of server tasks supported by the application.

**,MANAGE\_TASKS=NO**

The connecting address space starts the number of server instances as provided with parallel\_eu.

**,MANAGE\_TASKS=YES**

The connecting address space uses IWMSINF to obtain the number of server instances to start from WLM.

**,MF=S**

**,MF=(L,list addr)**

**,MF=(L,list addr,attr)**

**,MF=(L,list addr,0D)**

**,MF=(E,list addr)**

**,MF=(E,list addr,COMPLETE)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

*,list addr*

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

*,attr*

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

**,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

**,NODENM=nodenm**

**,NODENM=NO\_NODENM**

An optional input parameter, which contains the node name to be used for classifying work requests when Work\_Manager=Yes is specified or taken as default. The node name identifies a specific subcomponent of the generic subsystem type.

When Server\_Manager=Yes and Server\_Type=Queue is specified, the node name should match the node name specified on the corresponding Connect for the Queue\_Manager, for example, all servers associated with the Queue\_Manager have identical node names.

## IWM4CON Macro

If a product chooses to use both `Work_Manager=Yes` and `Server_Manager=Yes` on a single invocation of IWM4CON for a space, then the rules for `Server_Manager` apply, for example, the node name refers to the node name of the space playing the role of `Queue_Manager`.

If the caller connects to the WLM work queuing services, the combination of the subsystem type, node name and the subsystem name must be unique to that MVS system. Node name can be omitted. The default is `NO_NODENM`.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

### **,NUMBERASCB=numberascb**

When `TOPOLOGY=topology` and `WORK_MANAGER=YES` are specified, a required input parameter, which contains the number of ASCBs in the list passed via `xTOPOLOGY`. While there is no restriction on the number of entries in the list, the current support will only look at the first 10 entries. The number specified must be positive (hence also non-zero).

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

### **,PARALLEL\_EU=parallel\_eu**

When `SERVER_MANAGER=YES` is specified, a required input parameter, which contains the maximum number of tasks (TCBs) within the address space which will be used to concurrently process distinct work requests if `MANAGE_TASKS=YES` is not in effect. When `Select (IWM4SSL)` is used to obtain a work request, which might then be passed to another task (TCB) for processing under a `Begin (IWM4STBG)` environment, this count represents the number of tasks (TCBs) which can be running concurrently against these work requests, i.e. the number of concurrent `Begin` environments. It is important that this count represents the actual number of tasks (TCBs) which can be utilized, and not merely some approximate upper bound, as this value will influence system algorithms.

If `MANAGE_TASKS=YES` is in effect, the application environment managed by WLM `PARALLEL_EU` is not used. In this case the parameter is only used as described above if no procedure name was defined for the application environment.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

### **,PLISTVER=IMPLIED\_VERSION**

#### **,PLISTVER=MAX**

#### **,PLISTVER=0**

An optional input parameter that specifies the version of the macro. `PLISTVER` determines which parameter list the system generates. `PLISTVER` is an optional input parameter on all forms of the macro, including the list form. When using `PLISTVER`, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the `PLISTVER` parameter, `IMPLIED_VERSION` is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify `PLISTVER=MAX` on the list form of the macro. Specifying `MAX` ensures that

the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form, when both are assembled with the same level of the system. In this way, MAX ensures that the parameter list does not overwrite nearby storage.

- 0, if you use the currently available parameters.

**To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX
- A decimal value of 0

**,QMGR\_EXIT@=qmgr\_exit@**

**,QMGR\_EXIT@=NO\_QMGR\_EXIT@**

When QUEUE\_MANAGER=YES and ROUTER=NO are specified, an optional input parameter that is to contain the address of the Queue Manager Connect Exit to be invoked when the system wishes to inform the queue manager of actions it should perform. The exit will be called in enabled, unlocked TCB mode with no FRRs set, but may be called in a cross-memory environment. The mapping of the parameter list for the exit and its invocation environment is given by the list form of the IWMQCXIT macro.

The system may chose to discontinue calling the exit upon repetitive abnormal completions, i.e. where the system recovery routine is percolated to from an error within the exit. The exit must be callable from any address space and remain available after the queue manager disconnects or terminates. The default is NO\_QMGR\_EXIT@, which indicates that no queue manager exit is provided.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a pointer field.

**,QUEUE\_MANAGER=NO**

**,QUEUE\_MANAGER=YES**

When ROUTER=NO is specified, an optional parameter indicating that WLM Work Queuing services be available to the connecting address space. For example:

- Insert (IWM4QIN)
- Delete (IWM4QDE)

If YES is specified, the combination of the subsystem type and the subsystem name must be unique to that MVS system. The default is QUEUE\_MANAGER=NO.

**,QUEUE\_MANAGER=NO**

The connecting address space will not use the WLM Work Queuing services.

**,QUEUE\_MANAGER=YES**

The connecting address space will be using the WLM Work Queuing services.

**,RETCODE=retcode**

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,ROUTER=NO**

**,ROUTER=YES**

An optional parameter, which describes whether recommendations for sysplex

routing to servers associated with the same subsystem type and name are requested. The default is ROUTER=NO.

### **,ROUTER=NO**

indicates that recommendations for sysplex routing via IWMSRFSV are not required.

### **,ROUTER=YES**

indicates that recommendations for sysplex routing via IWMSRFSV is required. Note that only server spaces which have the same Subsystem type and name AND which specified Server\_Type=Routing are considered when IWMSRFSV is invoked.

If YES is specified, the combination of the subsystem type and the subsystem name must be unique to that MVS system.

### **,RSNCODE=*rsncode***

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

### **,SERVER\_DATA=*server\_data***

When SERVER\_TYPE=ROUTING and SERVER\_MANAGER=YES are specified, a required input parameter, which contains whatever data is needed to uniquely identify the server when recommended by MVS through use of the IWMSRFSV interface. The structure of this data is undefined to MVS, and will be returned to the program invoking IWMSRFSV when the server is returned.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32-character field.

### **,SERVER\_LIMIT=*server\_limit***

#### **,SERVER\_LIMIT=1000**

When MANAGE\_TASKS=YES, SERVER\_TYPE=QUEUE and SERVER\_MANAGER=YES are specified, an optional input parameter indicating the architectural limit of the application for the number of server instances which can be supported.

This parameter can be used to tell WLM the upper limit up to which WLM will recommend to start server instances. If the parameter is omitted or is set higher than 1000, WLM will use 1000 as upper limit instead. The default is 1000.

**To code:** Specify the RS-type address of a fullword field.

### **,SERVER\_MANAGER=NO**

#### **,SERVER\_MANAGER=YES**

An optional parameter indicating whether the space needs access to a family of services specified by SERVER\_TYPE. The default is SERVER\_MANAGER=NO.

#### **,SERVER\_MANAGER=NO**

The connecting address space will not use any of the various server related WLM services documented under SERVER\_TYPE.

#### **,SERVER\_MANAGER=YES**

The connecting address space will be acting in the role of a server and needs access to the family of services specified by SERVER\_TYPE.

Specification of both Queue\_Manager=Yes, and Server\_Manager=Yes requires that Server\_Type=Queue. Specification of Server\_Type=Routing is rejected.



Specification of both Router=Yes, and Server\_Manager=Yes requires that Server\_Type=Routing. Specification of Server\_Type=Queue is rejected.

**,SERVER\_TYPE=QUEUE**

**,SERVER\_TYPE=ROUTING**

When SERVER\_MANAGER=YES is specified, an optional parameter, which describes what type of services are used by the server. The default is SERVER\_TYPE=QUEUE.

**,SERVER\_TYPE=QUEUE**

indicates that the server selects work from a queue, and thus requests that WLM Work Execution services be available to the connecting address space. For example:

- Select (IWM4SSL)
- Begin (IWM4STBG)
- End (IWM4STEN)

The server also has the WLM Work Queuing services available to the connecting address space when the corresponding Queue Manager with the same subsystem type and name is active on the same MVS image (see following macros for macro-specific restrictions). For example:

- Insert (IWM4QIN)
- Delete (IWM4QDE)

**,SERVER\_TYPE=ROUTING**

indicates that the server receives work by way of routing, and may be selected by the IWMSRFSV (Find Server) macro interface. Note that the space which invokes the IWMSRFSV service must Connect with Router=Yes.

Termination of the router with the same subsystem type and name on the same MVS image will not cause notification to the server to terminate. This coordination, if required, must be handled through a different protocol than use of Connect.

**,SRV\_MGR\_EXIT@=srv\_mgr\_exit@**

When SERVER\_TYPE=ROUTING and SERVER\_MANAGER=YES are specified, a required input parameter that is to contain the address of the Server Manager Connect Exit to be invoked when the system wishes to inform the server of actions it should perform. This exit will be called in SRB mode, with a non cross-memory environment, where HASN=SASN=PASN=HASN at the time IWM4CON was invoked. The mapping of the parameter list for the exit and its invocation environment is given by the list form of the IWMSCXIT macro.

Note that it may be possible for the exit to be called before the caller has received control back from IWM4CON. The exit or any program it drives (synchronously or asynchronously) must synchronize with the program issuing IWM4CON to ensure that IWM4CON has returned a connect token prior to issuing IWM4DIS (disconnect) or any other services that need the connect token.

The system may cause the space to become ineligible to be recommended by IWMSRFSV upon repetitive errors in calling the exit specified. The exit must be callable from the server address space and remain available after the server manager disconnects or the connecting server TCB terminates. The exit need not persist upon memory termination of the server.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a pointer field.

### **,SUBSYS=***subsys*

A required input parameter, which contains the generic subsystem type (e.g. IMS, CICS, etc.). When WORK\_MANAGER=YES is specified, this is the primary category under which classification rules are grouped.

If the caller connects to the WLM work queueing services by specifying QUEUE\_MANAGER=YES, or requests sysplex routing by specifying ROUTER=YES, the combination of the subsystem type and the subsystem name must be unique to that MVS system.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 4-character field.

### **,SUBSYSNM=***subsysnm*

A required input parameter, which contains the subsystem name to be used for classifying work requests when Work\_Manager=Yes is specified or taken as default. The subsystem name identifies a specific instance of the generic subsystem type.

When Server\_Manager=Yes and Server\_Type=Queue is specified, the subsystem name should match the subsystem name specified on the corresponding Connect for the Queue\_Manager, i.e. all servers associated with the Queue\_Manager have identical subsystem names.

When Server\_Manager=Yes and Server\_Type=Routing is specified, the subsystem name should match the subsystem name specified on the corresponding Connect for Router=Yes, i.e. all servers associated with the Router have identical subsystem names.

If a product chooses to use both Work\_Manager=Yes and Server\_Manager=Yes on a single invocation of IWM4CON for a space, then the rules for Server\_Manager apply, i.e. the subsystem name refers to the subsystem name of the space playing the role of Queue\_Manager or Router.

If the caller connects to the WLM work queueing services, or to sysplex routing services, the combination of the subsystem type and the subsystem name must be unique to that MVS system.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

### **,TOPOLOGY=***topology*

### **,TOPOLOGY=NO\_TOPOLOGY**

When WORK\_MANAGER=YES is specified, an optional input parameter, which represents a list of ASCB addresses for the address spaces which comprise the subsystem. This list should ONLY include address spaces which do NOT surface as the current home address space when IWM4MINI or IWMMRELA are used to establish the delay monitoring environments, but that may participate as dispatchable units (TCBs or SRBs) in serving work requests. If the current primary or home space is a space not surfacing in a monitoring environment and its execution can affect the response time of work flowing through the subsystem, then it should appear in the list. Neither current primary nor current home are defaults. While there are no limits on the number of address spaces, this information is less precise than that provided by monitoring environments. The default is NO\_TOPOLOGY, which indicates that no topology information was passed.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a character field.

### **WORK\_MANAGER=**YES

**WORK\_MANAGER=NO**

An optional parameter indicating that WLM Work Management services be available to the connecting address space. For example:

- Classify (IWMCLSFY)
- Report (IWMRPT)
- Notify (IWMMNTFY)
- Enclave Create (IWM4ECRE)
- Modify Connect (IWMWMCON)

If NO is specified, the above services cannot be used, except for the form of Notify that does not pass an input connect token. The default is WORK\_MANAGER=YES.

**WORK\_MANAGER=YES**

The connecting address space will be using the WLM Work Management services.

**WORK\_MANAGER=NO**

The connecting address space will not use the WLM Work Management services. Specifying this keyword may reduce the use of system resources.

**ABEND Codes**

None.

**Return and Reason Codes**

When the IWM4CON macro returns control to your program:

- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, if you coded RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 70. Return and Reason Codes for the IWM4CON Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> IwmRetCodeOk <b>Meaning:</b> Successful completion. <b>Action:</b> None required.
4	—	<b>Equate Symbol:</b> IwmRetCodeWarning <b>Meaning:</b> Successful completion, unusual conditions noted.
4	xxxx0401	<b>Equate Symbol:</b> IwmRsnCodeNoWLM <b>Meaning:</b> System does not support work manager services. This return code is only set when the MVS release is prior to MVS/ESA SP5.1.0 or prior to HBB6603 for IWM4CON QUEUE_MANAGER/SERVER_MANAGER services. <b>Action:</b> None required.
8	—	<b>Equate Symbol:</b> IwmRetCodeInvocError <b>Meaning:</b> Invalid invocation environment or parameters.

Table 70. Return and Reason Codes for the IWM4CON Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0801	<b>Equate Symbol:</b> IwmRsnCodeSrbMode <b>Meaning:</b> Caller is in SRB mode. <b>Action:</b> Avoid requesting this function while in SRB mode.
8	xxxx0802	<b>Equate Symbol:</b> IwmRsnCodeXmemUserKeyTkn <b>Meaning:</b> Caller is in cross-memory mode while the token was requested in user key. <b>Action:</b> Avoid requesting this function while in cross-memory mode.
8	xxxx0803	<b>Equate Symbol:</b> IwmRsnCodeDisabled <b>Meaning:</b> Caller is disabled. <b>Action:</b> Avoid requesting this function while disabled.
8	xxxx0804	<b>Equate Symbol:</b> IwmRsnCodeLocked <b>Meaning:</b> Caller is locked. <b>Action:</b> Avoid requesting this function while locked.
8	xxxx080B	<b>Equate Symbol:</b> IwmRsnCodeBadPI <b>Meaning:</b> Error accessing parameter list. <b>Action:</b> Check for possible storage overlay.
8	xxxx0810	<b>Equate Symbol:</b> IwmRsnCodeEutFrr <b>Meaning:</b> Caller has EUT FRR established. <b>Action:</b> Avoid requesting this function with an EUT FRR set.
8	xxxx0812	<b>Equate Symbol:</b> IwmRsnCodeBadAscb <b>Meaning:</b> Bad ASCB address passed. <b>Action:</b> Check for possible storage overlay.
8	xxxx0823	<b>Equate Symbol:</b> IwmRsnCodeDatoff <b>Meaning:</b> Caller invoked service while DATOFF <b>Action:</b> Avoid requesting this function in this environment.
8	xxxx0824	<b>Equate Symbol:</b> IwmRsnCodeAmode24 <b>Meaning:</b> Caller invoked service but was in 24 bit addressing mode. <b>Action:</b> Request this function only when you are in 31 bit addressing mode.
8	xxxx0825	<b>Equate Symbol:</b> IwmRsnCodeAscModeNotPrimary <b>Meaning:</b> Caller invoked service but was not DAT on Primary ASC mode. <b>Action:</b> Avoid requesting this function in this environment.
8	xxxx0826	<b>Equate Symbol:</b> IwmRsnCodeTaskTerm <b>Meaning:</b> Caller invoked service while task termination is in progress for the TCB associated with the owner. <b>Action:</b> Avoid requesting this function in this environment.

Table 70. Return and Reason Codes for the IWM4CON Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0827	<p><b>Equate Symbol:</b> IwmRsnCodeRsvdNot0</p> <p><b>Meaning:</b> Reserved field in parameter list was non-zero.</p> <p><b>Action:</b> Check for possible storage overlay of the parameter list.</p>
8	xxxx0828	<p><b>Equate Symbol:</b> IwmRsnCodeBadVersion</p> <p><b>Meaning:</b> Version number in parameter list is not valid.</p> <p><b>Action:</b> Check for possible storage overlay of the parameter list.</p>
8	xxxx0829	<p><b>Equate Symbol:</b> IwmRsnCodeBadOptions</p> <p><b>Meaning:</b> Parameter list omits required parameters or supplies mutually exclusive parameters or provides data associated with options not selected.</p> <p><b>Action:</b> Check for possible storage overlay of the parameter list.</p>
8	xxxx082C	<p><b>Equate Symbol:</b> IwmRsnCodeBadNumberAscb</p> <p><b>Meaning:</b> NUMBERASCB variable is not a positive value.</p> <p><b>Action:</b> Check for possible storage overlay of the parameter list or variable.</p>
8	xxxx082E	<p><b>Equate Symbol:</b> IwmRsnCodeConnectExists</p> <p><b>Meaning:</b> Connect has already been established for the current home address space.</p> <p><b>Action:</b> Avoid requesting this function when a connection already exists.</p>
8	xxxx0840	<p><b>Equate Symbol:</b> IwmRsnCodeServiceNotEnabled</p> <p><b>Meaning:</b> Requested connection type cannot be established in the current execution environment. This occurs when SERVER_MANAGER=YES is specified and the program is run as a batch job in a WLM-managed job class.</p> <p><b>Action:</b> Run the program as a started task.</p>
8	xxxx0841	<p><b>Equate Symbol:</b> IwmRsnCodeXMemMode</p> <p><b>Meaning:</b> Caller is in cross memory mode.</p> <p><b>Action:</b> Invoke the function in non-cross memory mode.</p>
8	xxxx0847	<p><b>Equate Symbol:</b> IwmRsnCodeOtherSpaceConnected</p> <p><b>Meaning:</b> Another address space with the same subsystem type and name is connected to WLM on the MVS image and has the role of queue manager or router.</p> <p><b>Action:</b> Avoid requesting this function with duplicate values.</p>
8	xxxx0849	<p><b>Equate Symbol:</b> IwmRsnCodeWLMservBadAPPL</p> <p><b>Meaning:</b> The application environment name (APPLENV=) specified is not the same as the one used by WLM to start the server.</p> <p><b>Action:</b> Verify that the start parameters for the application environment are coded correctly in the WLM ISPF application, and that those parameters are used by the started JCL procedure.</p>

Table 70. Return and Reason Codes for the IWM4CON Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx084A	<p><b>Equate Symbol:</b> IwmRsnCodeWLMservBadSSN</p> <p><b>Meaning:</b> The subsystem name (SUBSYSNM=) specified is not the same as the one used by WLM to start the server.</p> <p><b>Action:</b> Verify that the start parameters for the application environment are coded correctly in the WLM ISPF application, and that those parameters are used by the started JCL procedure.</p>
8	xxxx084B	<p><b>Equate Symbol:</b> IwmRsnCodeWLMservBadSST</p> <p><b>Meaning:</b> The subsystem type (SUBSYS=) specified is not the same as the one used by WLM to start the server.</p> <p><b>Action:</b> Verify that the start parameters for the application environment are coded correctly in the WLM ISPF application, and that those parameters are used by the started JCL procedure.</p>
8	xxxx084D	<p><b>Equate Symbol:</b> IwmRsnCodeNotAuthConnect</p> <p><b>Meaning:</b> The caller must be supervisor state or have PSW key mask 0-7 authority to connect to the requested WLM services.</p> <p><b>Action:</b> Avoid requesting this function in this environment.</p>
8	xxxx084E	<p><b>Equate Symbol:</b> IwmRsnCodeWlmServBadType</p> <p><b>Meaning:</b> For WLM started servers, the SERVER_TYPE= is not the one used to start the server.</p> <p><b>Action:</b> Specify the correct SERVER_TYPE.</p>
8	xxxx0853	<p><b>Equate Symbol:</b> IwmRsnCodeWlmQmBadType</p> <p><b>Meaning:</b> There is a queue manager or router environment of the specified subsystem name, but of a different type than that specified by the caller.</p> <p><b>Action:</b> Verify that the option for queue manager/router is specified correctly on IWM4CON. If the option is correct, then server address spaces for a different Server_Type exist and must terminate before the current space may connect as a queue manager or router.</p>
8	xxxx0855	<p><b>Equate Symbol:</b> IwmRsnCodeBadNumEUMax</p> <p><b>Meaning:</b> PARALLEL_EU variable is greater than the maximum of 1000.</p> <p><b>Action:</b> Specify a value between 1 and 1000.</p>
8	xxxx0856	<p><b>Equate Symbol:</b> IwmRsnCodeBadNumEUMin</p> <p><b>Meaning:</b> PARALLEL_EU variable is less than the minimum of 1.</p> <p><b>Action:</b> Specify a value between 1 and 1000.</p>
8	xxxx085C	<p><b>Equate Symbol:</b> IwmRsnCodeWrongNumEU</p> <p><b>Meaning:</b> Caller invoked service with a PARALLEL_EU value which is different from the PARALLEL_EU of existing servers in the application environment</p> <p><b>Action:</b> Ensure that all servers in the application environment specify the same PARALLEL_EU value.</p>

Table 70. Return and Reason Codes for the IWM4CON Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0873	<p><b>Equate Symbol:</b> IwmRsnCodeWrongSrvLmt</p> <p><b>Meaning:</b> Caller invoked service with a SERVER_LIMIT parameter setting which is different from the SERVER_LIMIT of existing servers in the application environment</p> <p><b>Action:</b> Ensure that all servers in the application environment specify the same SERVER_LIMIT value.</p>
8	xxxx0874	<p><b>Equate Symbol:</b> IwmRsnCodeWrongMngTsk</p> <p><b>Meaning:</b> Caller invoked service with a MANAGE_TASKS parameter setting which is different from the MANAGE_TASKS of existing servers in the application environment</p> <p><b>Action:</b> Ensure that all servers in the application environment specify the same MANAGE_TASKS value.</p>
8	xxxx0878	<p><b>Equate Symbol:</b> IwmRsnCodeBadNumLimitMax</p> <p><b>Meaning:</b> Caller invoked service with a SERVER_LIMIT parameter setting which exceeds the maximum number of tasks which can be started in a server address space.</p> <p><b>Action:</b> Correct number or do not specify SERVER_LIMIT parameter in order to use the default.</p>
8	xxxx0879	<p><b>Equate Symbol:</b> IwmRsnCodeBadNumLimitMin</p> <p><b>Meaning:</b> Caller invoked service with a SERVER_LIMIT parameter setting which is less than what has been defined on the PARALLEL_EU parameter.</p> <p><b>Action:</b> Ensure that SERVER_LIMIT is always greater or equal to PARALLEL_EU.</p>
8	xxxx087A	<p><b>Equate Symbol:</b> IwmRsnCodeNoQServer</p> <p><b>Meaning:</b> The MANAGE_TASKS parameter is not allowed when QUEUE_SERVER=YES has been specified.</p> <p><b>Action:</b> Ensure to use the parameters correctly.</p>
8	xxxx088E	<p><b>Equate Symbol:</b> IwmRsnCodeWlmServBadSSND</p> <p><b>Meaning:</b> For WLM started servers, the NODENM= is not the one used to start the server.</p> <p><b>Action:</b> Specify the correct NODENM.</p>
8	xxxx088F	<p><b>Equate Symbol:</b> IwmRsnCodeApplNotSSN</p> <p><b>Meaning:</b> The application environment name is defined for use by a different subsystem node.</p> <p><b>Action:</b> Check whether the correct application environment name is being used.</p>
C	—	<p><b>Equate Symbol:</b> IwmRetCodeEnvError</p> <p><b>Meaning:</b> Environmental error.</p>
C	xxxx0C01	<p><b>Equate Symbol:</b> IwmRsnCodeNoStg</p> <p><b>Meaning:</b> Storage is not available for the request.</p> <p><b>Action:</b> There is a storage shortage. The function may work successfully at a later time.</p>

Table 70. Return and Reason Codes for the IWM4CON Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
C	xxxx0C09	<p><b>Equate Symbol:</b> IwmRsnCodeNoResmgr</p> <p><b>Meaning:</b> Resource manager could not be established.</p> <p><b>Action:</b> No action required. This condition may be due to a storage shortage condition.</p>
C	xxxx0C14	<p><b>Equate Symbol:</b> IwmRsnCodeNoWorkShutDown</p> <p><b>Meaning:</b> No work selected. Caller is to shutdown.</p> <p><b>Action:</b> The server should shut down (terminate).</p>
C	xxxx0C19	<p><b>Equate Symbol:</b> IwmRsnCodeNotSecAuthConnect</p> <p><b>Meaning:</b> The caller is not authorized by SAF to connect to WLM with SERVER_MANAGER=YES.</p> <p><b>Action:</b> The security administrator must grant access to the appropriate resource.</p>
C	xxxx0C1A	<p><b>Equate Symbol:</b> IwmRsnCodeApplNotDefined</p> <p><b>Meaning:</b> The application environment name is not defined in the active WLM policy.</p> <p><b>Action:</b> Check whether the correct application environment name is being used. If so, a service administrator must define the application environment in the WLM service definition.</p>
C	xxxx0C1B	<p><b>Equate Symbol:</b> IwmRsnCodeApplNotSST</p> <p><b>Meaning:</b> The application environment name is defined for use by a different subsystem type in the active WLM policy.</p> <p><b>Action:</b> Check whether the correct application environment name is being used. If so, a service administrator must change the application environment in the WLM service definition to specify the correct subsystem type.</p>
C	xxxx0C1F	<p><b>Equate Symbol:</b> IwmRsnCodeServerExists</p> <p><b>Meaning:</b> A server exists for the specified application environment which only allows 1 such server in the sysplex.</p> <p><b>Action:</b> Check whether the correct application environment name is being used. If so and the current server is shutting down, a retry may be successful after a delay.</p>
C	xxxx0C22	<p><b>Equate Symbol:</b> IwmRsnCodeApplEnvQuiesced</p> <p><b>Meaning:</b> The specified application environment has been quiesced, server cannot be started for the request.</p> <p><b>Action:</b> Restart the application environment and then retry the request.</p>
10	—	<p><b>Equate Symbol:</b> IwmRetCodeCompError</p> <p><b>Meaning:</b> Component error.</p> <p><b>Action:</b> No action required. The function may be successful if invoked again.</p>



---

## Examples

To connect to workload management with a key value of 8, and a list of 7 address spaces involved in processing work, specify:

```
IWM4CON  SUBSYS=GENSUB,SUBSYSNM=SUBNAME,  
         TOPOLOGY=LISTASCBS,NUMBERASCB=NUMSPACE  
         CONNTKN=CTKN,CONNTKNKEYP=VALUE,CONNTKNKEY=KEY,  
         RETCODE=RC,RSNCODE=RSN,
```

Where the following are declared:

GENSUB	DS	CL4	Generic subsystem type
SUBNAME	DS	CL8	Subsystem name
LISTASCBS	DS	CL28	List of 7 address spaces
NUMSPACE	DC	F'7'	Number of ASCBs
CTKN	DS	FL4	Connect token
KEY	DS	XL1	Key value



---

## Chapter 66. IWM4DIS – Disconnecting from Workload Management

IWM4DIS allows the caller to disconnect from the workload management services. This means that the input connect token can no longer be passed to workload management macros such as IWMCLSFY and IWMRPT. When a program disconnects, any enclaves associated with the input connect token are deleted from the system. Any SRBs running in the enclave are run as preemptible SRBs at the priority of the home address space. Any enclave TCBs are converted to ordinary TCBs.

You should issue this macro once during shutdown of the connecting address space.

### Information

Note that this service was previously called IWMDISC for 31-bit address space only (see “IWMDISC – Disconnecting from Workload Management” on page 792).

---

## Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	When the corresponding Connect (IWM4CON) invocation specified WORK_MANAGER=YES or QUEUE_MANAGER=YES, ROUTER=YES, or EXPTIMPT=YES, supervisor state or program key mask (PKM) allowing keys 0-7.  When the corresponding Connect (IWM4CON) invocation specified WORK_MANAGER=NO, QUEUE_MANAGER=NO, ROUTER=NO, EXPTIMPT=NO, and SERVER_MANAGER=YES, problem state with any PSW key.
<b>Dispatchable unit mode:</b>	Task or SRB  When the corresponding Connect (IWM4CON) invocation specified SERVER_MANAGER=YES, task mode.
<b>Cross memory mode:</b>	The current Home address space must be the same as Home when the corresponding Connect was invoked. Any PASN, any SASN.  When the corresponding Connect (IWM4CON) invocation specified SERVER_MANAGER=YES, non-cross memory mode, P=S=H.
<b>AMODE:</b>	31- or 64-bit
<b>ASC mode:</b>	Primary
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks may be held.  When the corresponding Connect (IWM4CON) invocation specified SERVER_MANAGER=YES, SERVER_TYPE=ROUTING, NO FRRs may be set.

**Control parameters:** Control parameters must be in the primary address space.

---

### Programming Requirements

1. The macro CVT must be included to use this macro.
2. The macro IWMYCON must be included to use this macro.
3. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
4. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, IWMSNCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.

---

### Restrictions

1. No FRRs may be set when calling to disconnect a space which is connected as a server manager with server type = routing.
2. If the key specified on IWM4CON was a user key (8-F), then the following must ALL be true:
  - Caller must be in non-cross-memory mode (P=S=H). This implies that the current primary must match the primary at the time that IWM4CON was invoked. Running in a subspace is not supported.
  - Must be in TCB mode (not SRB)
  - Current TCB must match the TCB at the time that IWM4CON was invoked.
3. This service should not be invoked while in a RTM termination routine (resource manager) for the TCB owning the connect token since MVS will have its own resource cleanup routine and unpredictable results would occur. It is legitimate to use this service while in a recovery routine, however, or in mainline processing.

---

### Input Register Information

Before issuing the IWM4DIS macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

---

### Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Reason code if GR15 return code is non-zero
1	Used as work registers by the system
2-13	Unchanged
14	Used as work registers by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

---

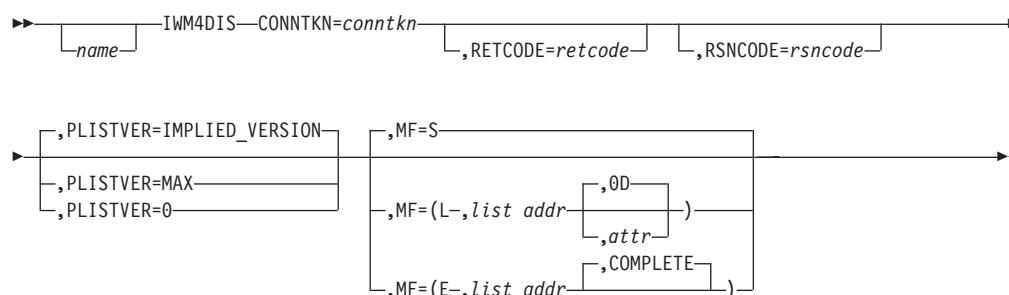
## Performance Implications

None.

---

## Syntax

### main diagram




---

## Parameters

The parameters are explained as follows:

### *name*

An optional symbol, starting in column 1, that is the name on the IWM4DIS macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

### CONNTKN=*conntkn*

A required input parameter, which contains the connect token for the environment to be disconnected.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32 bit field.

### ,MF=S

,MF=(L,*list addr*)

,MF=(L,*list addr*,*attr*)

,MF=(L,*list addr*,**0D**)

,MF=(E,*list addr*)

,MF=(E,*list addr*,**COMPLETE**)

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

*,list addr*

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

*,attr*

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

**,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

**,PLISTVER=IMPLIED\_VERSION**

**,PLISTVER=MAX**

**,PLISTVER=0**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form, when both are assembled with the same level of the system. In this way, MAX ensures that the parameter list does not overwrite nearby storage.

- **0**, if you use the currently available parameters.

**To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX
- A decimal value of 0

**,RETCODE=*retcode***

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,RSNCODE=*rsncode***

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

## ABEND Codes

None.

## Return and Reason Codes

When the IWM4DIS macro returns control to your program:

- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, if you coded RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 71. Return and Reason Codes for the IWM4DIS Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> <code>IwmRetCodeOk</code> <b>Meaning:</b> Successful completion. <b>Action:</b> None required.
4	—	<b>Equate Symbol:</b> <code>IwmRetCodeWarning</code> <b>Meaning:</b> Successful completion, unusual conditions noted.
4	xxxx0401	<b>Equate Symbol:</b> <code>IwmRsnCodeNoWLM</code> <b>Meaning:</b> System does not support work manager services. This return code is only set when the MVS release is prior to MVS/ESA SP5.1.0. <b>Action:</b> None required.
4	xxxx0409	<b>Equate Symbol:</b> <code>IwmRsnCodeNoConn</code> <b>Meaning:</b> Input connection token does not reflect an active connection to WLM. <b>Action:</b> None required.
8	—	<b>Equate Symbol:</b> <code>IwmRetCodeInvocError</code> <b>Meaning:</b> Invalid invocation environment or parameters.
8	xxxx0801	<b>Equate Symbol:</b> <code>IwmRsnCodeSrbMode</code> <b>Meaning:</b> Caller is in SRB mode. <b>Action:</b> Avoid requesting this function while in SRB mode.
8	xxxx0802	<b>Equate Symbol:</b> <code>IwmRsnCodeXmemUserKeyTkn</code> <b>Meaning:</b> Caller is in cross-memory mode while the token was obtained in a user key. <b>Action:</b> Avoid requesting this function while in cross-memory mode.
8	xxxx0803	<b>Equate Symbol:</b> <code>IwmRsnCodeDisabled</code> <b>Meaning:</b> Caller is disabled. <b>Action:</b> Avoid requesting this function while disabled.

Table 71. Return and Reason Codes for the IWM4DIS Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0804	<b>Equate Symbol:</b> IwmRsnCodeLocked <b>Meaning:</b> Caller is locked. <b>Action:</b> Avoid requesting this function while locked.
8	xxxx0809	<b>Equate Symbol:</b> IwmRsnCodeSrbUserKeyTkn <b>Meaning:</b> Caller is in SRB mode, while the token was obtained in a user key (8-F). <b>Action:</b> Avoid requesting this function in SRB mode for tokens associated with user key.
8	xxxx080A	<b>Equate Symbol:</b> IwmRsnCodeTcbNotOwnerUserKeyTkn <b>Meaning:</b> Current TCB is not the owner, while the token was obtained in a user key (8-F). <b>Action:</b> Avoid requesting this function under a TCB other than the owner for a token associated with user key.
8	xxxx080B	<b>Equate Symbol:</b> IwmRsnCodeBadPI <b>Meaning:</b> Error accessing parameter list. <b>Action:</b> Check for possible storage overlay.
8	xxxx0810	<b>Equate Symbol:</b> IwmRsnCodeEutFrr <b>Meaning:</b> Caller has EUT FRR established. <b>Action:</b> Avoid requesting this function with an EUT FRR set.
8	xxxx0823	<b>Equate Symbol:</b> IwmRsnCodeDatoff <b>Meaning:</b> Caller invoked service while DATOFF <b>Action:</b> Avoid requesting this function in this environment.
8	xxxx0824	<b>Equate Symbol:</b> IwmRsnCodeAmode24 <b>Meaning:</b> Caller invoked service but was in 24 bit addressing mode. <b>Action:</b> Request this function only when you are in 31 bit addressing mode.
8	xxxx0825	<b>Equate Symbol:</b> IwmRsnCodeAscModeNotPrimary <b>Meaning:</b> Caller invoked service but was not DAT on Primary ASC mode. <b>Action:</b> Avoid requesting this function in this environment.
8	xxxx0826	<b>Equate Symbol:</b> IwmRsnCodeTaskTerm <b>Meaning:</b> Caller invoked service while task termination is in progress for the TCB associated with the owner. <b>Action:</b> Avoid requesting this function in this environment.
8	xxxx0827	<b>Equate Symbol:</b> IwmRsnCodeRsvdNot0 <b>Meaning:</b> Reserved field in parameter list was non-zero. <b>Action:</b> Check for possible storage overlay of the parameter list.



Table 71. Return and Reason Codes for the IWM4DIS Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0828	<p><b>Equate Symbol:</b> lwmRsnCodeBadVersion</p> <p><b>Meaning:</b> Version number in parameter list is not valid.</p> <p><b>Action:</b> Check for possible storage overlay of the parameter list.</p>
8	xxxx082F	<p><b>Equate Symbol:</b> lwmRsnCodeWrongHome</p> <p><b>Meaning:</b> Caller invoked the service from the wrong home address space.</p> <p><b>Action:</b> Invoke the function with the correct home address space.</p>
8	xxxx0841	<p><b>Equate Symbol:</b> lwmRsnCodeXMemMode</p> <p><b>Meaning:</b> Caller is in cross memory mode.</p> <p><b>Action:</b> Invoke the function in non-cross memory mode.</p>
8	xxxx084D	<p><b>Equate Symbol:</b> lwmRsnCodeNotAuthConnect</p> <p><b>Meaning:</b> The caller must be supervisor state or have PSW key mask 0-7 authority to disconnect from the requested WLM services.</p> <p><b>Action:</b> Avoid requesting this function in this environment.</p>
10	—	<p><b>Equate Symbol:</b> lwmRetCodeCompError</p> <p><b>Meaning:</b> Component error.</p> <p><b>Action:</b> No action required. The function may be successful if invoked again.</p>



---

## Chapter 67. IWM4ECRE – Create an Enclave

The purpose of this service is to create an enclave where possibly multiple SRBs and/or TCBs may be simultaneously executing or scheduled. For the duration of each enclave, all SRBs and TCBs associated with the enclave are treated as part of a single work request. All SRBs and/or TCBs associated with the enclave accumulate service as a single entity and are managed as a single entity. The address spaces where enclave SRBs are dispatched, as defined by the ENV= parameter of IEAMSCHD, should be non-swappable.

For more information about managing address spaces with enclaves, see “Performance Management of Address Spaces with Enclaves” on page 45.

**Note:** An address space must be non-swappable if it has enclave SRBs dispatched and SYSEVENT ENCASSOC has not been issued.

For TYPE=INDEPENDENT enclaves, a new work business unit of work is created and classified according to the input Connect token’s subsystem type and subsystem name, along with whatever other attributes are passed via the Classify parameter list. The current home address space is considered the owner.

For TYPE=DEPENDENT enclaves, SRM considers the enclave to be part of the current home address space’s transaction, which then becomes the owning space. This space need not be connected to WLM via IWM4CON.

For TYPE=MONENV enclaves, SRM considers the enclave to be part of the address space’s transaction which is delayed according to the input monitoring environment, as set when IWM4MINI or IWMMRELA was used. This space becomes the owning space. This space need not be connected to WLM via IWM4CON.

For both TYPE=MONENV and TYPE=DEPENDENT enclaves, SRM will change the enclave to TYPE=INDEPENDENT if the owning address space’s transaction ends.

For both TYPE=MONENV and TYPE=DEPENDENT enclaves, SRM will check the enclave for period switch when the owning address space is swapped in. If the owning address space is swapped out SRM will continue to accumulate service for any enclaves owned by the space, but will not check the address space and any owned enclave for period switch until the address space is swapped in again. The presence of enclaves does not make the address space appear to be ready from an SRM point of view.

Enclaves are deleted if the owning address space terminates. TYPE=INDEPENDENT enclaves are deleted if the owning address space disconnects or the TCB which connected terminates.

Enclaves should only be created when this environment is ready for execution, and should not be used when prolonged queueing effects are possible prior to the scheduling of the first SRB (IEAMSCHD) or the first task join (IWMEJOIN). “Prolonged” would certainly include times measured in seconds. The service allows the caller to pass the queueing time prior to creation of the enclave so that this may be separately reported.

**Information**

Note that this service was previously called IWMECREA for 31-bit address space only (see “IWMECREA – Create an Enclave” on page 798).

---

**Environment**

The requirements for the caller are:

<b>Minimum authorization:</b>	Supervisor state or program key mask (PKM) allowing keys 0-7.
<b>Dispatchable unit mode:</b>	Task or SRB
<b>Cross memory mode:</b>	Any PASN, any HASN, any SASN
<b>AMODE:</b>	31- or 64-bit
<b>ASC mode:</b>	Primary or access register (AR)
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks may be held.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

---

**Programming Requirements**

1. The macro IWMYCON must be included to use this macro.
2. The macro CVT must be included to use this macro.
3. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
4. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, IWMSRNCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.
5. All character data, unless otherwise specified, is assumed to be left justified and padded with blanks on the right, as needed, to occupy the specified number of bytes.

---

**Restrictions**

1. The Connect token from the input classify parameter list must be owned by the current home address space and must be associated with a system key (0-7), as specified on IWM4CON. The Classify parameter list and hence the Connect token is only relevant for TYPE=INDEPENDENT enclaves.
2. Since this service may only be used by system-like code, some validity checking on the parameter list is not performed. These checks would only be needed if the macro were not used to invoke the service routine.
3. The variable length fields associated with the classify parameter list (the classify parameter list is only relevant for certain options) given by the CLSFY keyword have the following limitations in addition to those documented in IWMCLSFY:
  - SUBSYSPM is limited to 255 bytes
  - COLLECTION is limited to 18 bytes
  - CORRELATION is limited to 12 bytes
4. When TYPE(MONENV) is specified the following apply:

- If the key specified on IWM4ECRE was a user key (8-F), then primary or home addressability must exist to the performance block IWM4MCRE obtained. This condition is satisfied by ensuring that the current primary or home address space matches primary (=home) at the time that IWM4ECRE was invoked.
  - The caller must serialize to prevent any delay monitoring services from being invoked concurrently for the environment represented by the monitoring token.
  - Only limited checking is done against the input monitoring token.
  - TYPE=MONENV enclaves cannot be created for report-only monitoring environments.
5. This macro may only be used on z/OS R2 or higher levels for EXSTARTDEFER keyword.

---

## Input Register Information

Before issuing the IWM4ECRE macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

---

## Output Register Information

When control returns to the caller, the GPRs contain:

<b>Register</b>	<b>Contents</b>
<b>0</b>	Reason code if GR15 return code is non-zero
<b>1</b>	Used as work registers by the system
<b>2-13</b>	Unchanged
<b>14</b>	Used as work registers by the system
<b>15</b>	Return code

When control returns to the caller, the ARs contain:

<b>Register</b>	<b>Contents</b>
<b>0-1</b>	Used as work registers by the system
<b>2-13</b>	Unchanged
<b>14-15</b>	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

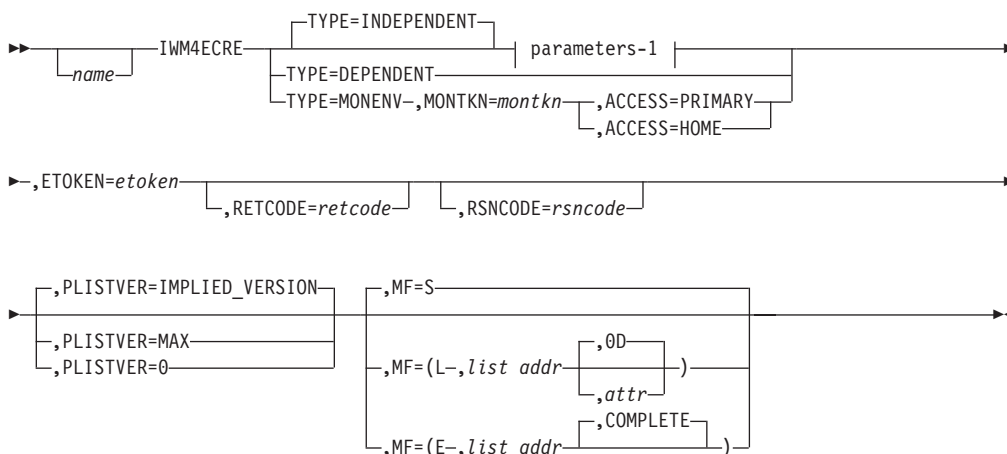
---

## Performance Implications

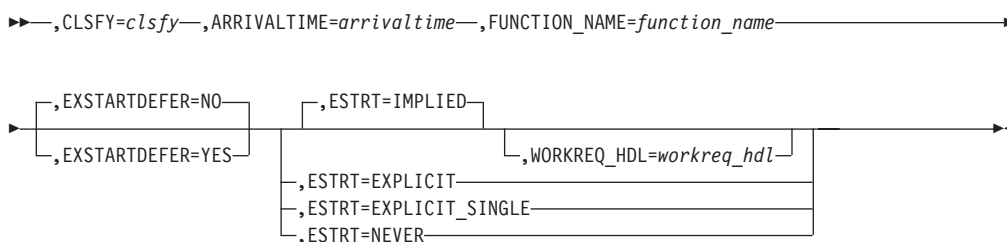
None.

## Syntax

### main diagram



### parameters-1



## Parameters

The parameters are explained as follows:

### *name*

An optional symbol, starting in column 1, that is the name on the IWM4ECRE macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

### **,ACCESS=PRIMARY**

### **,ACCESS=HOME**

When TYPE=MONENV is specified, a required parameter, which describes how to access the monitoring environment.

### **,ACCESS=PRIMARY**

indicates that the monitoring environment can be accessed in the caller's primary address space. This would be appropriate if the monitoring environment was established (by IWM4MCRE) to be used by routines in a specific system key or if it was established to be used in a specific user key in the current primary.

### **,ACCESS=HOME**

indicates that the monitoring environment must be accessed in the home address space, which is not the caller's primary address space. This would be appropriate if the monitoring environment was established (by IWM4MCRE) for use by a specific user key.

**,ARRIVALTIME=arrivaltime**

When TYPE=INDEPENDENT is specified, a required input parameter, which contains the work arrival time in STCK format. This is the time at which the business work request is considered to have arrived and from which point the system evaluates elapsed time for the work request.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 64 bit field.

**,CLSFY=clsfy**

When TYPE=INDEPENDENT is specified, a required input parameter, which contains the classification information in the format of the parameter list for IWMCLSFY. NOTE that this name is the data area name, not its pointer. IWMCLSFY MF(M) should be used to initialize the area prior to invocation of IWM4ECRE. The Classify parameter list must be from MVS/ESA SP5.2.0 or later.

Note that the variable length fields associated with the classify parameter list given by the CLSFY keyword have the following limitations in addition to those documented in IWMCLSFY:

- SUBSYSPM is limited to 255 bytes
- COLLECTION is limited to 18 bytes
- CORRELATION is limited to 12 bytes

**To code:** Specify the RS-type address, or address in register (2)-(12), of a character field.

**,ESTRT=IMPLIED****,ESTRT=EXPLICIT****,ESTRT=EXPLICIT\_SINGLE****,ESTRT=NEVER**

When TYPE=INDEPENDENT is specified, an optional parameter, which denotes how the work manager indicates the start and end point of an EWLM work request when participating in cross-platform enterprise workload management (EWLM).

**,ESTRT=IMPLIED**

If the work manager previously connected to WLM with IWM4CON EWLM=YES, a work request is started implicitly when the enclave is created. If IWMESTOP was not invoked before, the work request will be stopped implicitly when the enclave is deleted. ESTRT=IMPLIED is the default.

**,ESTRT=EXPLICIT**

The work manager indicates the start and end point of an EWLM work request by invoking the services IWMESTRT and IWMESTOP. Note that this option is only meaningful, if the work manager previously connected to WLM with IWM4CON EWLM=YES.

**,ESTRT=EXPLICIT\_SINGLE**

Indicates the same as option ESTRT=EXPLICIT and, in addition, the application ensures that only one work request is active. No nested calls to IWMESTRT are allowed. If this option is specified the CPU consumption on all EWLM enclave services (IWMEGCOR, IWMESTRT, IWMESTOP, IWMEBLK, IWMEUBLK) will be reduced. If ESTRT=EXPLICIT\_SINGLE is specified on IWMECREA, the application must also add the EWLMODE=EXPLICIT\_SINGLE parameter on all calls to IWMEGCOR, IWMESTRT, IWMESTOP, IWMEBLK and IWMEUBLK. If this parameter is used, the application has some restrictions on all calls to IWMEGCOR,

## IWM4ECRE Macro

IWMESTRT, IWMESTOP, IWMEBLK and IWMEUBLK. Refer to the corresponding macro descriptions for details.

### **,ESTRT=NEVER**

Indicates that this enclave will never use any EWLM-related enclave services (IWMEGCOR, IWMESTRT, IWMESTOP, IWMEBLK, IWMEUBLK) after the enclave has been created, even if the work manager has registered (IWM4CON or IWMCONN) with EWLM=YES. Also, IWM4ECRE will not start an EWLM work request on the enclave and will not do any EWLM-related processing.

### **,ETOKEN=*etoken***

A required output parameter, which will receive the enclave token.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

### **,EXSTARTDEFER=NO**

### **,EXSTARTDEFER=YES**

When TYPE=INDEPENDENT is specified, an optional parameter, which indicates whether the enclave execution start time should begin when the first IWM4STBG or IWMEJOIN is executed. The time between enclave create and the first IWM4STBG or IWMEJOIN is assumed to be the queue time. The default is EXSTARTDEFER=NO

### **,EXSTARTDEFER=NO**

indicates that the enclave execution start time should not begin when the first IWM4STBG or IWMEJOIN is executed.

### **,EXSTARTDEFER=YES**

indicates that the enclave execution start time should begin when the first IWM4STBG or IWMEJOIN is executed.

### **,FUNCTION\_NAME=*function\_name***

When TYPE=INDEPENDENT is specified, a required input parameter, which contains the descriptive name for the function for which the enclave was created.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

### **,MF=S**

### **,MF=(L,*list addr*)**

### **,MF=(L,*list addr,attr*)**

### **,MF=(L,*list addr,OD*)**

### **,MF=(E,*list addr*)**

### **,MF=(E,*list addr,COMPLETE*)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant



code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

**,list *addr***

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

**,attr**

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

**,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

**,MONTKN=*montkn***

When TYPE=MONENV is specified, a required input parameter, which contains the delay monitoring token which describes the current business unit of work. If the monitoring environment is related to an address space, then it must be the current home address space.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32 bit field.

**,PLISTVER=IMPLIED\_VERSION**

**,PLISTVER=MAX**

**,PLISTVER=0**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form, when both are assembled with the same level of the system. In this way, MAX ensures that the parameter list does not overwrite nearby storage.

- **0**, if you use the currently available parameters.

**To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX
- A decimal value of 0

**,RETCODE=*retcode***

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

## IWM4ECRE Macro

**,RSNCODE=rsncode**

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**TYPE=INDEPENDENT**

**TYPE=DEPENDENT**

**TYPE=MONENV**

An optional parameter, which indicates the type of enclave being created. The default is TYPE=INDEPENDENT.

**TYPE=INDEPENDENT**

indicates that the enclave represents a new business unit of work with its own business objectives.

**TYPE=DEPENDENT**

indicates that the enclave represents a continuation of the business unit of work represented by the current home address space.

**TYPE=MONENV**

indicates that the enclave represents a continuation of the business unit of work represented by the input monitoring environment.

**,WORKREQ\_HDL=workreq\_hdl**

When ESTRT=IMPLIED and TYPE=INDEPENDENT are specified, an optional output parameter that will receive the handle which represents the work request. The application must pass this handle to the other work request services IWMESTOP, IWMEBLK, IWMEUBLK, and IWMEGCOR.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

---

## ABEND Codes

None.

---

## Return and Reason Codes

When the IWM4ECRE macro returns control to your program:

- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, if you coded RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 72. Return and Reason Codes for the IWM4ECRE Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> IwmRetCodeOk <b>Meaning:</b> Successful completion. <b>Action:</b> None required.
8	—	<b>Equate Symbol:</b> IwmRetCodeInvocError <b>Meaning:</b> Invalid invocation environment or parameters.

Table 72. Return and Reason Codes for the IWM4ECRE Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0803	<p><b>Equate Symbol:</b> IwmRsnCodeDisabled</p> <p><b>Meaning:</b> Caller is disabled.</p> <p><b>Action:</b> Avoid requesting this function while disabled.</p>
8	xxxx0804	<p><b>Equate Symbol:</b> IwmRsnCodeLocked</p> <p><b>Meaning:</b> Caller is locked.</p> <p><b>Action:</b> Avoid requesting this function while locked.</p>
8	xxxx080B	<p><b>Equate Symbol:</b> IwmRsnCodeBadPI</p> <p><b>Meaning:</b> Error accessing parameter list.</p> <p><b>Action:</b> Check for possible storage overlay.</p>
8	xxxx0820	<p><b>Equate Symbol:</b> IwmRsnCodeBadMonEnv</p> <p><b>Meaning:</b> Monitoring environment does not pass short form verification.</p> <p><b>Action:</b> Check for possible storage overlay.</p>
8	xxxx0821	<p><b>Equate Symbol:</b> IwmRsnCodeBadConn</p> <p><b>Meaning:</b> Connect token from the input classify parameter list does not pass validity checking.</p> <p><b>Action:</b> Check for possible storage overlay.</p>
8	xxxx0824	<p><b>Equate Symbol:</b> IwmRsnCodeAmode24</p> <p><b>Meaning:</b> Caller invoked service but was in 24 bit addressing mode.</p> <p><b>Action:</b> Request this function only when you are in 31 bit addressing mode.</p>
8	xxxx0828	<p><b>Equate Symbol:</b> IwmRsnCodeBadVersion</p> <p><b>Meaning:</b> Version number in parameter list is not valid.</p> <p><b>Action:</b> Check for possible storage overlay of the parameter list.</p>
8	xxxx0836	<p><b>Equate Symbol:</b> IwmRsnCodeMaxEnclave</p> <p><b>Meaning:</b> Enclave could not be created because the enclave limit has been reached.</p> <p><b>Action :</b> Check for possible problems wherein enclaves are not being deleted as expected or excessive numbers of enclaves are being created in a loop.</p>
8	xxxx0837	<p><b>Equate Symbol:</b> IwmRsnCodeUserKeyConntkn</p> <p><b>Meaning:</b> Connect token from the input classify parameter list is associated with a user key.</p> <p><b>Action:</b> Invoke the function with a token associated with a system key.</p>
8	xxxx0838	<p><b>Equate Symbol:</b> IwmRsnCodeClisfyAreaTooBig</p> <p><b>Meaning:</b> Input area associated with classification information is larger than supported.</p> <p><b>Action:</b> Invoke the function with an area of the proper size. Check for possible storage overlay.</p>

## IWM4ECRE Macro

Table 72. Return and Reason Codes for the IWM4ECRE Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0839	<p><b>Equate Symbol:</b> IwmRsnCodeClisfyPITooSmall</p> <p><b>Meaning:</b> Input Classify parameter list is too small.</p> <p><b>Action:</b> Invoke the function with an area of the proper size. Check for possible storage overlay.</p>
8	xxxx083B	<p><b>Equate Symbol:</b> IwmRsnCodeHomeNotOwnConn</p> <p><b>Meaning:</b> Home address space does not own the connect token from the input classify parameter list.</p> <p><b>Action:</b> Invoke the function with the correct home address space.</p>
8	xxxx0840	<p><b>Equate Symbol:</b> IwmRsnCodeServiceNotEnabled</p> <p><b>Meaning:</b> Caller's space connection is not enabled for this service.</p> <p><b>Action:</b> Avoid requesting this function under the input connection. IWM4CON options must be specified previously to enable this service.</p>
8	xxxx085D	<p><b>Equate Symbol:</b> IwmRsnCodeMonenvNotHome</p> <p><b>Meaning:</b> The input monitoring environment is related to an address space other than home.</p> <p><b>Action:</b> None required.</p>
8	xxxx0894	<p><b>Equate Symbol:</b> IwmRsnCodeInvalidEWLMCorr</p> <p><b>Meaning:</b> Passed classification information contains an EWLM correlator (EWLM_CORR) that does not pass validity checking. The architected ARM correlator length field in the first two bytes of the EWLM_CORR is either less than 4 (X'0004') or greater than 512 (X'0200').</p> <p><b>Action:</b> Check the specification of the EWLM correlator in the classification information.</p>
8	xxxx0895	<p><b>Equate Symbol:</b> IwmRsnCodeEWLMSErvNotEnabled</p> <p><b>Meaning:</b> Service is not enabled because caller invoked the IWM4CON service with EWLM=NO.</p> <p><b>Action:</b> Specify the parameter WORKREQ_HDL only when connected with EWLM=YES.</p>
C	—	<p><b>Equate Symbol:</b> IwmRetCodeEnvError</p> <p><b>Meaning:</b> Environmental error.</p>
C	xxxx0C01	<p><b>Equate Symbol:</b> IwmRsnCodeNoStg</p> <p><b>Meaning:</b> Storage is not available for the request.</p> <p><b>Action:</b> There is a storage shortage. The function may work successfully at a later time.</p>
C	xxxx0C0C	<p><b>Equate Symbol:</b> IwmRsnCodeClassifyFail</p> <p><b>Meaning:</b> Received a non-zero return code from the classification service, IWMCLSFY.</p> <p><b>Action:</b> No action required. Reinvoking the function later may succeed.</p>

Table 72. Return and Reason Codes for the IWM4ECRE Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
C	xxxx0C0D	<p><b>Equate Symbol:</b> IwmRsnCodeBadClsfy</p> <p><b>Meaning:</b> Classification apparently can not access the current policy, possibly due to a policy switch in progress.</p> <p><b>Action:</b> Invoke the function when the conditions are alleviated.</p>
C	xxxx0C20	<p><b>Equate Symbol:</b> IwmRsnCodeDepClassifyFail</p> <p><b>Meaning:</b> Unable to obtain classification attributes for a dependent enclave.</p> <p><b>Action:</b> None required.</p>
C	xxxx0C21	<p><b>Equate Symbol:</b> IwmRsnCodeNoMonEnvErr</p> <p><b>Meaning:</b> Input monitoring token indicates no monitoring environment was established.</p> <p><b>Action:</b> None required.</p>
10	—	<p><b>Equate Symbol:</b> IwmRetCodeCompError</p> <p><b>Meaning:</b> Component error.</p> <p><b>Action:</b> No action required. The function may be successful if invoked again.</p>



---

## Chapter 68. IWM4EDEL – Delete an Enclave

The purpose of this service is to delete an enclave, so that no SRBs or TCBs exist within the enclave and no new SRBs may be scheduled into the enclave, nor may any TCBs join the enclave. Some residual enclave-related CPU time will not be accounted back to the work request whenever active enclave SRBs/TCBs were present at the time IWM4EDEL is invoked. SRBs scheduled to the enclave which have not completed will be converted to ordinary preemptable SRBs. TCBs joined to the enclave which have not completed will be converted to ordinary TCBs.

If IWM4EDEL is invoked for an enclave which is registered, the enclave is considered only logically deleted while all its functionality stays in place. Physical deletion is deferred until all interested parties have deregistered the enclave. The caller does not receive any notice when the physical deletion of the enclave is done.

When an enclave is deleted, the work request is considered to have finished and all related resource accounting will be finalized.

IWM4EDEL cannot be used to delete a foreign enclave. The IWMUIMPT macro must be used instead.

### Information

Note that this service was previously called IWMEDEL for 31-bit address space only (see “IWMEDELE – Delete an Enclave” on page 809).

---

## Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Supervisor state or program key mask (PKM) allowing keys 0-7.
<b>Dispatchable unit mode:</b>	Task or SRB
<b>Cross memory mode:</b>	Any PASN, any HASN, any SASN
<b>AMODE:</b>	31- or 64-bit
<b>ASC mode:</b>	Primary or access register (AR)
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks may be held. FRR environments may be established.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

---

## Programming Requirements

1. The macro CVT must be included to use this macro.
2. The macro IWMYCON must be included to use this macro.
3. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
4. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded

## IWM4EDEL Macro

from comparison with the reason code values described above. The constant, IWMRSNCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.

5. All character data, unless otherwise specified, is assumed to be left justified and padded with blanks on the right, as needed, to occupy the specified number of bytes.
6. Since this service may only be used by system-like code, some validity checking on the parameter list is not performed. These checks would only be needed if the macro were not used to invoke the service routine.

---

## Restrictions

This macro supports multiple versions. Some keywords are only supported by certain versions. Refer to the PLISTVER parameter description for further information.

---

## Input Register Information

Before issuing the IWM4EDEL macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

---

## Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Reason code if GR15 return code is non-zero
1	Used as work register by the system
2-13	Unchanged
14	Used as work registers by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

---

## Performance Implications

None.

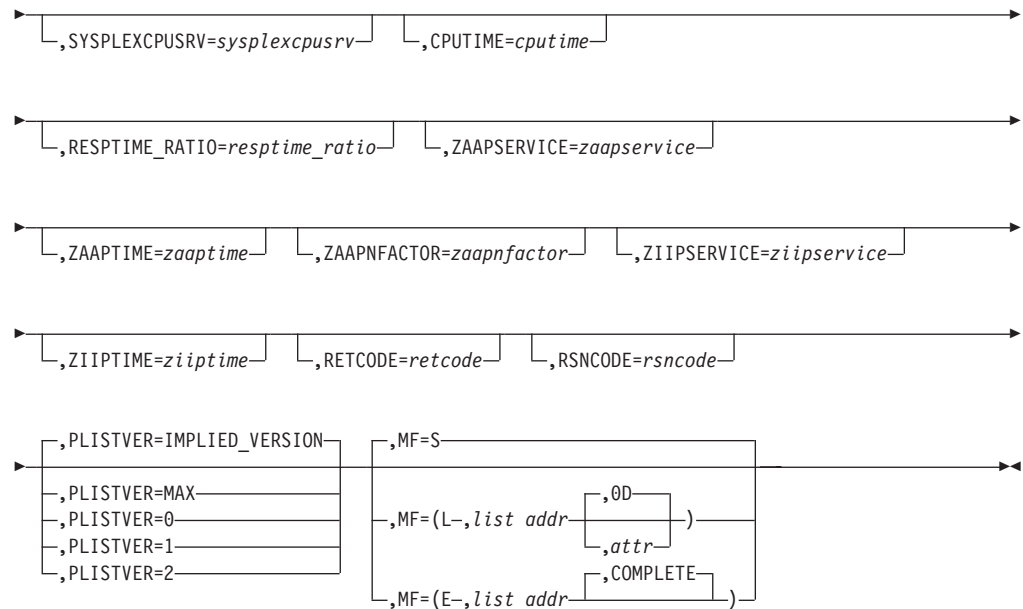
---

## Syntax

main diagram

→ `[name] IWM4EDEL—ETOKEN=etoken [,CPUSERVICE=cpuservice]` →





## Parameters

The parameters are explained as follows:

### *name*

An optional symbol, starting in column 1, that is the name on the IWM4EDEL macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

### **,CPUSERVICE=cpuservice**

An optional output parameter, which will contain the CPU service accumulated by the enclave on the local system.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 64-bit field.

### **,CPUTIME=cputime**

An optional output parameter, which will contain the total CPU time accumulated by the enclave on the local system.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 64-bit field.

### **ETOKEN=etoken**

A required input parameter, which contains the enclave token to be returned.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

### **,MF=S**

**,MF=(L,list addr)**

**,MF=(L,list addr,attr)**

**,MF=(L,list addr,0D)**

**,MF=(E,list addr)**

**,MF=(E,list addr,COMPLETE)**

An optional input parameter that specifies the macro form.

## IWM4EDEL Macro

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

*,list addr*

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

*,attr*

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

**,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

**,PLISTVER=IMPLIED\_VERSION**

**,PLISTVER=MAX**

**,PLISTVER=0**

**,PLISTVER=1**

**,PLISTVER=2**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form, when both are assembled with the same level of the system. In this way, MAX ensures that the parameter list does not overwrite nearby storage.

- **0**, which supports all parameters except those specifically referenced in higher versions.
- **1**, which supports both the following parameters and those from version 0:

RESPTIME\_RATIO

- 2, which supports both the following parameters and those from version 0 and 1:

ZAAPNFACTOR	ZAAPTIME	ZIIPTIME
ZAAPSERVICE	ZIIPSERVICE	

**To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX
- A decimal value of 0, 1, or 2

**,RESPTIME\_RATIO=***resptime\_ratio*

An optional output parameter, which contains the response time ratio times 100:  $\text{act.resp.time} / \text{goal} * 100$  if the enclave has a response time goal (limited to:  $1 \leq \text{RESPTIME\_RATIO} \leq 1000$ ) 0 otherwise

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32-bit field.

**,RETCODE=***retcode*

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,RSNCODE=***rsncode*

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,SYSPLEXCPUSRV=***sysplexcpusrv*

An optional output parameter, which will contain the CPU service accumulated by the enclave on the local system and on other systems through the use of the IWMEXPT and IWMIMPT services. If the IWMEXPT and IWMIMPT services were not used, SYSPLEXCPUSRV returns the same value as CPUSERVICE.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 64-bit field.

**,ZAAPNFACTOR=***zaapnfactor*

An optional output parameter, which contains the normalization factor for application assist processors (zAAPs). If zAAPs are running at a different speed, multiply zAAP service and times with this factor and divide the result by 256 to normalize the values to the speed of regular CPs. Note however, that if there has been a speed change of zAAP processors during the life time of the enclave, this calculation will return imprecise data.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

**,ZAAPSERVICE=***zaapservice*

An optional output parameter, which contains the application assist processor (zAAP) service accumulated by the enclave on the local system. The value is not normalized to the speed of regular CPs, but is expressed in zAAP speed which might be different. You may use ZAAPNFACTOR to normalize the value to the speed of regular CPs. Note however, that if the zAAP speed changed during the life time of the enclave, this value cannot be normalized precisely.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 64-bit field.

## IWM4EDEL Macro

### **,ZAAPTIME=***zaaptime*

An optional output parameter, which contains the total application assist processor (zAAP) time accumulated by the enclave on the local system. The value is not normalized to the speed of regular CPUs, but is expressed in zAAP speed which might be different. You may use ZAAPNFACTOR to normalize the value to the speed of regular CPUs. Note however, that if the zAAP speed changed during the life time of the enclave, this value cannot be normalized precisely.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 64-bit field.

### **,ZIIPSERVICE=***ziipservice*

An optional output parameter, which contains the integrated information processor (zIIP) service accumulated by the enclave on the local system. The service is normalized to standard processor speed.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 64-bit field.

### **,ZIIPTIME=***ziiptime*

An optional output parameter, which contains the total integrated information processor (zIIP) time accumulated by the enclave on the local system. The time is normalized to standard processor speed.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 64-bit field.

---

## ABEND Codes

None.

---

## Return and Reason Codes

When the IWM4EDEL macro returns control to your program:

- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, if you coded RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 73. Return and Reason Codes for the IWM4EDEL Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> <code>IwmRetCodeOk</code> <b>Meaning:</b> Successful completion. <b>Action:</b> None required.
4	—	<b>Equate Symbol:</b> <code>IwmRetCodeWarning</code> <b>Meaning:</b> Successful completion, unusual conditions noted.
4	xxxx0411	<b>Equate Symbol:</b> <code>IwmRsnCodeEnclActive</code> <b>Meaning:</b> Input enclave had 1 or more SRBs scheduled or running, or 1 or more TCBs joined to the enclave. <b>Action:</b> None required.

Table 73. Return and Reason Codes for the IWM4EDEL Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	—	<b>Equate Symbol:</b> IwmRetCodeInvocError <b>Meaning:</b> Invalid invocation environment or parameters.
8	xxxx0803	<b>Equate Symbol:</b> IwmRsnCodeDisabled <b>Meaning:</b> Caller is disabled. <b>Action:</b> Avoid requesting this function while disabled.
8	xxxx0804	<b>Equate Symbol:</b> IwmRsnCodeLocked <b>Meaning:</b> Caller is locked. <b>Action:</b> Avoid requesting this function while locked.
8	xxxx080B	<b>Equate Symbol:</b> IwmRsnCodeBadPI <b>Meaning:</b> Error accessing parameter list. <b>Action:</b> Check for possible storage overlay.
8	xxxx0824	<b>Equate Symbol:</b> IwmRsnCodeAmode24 <b>Meaning:</b> Caller invoked service but was in 24 bit addressing mode. <b>Action:</b> Request this function only when you are in 31 bit addressing mode.
8	xxxx0828	<b>Equate Symbol:</b> IwmRsnCodeBadVersion <b>Meaning:</b> Version number in parameter list is not valid. <b>Action:</b> Check for possible storage overlay of the parameter list.
8	xxxx083A	<b>Equate Symbol:</b> IwmRsnCodeBadEnclave <b>Meaning:</b> Enclave token does not pass verification. <b>Action:</b> Check for possible storage overlay of the enclave token, or asynchronous events which may have deleted the enclave.
8	xxxx0872	<b>Equate Symbol:</b> IwmRsnCodeForeignEnclave <b>Meaning:</b> The enclave is foreign. <b>Action:</b> Use the IWMUIMPT macro to delete a foreign enclave.
10	—	<b>Equate Symbol:</b> IwmRetCodeCompError <b>Meaning:</b> Component error. <b>Action:</b> No action required. The function may be successful if invoked again.

## Example

To allow the current task to join an enclave, specify:

```
IWM4EDEL ETOKEN=ENCTOKEN,RETCODE=RC,RSNCODE=RSN
*
* Storage areas
*
ENCTOKEN DS    CL8           Contains the enclave token
```

## IWM4EDEL Macro

*				
RC	DS	F		Return code
RSN	DS	F		Reason code

---

## Chapter 69. IWM4HLTH – Set Server Health Indicator

The IWM4HLTH service is used to inform WLM about the health state of a server. The health indicator is a number which shows in percent how well the server is performing. It can be an integer value between 0 and 100 and means the following:

<b>Value</b>	<b>Meaning</b>
<b>100</b>	The server is fully capable to do work without any health problems
<b>0</b>	The server is not able to do any work occasionally
<b>Any value between 0 and 100</b>	Indicates the level of health of the server

The health indicator is activated when one of the routing services IWM4SRSC or IWMSRSRS with FUNCTION=SPECIFIC are used to get routing recommendations. The weights are reduced to the percentage given by the health indicator of the server address space.

The health indicator of a server keeps its value until it is modified by resetting it with the IWM4HLTH service, or with the IWMSRSRG service.

---

### Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Problem state with any PSW key.
<b>Dispatchable unit mode:</b>	Task or SRB
<b>Cross memory mode:</b>	Any PASN, any HASN, any SASN
<b>AMODE:</b>	31- or 64-bit
<b>ASC mode:</b>	Primary
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks may be held. FRR environments may be established.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

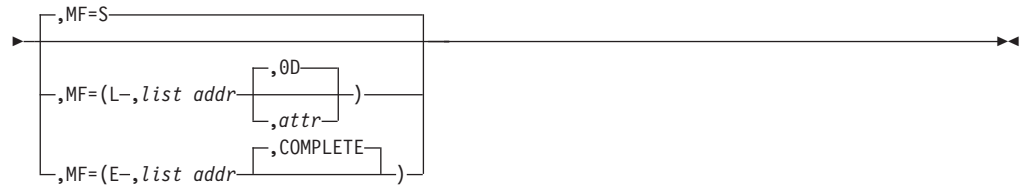
---

### Programming Requirements

1. The macro CVT must be included to use this macro.
2. The macro IWMYCON must be included to use this macro.
3. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
4. Note that the high-order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high-order halfword should thus be excluded from comparison with the reason code values described above. The constant, IWMSRNCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.
5. All character data, unless otherwise specified, is assumed to be left justified and padded with blanks on the right, as needed, to occupy the specified number of bytes.







## Parameters

The parameters are explained as follows:

### *name*

An optional symbol, starting in column 1, that is the name on the IWM4HLTH macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

### **,HEALTH=health**

A required input parameter, which contains the health factor associated with the address space. This value shows the percentage up to which this address space is capable to handle requests.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

### **,MF=S**

**,MF=(L, list addr)**

**,MF=(L, list addr, attr)**

**,MF=(L, list addr, OD)**

**,MF=(E, list addr)**

**,MF=(E, list addr, COMPLETE)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

### *,list addr*

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

### *,attr*

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

## IWM4HLTH Macro

### ,COMPLETE

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

### ,PLISTVER=IMPLIED\_VERSION

### ,PLISTVER=MAX

### ,PLISTVER=0

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form, when both are assembled with the same level of the system. In this way, MAX ensures that the parameter list does not overwrite nearby storage.

- **0**, if you use the currently available parameters.

**To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX
- A decimal value of 0

### ,RETCODE=*retcode*

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

### ,RSNCODE=*rsncode*

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

### STOKEN=*stoken*

A required input parameter, which contains the space token of the server.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 64 bit field.

---

## ABEND Codes

None.

---

## Return and Reason Codes

When the IWM4HLTH macro returns control to your program:

- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, if you coded RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 74. Return and Reason Codes for the IWM4HLTH Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> IwmRetCodeOk <b>Meaning:</b> Successful completion. <b>Action:</b> None required.
4	—	<b>Equate Symbol:</b> IwmRetCodeWarning <b>Meaning:</b> Successful completion, unusual conditions noted.
8	—	<b>Equate Symbol:</b> IwmRetCodeInvocError <b>Meaning:</b> Invalid invocation environment or parameters.
8	xxxx0803	<b>Equate Symbol:</b> IwmRsnCodeDisabled <b>Meaning:</b> Caller is disabled. <b>Action:</b> Avoid requesting this function while disabled.
8	xxxx0804	<b>Equate Symbol:</b> IwmRsnCodeLocked <b>Meaning:</b> Caller is locked. <b>Action:</b> Avoid requesting this function while locked.
8	xxxx0807	<b>Equate Symbol:</b> IwmRsnCodeBadSTOKEN <b>Meaning:</b> Bad STOKEN passed. <b>Action:</b> Check for possible storage overlay.
8	xxxx080B	<b>Equate Symbol:</b> IwmRsnCodeBadPI <b>Meaning:</b> Error accessing parameter list. <b>Action:</b> Check for possible storage overlay.
8	xxxx0823	<b>Equate Symbol:</b> IwmRsnCodeDatoff <b>Meaning:</b> Caller invoked service while DATOFF <b>Action:</b> Avoid requesting this function in this environment.
8	xxxx0824	<b>Equate Symbol:</b> IwmRsnCodeAmode24 <b>Meaning:</b> Caller invoked service but was in 24-bit addressing mode. <b>Action:</b> Request this function only when you are in 31 bit addressing mode.
8	xxxx0825	<b>Equate Symbol:</b> IwmRsnCodeAscModeNotPrimary <b>Meaning:</b> Caller invoked service but was not DAT on Primary ASC mode. <b>Action:</b> Avoid requesting this function in this environment.
8	xxxx0828	<b>Equate Symbol:</b> IwmRsnCodeBadVersion <b>Meaning:</b> Version number in parameter list or version length field is not valid. Or This service was called on a z/OS release not supported. <b>Action:</b> Check for possible storage overlay of the parameter list.

## IWM4HLTH Macro

Table 74. Return and Reason Codes for the IWM4HLTH Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx08A2	<p><b>Equate Symbol:</b> IwmRsnCodeBadHealth</p> <p><b>Meaning:</b> Health Value out of range</p> <p><b>Action:</b> Check for possible storage overlay.</p>
C	—	<p><b>Equate Symbol:</b> IwmRetCodeEnvError</p> <p><b>Meaning:</b> Environmental error.</p>
C	xxxx0C01	<p><b>Equate Symbol:</b> IwmRsnCodeNoStg</p> <p><b>Meaning:</b> Storage is not available for the request.</p> <p><b>Action:</b> There is a storage shortage. The function may work successfully at a later time.</p>
10	—	<p><b>Equate Symbol:</b> IwmRetCodeCompError</p> <p><b>Meaning:</b> Component error.</p> <p><b>Action:</b> No action required. The function may be successful if invoked again.</p>

## Example

To set the health indication value for a particular server, specify:

```

IWM4HLTH STOKEN=STKN
          HEALTH=HLTH
          RETCODE=RC,RSNCODE=RSN
*
* Storage areas
*
STKN    DS    CL8           Contains the STOKEN
*                               associated with the address
*                               space
HLTH    DS    F             Field to input the health value
RC      DS    F             Return code
RSN     DS    F             Reason code

```

---

## Chapter 70. IWM4MCHS – Monitoring Change State Of Work Unit

The purpose of this service is to reflect in a monitoring environment what the current state of a work request is with respect to delays.

### Information

Note that this service was previously called IWMMCHST for 31-bit address space only (see “IWMMCHST – Monitoring Change State Of Work Unit” on page 816).

---

## Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Problem state. PSW key must either be 0 or match the value supplied on IWM4MCRE.
<b>Dispatchable unit mode:</b>	Task or SRB
<b>Cross memory mode:</b>	Any PASN, any HASN, any SASN
<b>AMODE:</b>	31- or 64-bit
<b>ASC mode:</b>	Primary
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	Suspend locks are allowed.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

---

## Programming Requirements

1. The macro IWMYCON must be included to use this macro.
2. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
3. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, IWMRSNCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.

---

## Restrictions

1. Caller is responsible for error recovery
2. Only limited checking is done against the input monitoring token.
3. If the key specified on IWM4MCRE was a user key (8-F), then the primary addressability must exist to the performance block IWM4MCRE obtained. This condition is satisfied by ensuring that current primary matches primary at the time that IWM4MCRE was invoked. If this service is invoked in a subspace, the condition may be satisfied by ensuring that the performance block is shared with the base space.
4. The caller must serialize to prevent any delay monitoring services from being invoked concurrently for the environment represented by the monitoring token.

## IWM4MCHS Macro

5. This macro may only be used on z/OS R2 or higher levels for the following state/resources:
  - STATE(ACTIVE\_APPL)
  - RESOURCE(SSL\_THREAD)
  - RESOURCE(REG\_THREAD)
  - RESOURCE(REG\_TO\_WRKTB)
  - RESOURCE(TYPE1)
  - RESOURCE(TYPE2)
  - RESOURCE(TYPE3)
  - RESOURCE(TYPE4)
  - RESOURCE(TYPE5)
6. This macro may only be used on z/OS R8 or higher versions for the following resources:
  - RESOURCE(BUFFER\_POOL\_IO)
7. This macro may only be used on z/OS R8 or higher versions for RESTKN keyword.

---

## Input Register Information

Before issuing the IWM4MCHS macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

---

## Output Register Information

When control returns to the caller, the GPRs contain:

<b>Register</b>	<b>Contents</b>
<b>0</b>	Reason code if GR15 return code is non-zero
<b>1</b>	Used as work registers by the system
<b>2-13</b>	Unchanged
<b>14</b>	Used as work registers by the system
<b>15</b>	Return code

When control returns to the caller, the ARs contain:

<b>Register</b>	<b>Contents</b>
<b>0-1</b>	Used as work registers by the system
<b>2-13</b>	Unchanged
<b>14-15</b>	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

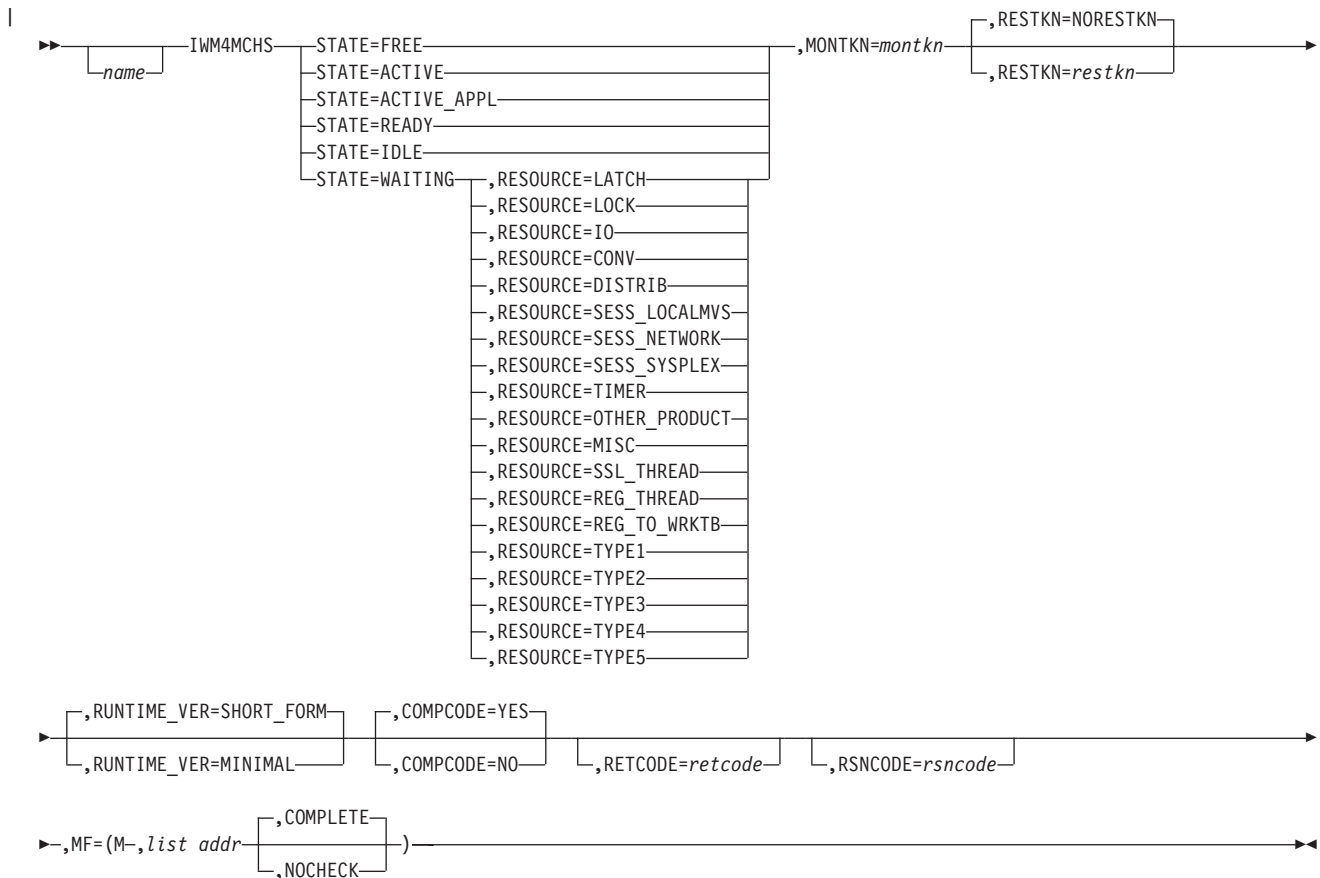
---

## Performance Implications

None.

## Syntax

### main diagram



## Parameters

The parameters are explained as follows:

### *name*

An optional symbol, starting in column 1, that is the name on the IWM4MCHS macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

### **,COMPCODE=YES**

### **,COMPCODE=NO**

An optional parameter, which indicates whether completion status for this service is needed. The default is COMPCODE=YES.

### **,COMPCODE=YES**

indicates that completion status is needed.

### **,COMPCODE=NO**

indicates that completion status is not needed. Registers 0, 15 cannot be used as reason code and return code registers upon completion of the macro expansion. Therefore, neither RETCODE NOR RSNCODE may be specified when COMPCODE(NO) is specified.

**,MONTKN=montkn**

A required input parameter, which contains the delay monitoring token

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32-bit field.

**,RESOURCE=LATCH**

**,RESOURCE=LOCK**

**,RESOURCE=IO**

**,RESOURCE=CONV**

**,RESOURCE=DISTRIB**

**,RESOURCE=SESS\_LOCALMVS**

**,RESOURCE=SESS\_NETWORK**

**,RESOURCE=SESS\_SYSPLEX**

**,RESOURCE=TIMER**

**,RESOURCE=OTHER\_PRODUCT**

**,RESOURCE=MISC**

**,RESOURCE=SSL\_THREAD**

**,RESOURCE=REG\_THREAD**

**,RESOURCE=REG\_TO\_WRKTB**

**,RESOURCE=TYPE1**

**,RESOURCE=TYPE2**

**,RESOURCE=TYPE3**

**,RESOURCE=TYPE4**

**,RESOURCE=TYPE5**

When STATE=WAITING is specified, a required parameter, which indicates the resource that the work manager is waiting for on behalf of the work request described by the monitoring environment.

**,RESOURCE=LATCH**

indicates that the work manager is waiting for a latch.

**,RESOURCE=LOCK**

indicates that the work manager is waiting for a lock.

**,RESOURCE=IO**

indicates that the work manager is waiting for an activity related to an I/O request. This may either be an actual I/O operation or some function associated with an IO request that cannot be more precisely determined by the work manager, for example, locks, or buffers.

**,RESOURCE=CONV**

indicates that the work manager is waiting for a conversation. This may be used in conjunction with IWMMSWCH to identify where the target is located.

**,RESOURCE=DISTRIB**

indicates that the work manager is waiting for a distributed request. This expresses at a high level that some function or data must be routed before resuming the work request. This is to be contrasted with Waiting on Conversation, which is a low level view of the precise resource that is needed. A distributed request could involve waiting on a conversation as part of its processing.

**,RESOURCE=SESS\_LOCALMVS**

indicates that the work manager is waiting to establish a session somewhere in the current MVS image.



**,RESOURCE=SESS\_NETWORK**

indicates that the work manager is waiting to establish a session somewhere in the network.

**,RESOURCE=SESS\_SYSPLEX**

indicates that the work manager is waiting to establish a session somewhere in the sysplex.

**,RESOURCE=TIMER**

indicates that the work request is waiting for a timer.

**,RESOURCE=OTHER\_PRODUCT**

indicates that the work manager is waiting for another product to complete its function.

**,RESOURCE=MISC**

indicates that the work manager is waiting for some unidentified resource, possibly among the previous categories.

**,RESOURCE=SSL\_THREAD**

indicates that the work manager is waiting for an SSL thread.

**,RESOURCE=REG\_THREAD**

indicates that the work manager is waiting for a regular processing thread.

**,RESOURCE=REG\_TO\_WRKTB**

indicates that the work manager is waiting for the registration to a worktable.

**,RESOURCE=TYPE1**

indicates that the work manager is waiting for resource type 1.

**,RESOURCE=TYPE2**

indicates that the work manager is waiting for resource type 2.

**,RESOURCE=TYPE3**

indicates that the work manager is waiting for resource type 3.

**,RESOURCE=TYPE4**

indicates that the work manager is waiting for resource type 4.

**,RESOURCE=TYPE5**

indicates that the work manager is waiting for resource type 5.

**,RESTKN=*restkn*****,RESTKN=NORESTKN**

An optional input parameter, which contains the token of the managed resource previously registered with register resource (IWMMREG). The default is NORESTKN, which indicates that no resource token is provided.

NORESTKN preserves the existing resource token, if any.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

**,RETCODE=*retcode***

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,RSNCODE=*rsncode***

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,RUNTIME\_VER=SHORT\_FORM**

**,RUNTIME\_VER=MINIMAL**

An optional parameter, which indicates what level of runtime verification will be performed. The default is `RUNTIME_VER=SHORT_FORM`.

**,RUNTIME\_VER=SHORT\_FORM**

indicates that checking should verify that a monitoring environment is established and passes a short form of verification before being used.

**,RUNTIME\_VER=MINIMAL**

indicates that checking will only be done to verify that a monitoring environment may be established, assuming that it would be valid and useable if established.

**STATE=FREE**

**STATE=ACTIVE**

**STATE=ACTIVE\_APPL**

**STATE=READY**

**STATE=IDLE**

**STATE=WAITING**

A required parameter, which indicates the current state for the work request.

**STATE=FREE**

indicates that the work manager has no work request associated with the monitoring environment.

**STATE=ACTIVE**

indicates that there is a program executing on behalf of the work request described by the monitoring environment. This is an indication from the perspective of the work manager using this service, who should not try to factor in MVS decisions in preempting work, etc.

**STATE=ACTIVE\_APPL**

indicates that there is an application program executing on behalf of the work request described by the monitoring environment. This is an indication from the perspective of the work manager using this service, who should not try to factor in MVS decisions in preempting work. This state represents the application activity in contrast to the active (subsystem) state.

**STATE=READY**

indicates that there is a program ready to execute on behalf of the work request described by the monitoring environment, but the work manager has given priority to another work request.

**STATE=IDLE**

indicates that the work manager has no work requests that it is allowed to service within the monitoring environment. This represents a delay that is not under the control of the work manager itself and which it cannot eliminate. This may be caused by limits imposed by the installation or by the kind of work request itself.

**STATE=WAITING**

indicates that the work manager is waiting for a resource on behalf of the work request described by the monitoring environment.

---

## ABEND Codes

None.

## Return and Reason Codes

When the IWM4MCHS macro returns control to your program:

- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, if you coded RSNCODE) contains a reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 75. Return and Reason Codes for the IWM4MCHS Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<p><b>Equate Symbol:</b> <code>IwmRetCodeOk</code></p> <p><b>Meaning:</b> Successful completion.</p> <p><b>Action:</b> None required.</p>
4	—	<p><b>Equate Symbol:</b> <code>IwmRetCodeWarning</code></p> <p><b>Meaning:</b> Successful completion, unusual conditions noted.</p>
4	xxxx0402	<p><b>Equate Symbol:</b> <code>IwmRsnCodeNoMonEnv</code></p> <p><b>Meaning:</b> Input monitoring token indicates no monitoring environment was established.</p> <p><b>Action:</b> None required.</p>
8	—	<p><b>Equate Symbol:</b> <code>IwmRetCodeInvocError</code></p> <p><b>Meaning:</b> Invalid invocation environment or parameters.</p>
8	xxxx0820	<p><b>Equate Symbol:</b> <code>IwmRsnCodeBadMonEnv</code></p> <p><b>Meaning:</b> Monitoring environment does not pass short form verification.</p> <p><b>Action:</b> Check for possible storage overlay.</p>



---

## Chapter 71. IWM4MCRE – Create Delay Monitoring Environment

The purpose of this service is to create a single delay monitoring environment or a number of delay monitoring environments so that work and resource managers may utilize other delay monitoring services to reflect to MVS the execution states and delays associated with work requests.

There are three types of monitoring environments available, management monitoring environments, report-only monitoring environments and buffer pool management only environments. Management monitoring environments provide both, performance management and performance reporting. Report-only monitoring environments can be used for performance reporting only. Buffer pool management only monitoring environments provide only buffer pool performance management for enclaves.

Optionally, if you specify REPORTONLY=YES, the monitoring environment is used for reporting purposes only.

Furthermore, if you specify BPMGMTONLY=YES, the monitoring environment is used for buffer pool management for enclaves only.

### Information

Note that this service was previously called IWMMCREA for 31-bit address space only (see “IWMMCREA – Create Delay Monitoring Environment” on page 822).

---

## Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Supervisor state or program key mask (PKM) allowing keys 0-7.
<b>Dispatchable unit mode:</b>	Task
<b>Cross memory mode:</b>	Non-XMEM or XMEM. Any P.S.H.
<b>AMODE:</b>	31- or 64-bit
<b>ASC mode:</b>	Primary
<b>Interrupt status:</b>	Enabled for I/O and external interrupts No (EUT) FRR established.
<b>Locks:</b>	No locks may be held.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

---

## Programming Requirements

1. The macro CVT must be included to use this macro.
2. The macro IWMYCON must be included to use this macro.
3. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
4. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT

## IWM4MCRE Macro

part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, IWMRSNCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.

5. All character data, unless otherwise specified, is assumed to be left justified and padded with blanks on the right, as needed, to occupy the specified number of bytes.

---

## Restrictions

1. This macro may not be used during task/address space termination.
2. If the key specified on IWM4MCRE is a user key (8-F), then the caller must be in non-cross-memory mode (P=S=H)
3. While not a restriction for IWM4MCRE, it should be noted that when the key specified is a user key (8-F), the delay monitoring token may only be passed (to any service whatsoever), when primary addressability exists to the performance block obtained by IWM4MCRE. This condition may be satisfied by ensuring that the then current primary matches primary at the time that IWM4MCRE was invoked. If these other services are invoked in a subspace, the condition may be satisfied by ensuring that the performance block is shared with the base space.
4. This service provides a task and address space resource managers to clean up any resources obtained during task and address space terminations. Once the calling task or address space terminates, the monitoring token returned by IWM4MCRE must not be used for any services.
5. This macro supports multiple versions. Some keywords are only supported by certain versions. Refer to the PLISTVER parameter description for further information.

---

## Input Register Information

Before issuing the IWM4MCRE macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

---

## Output Register Information

When control returns to the caller, the GPRs contain:

<b>Register</b>	<b>Contents</b>
<b>0</b>	Reason code if GR15 return code is non-zero
<b>1</b>	Used as work registers by the system
<b>2-13</b>	Unchanged
<b>14</b>	Used as work registers by the system
<b>15</b>	Return code

When control returns to the caller, the ARs contain:

<b>Register</b>	<b>Contents</b>
<b>0-1</b>	Used as work registers by the system
<b>2-13</b>	Unchanged
<b>14-15</b>	Used as work registers by the system

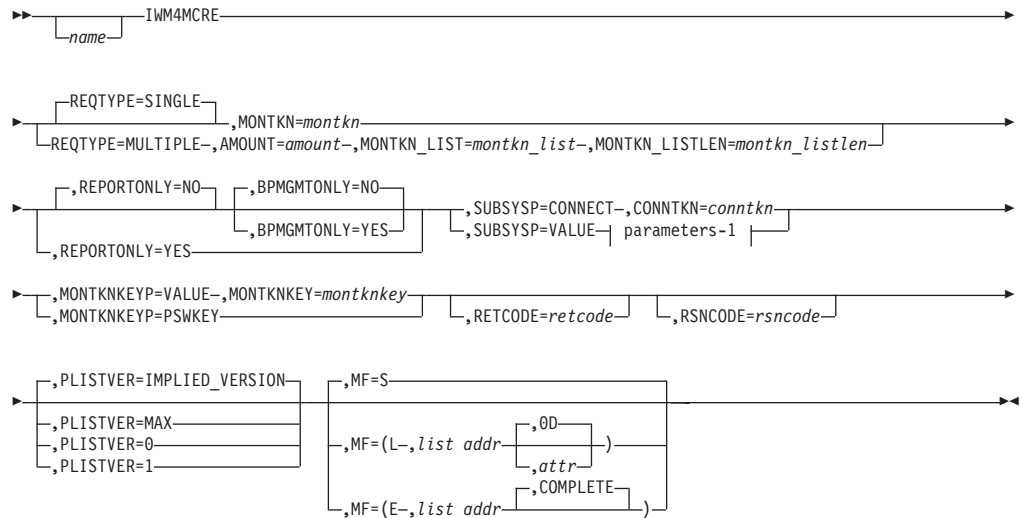
Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

## Performance Implications

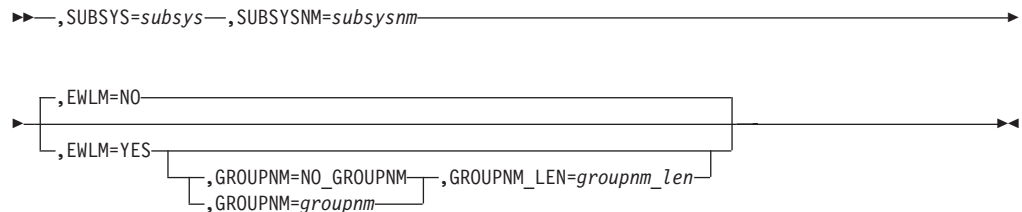
None.

## Syntax

### main diagram



### parameters-1



## Parameters

The parameters are explained as follows:

### *name*

An optional symbol, starting in column 1, that is the name on the IWM4MCRE macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

### **,AMOUNT=amount**

When REQTYPE=MULTIPLE is specified, a required input parameter, which specifies the number of delay monitoring environments to be created.

While there is no restriction on the number of delay monitoring environments to be created, the caller should only create the minimum number of delay monitoring environments that are needed.

If there are too many unused delay monitoring environments existing in the system, the storage and CPU overheads may be significant.

## IWM4MCRE Macro

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

|  
|  
| **,BPMGMTONLY=NO**  
| **,BPMGMTONLY=**YES

| When REPORTONLY=NO is specified, an optional parameter, which indicates  
| whether the monitoring environment is for buffer pool management purposes  
| only (YES) or (NO). The default is BPMGMTONLY=NO.

| **,BPMGMTONLY=NO**

| indicates that the monitoring environment is not for buffer pool management  
| purposes.

| **,BPMGMTONLY=YES**

| indicates that the monitoring environment is for buffer pool management  
| purposes only.

| **,CONNTKN=***conntkn*

| When SUBSYSP=CONNECT is specified, a required input parameter, which  
| contains the connect token associated with the subsystem.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32-bit field.

| **,EWLM=NO**  
| **,EWLM=**YES

| IF SUBSYSP=CONNECT is specified, this parameter is not allowed. In this  
| case the EWLM=YES/NO specification is inherited from the IWM4CON (or  
| IWMCONN) EWLM=YES/NO specification. When SUBSYSP=VALUE is  
| specified, an optional parameter, which indicates if the created monitoring  
| environment is intended to participate in Enterprise Workload Management  
| (EWLM). The default is EWLM=NO.

| **,EWLM=NO**

| The monitoring environment cannot be used to report on ARM work  
| requests.

| **,EWLM=YES**

| The monitoring environment participates in cross-platform Enterprise  
| Workload Management and interacts with EWLM. An ARM application  
| instance will be registered and started using the passed subsystem type  
| (SUBSYS), subsystem name (SUBSYSNM), and the new parameter group  
| name (GROUPNM, GROUPNM\_LEN) - an already existing ARM  
| registration for the same address space with identical SUBSYS,  
| SUBSYSNM, GROUPNM and GROUPNM\_LEN parameters will be reused.  
| All ARM work requests associated with the created monitoring environment  
| are reported for this ARM application instance.

| **,GROUPNM=***groupnm*  
| **,GROUPNM=**NO\_GROUPNM

| When EWLM=YES and SUBSYSP=VALUE are specified, an optional input  
| parameter, which contains the name of an application group, for example, a  
| group of similar or cooperating subsystem instances. A group name can be up  
| to 255 characters long. Provision of a data area initialized to all blanks is  
| equivalent to specification of NO\_GROUPNM. NO\_GROUPNM indicates that no  
| group name is passed. This is the default.

| **To code:** Specify the RS-type address, or address in register (2)-(12), of a  
| character field.



**,GROUPNM\_LEN=groupnm\_len**

When GROUPNM=*groupnm*, EWLM=YES and SUBSYSP=VALUE are specified, a required input parameter, which contains the length of the group name. A group name can be up to 255 characters long.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

**,MF=S****,MF=(L,list addr)****,MF=(L,list addr,attr)****,MF=(L,list addr,0D)****,MF=(E,list addr)****,MF=(E,list addr,COMPLETE)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

**,list addr**

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

**,attr**

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

**,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

**,MONTKN=montkn**

When REQTYPE=SINGLE is specified, a required output parameter, which will receive the delay monitoring token.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32-bit field.

**,MONTKN\_LIST=montkn\_list**

When REQTYPE=MULTIPLE is specified, a required input parameter, which specifies an area into which a list of delay monitoring tokens will be placed.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a character field.

**,MONTKN\_LISTLEN=montkn\_listlen**

When REQTYPE=MULTIPLE is specified, a required input parameter, which specifies the length (in bytes) of the area identified by the **MONTKN\_LIST** keyword.

Size of this area must be at least the size of **MONTKN** (see **MONTKN** keyword) times **AMOUNT**. If the user specified area is not large enough to return the delay monitoring tokens, a specific return/reason code will be returned and the request will not be processed.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

**,MONTKNKEY=montknkey**

When MONTKNKEYP=VALUE is specified, a required input parameter, which contains the key in which the delay monitoring services will be invoked subsequently when using the output MONTKN. The low order 4 bits (bits 4-7) contain the key value. The high order 4 bits (bits 0-3) must be 0.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-bit field.

**,MONTKNKEYP=VALUE****,MONTKNKEYP=PSWKEY**

A required parameter, which describes how the input key should be obtained.

**,MONTKNKEYP=VALUE**

indicates that the key is being passed explicitly via MONTKNKEY.

**,MONTKNKEYP=PSWKEY**

indicates that the current PSW key should be used.

**,PLISTVER=IMPLIED\_VERSION****,PLISTVER=MAX****,PLISTVER=0****,PLISTVER=1**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form, when both are assembled with the same level of the system. In this way, MAX ensures that the parameter list does not overwrite nearby storage.

- **0**, which supports all parameters except those specifically referenced in higher versions.

- **1**, which supports both the following parameters and those from version 0:

BPMGMTONLY

GROUPNM

EWLM

GROUPNM\_LEN

**To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX
- A decimal value of 0, or 1

**,REPORTONLY=NO**

**,REPORTONLY=YES**

An optional parameter, which indicates whether the monitoring environment is for reporting purposes only (YES) or (NO). The default is REPORTONLY=NO.

**,REPORTONLY=NO**

indicates that the monitoring environment is for management and reporting purposes.

**,REPORTONLY=YES**

indicates that the monitoring environment is for reporting purposes only.

**REQTYPE=SINGLE**

**REQTYPE=MULTIPLE**

An optional parameter that indicates whether the request is to create a single delay monitoring environment or to create multiple delay monitoring environments. The default is REQTYPE=SINGLE.

**REQTYPE=SINGLE**

requests to create a single delay monitoring environment.

**REQTYPE=MULTIPLE**

requests to create a number of delay monitoring environments.

**,RETCODE=*retcode***

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,RSNCODE=*rsncode***

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,SUBSYS=*subsys***

When SUBSYSP=VALUE is specified, a required input parameter, which contains the generic subsystem type, for example, IMS, or CICS.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 4-character field.

**,SUBSYSNM=*subsysnm***

When SUBSYSP=VALUE is specified, a required input parameter, which contains the subsystem name.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

**,SUBSYSP=CONNECT**

**,SUBSYSP=VALUE**

A required parameter, which describes how the calling subsystem is providing identification.

**,SUBSYSP=CONNECT**

indicates that the connect token is being passed.

**,SUBSYSP=VALUE**

indicates that the subsystem name is being passed directly.

**ABEND Codes**

None.

**Return and Reason Codes**

When the IWM4MCRE macro returns control to your program:

- GPR 15 (and *retcode*, if you code RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, if you coded RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 76. Return and Reason Codes for the IWM4MCRE Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> <code>IwmRetCodeOk</code> <b>Meaning:</b> Successful completion. <b>Action:</b> None required.
4	—	<b>Equate Symbol:</b> <code>IwmRetCodeWarning</code> <b>Meaning:</b> Successful completion, unusual conditions noted.
4	xxxx0401	<b>Equate Symbol:</b> <code>IwmRsnCodeNoWLM</code> <b>Meaning:</b> System does not support delay monitoring services. The delay monitoring token returned is useable in other services. However use of this token will NOT result in the action requested of those services. This return code is only set when the MVS release is prior to MVS/ESA SP5.1.0. <b>Action:</b> None required.
4	xxxx0409	<b>Equate Symbol:</b> <code>IwmRsnCodeNoConn</code> <b>Meaning:</b> Connect token does not reflect a successful Connect. The delay monitoring token returned is useable in other services. However use of this token will NOT result in the action requested of those services. <b>Action:</b> None required.
8	—	<b>Equate Symbol:</b> <code>IwmRetCodeInvocError</code> <b>Meaning:</b> Invalid invocation environment or parameters.
8	xxxx0801	<b>Equate Symbol:</b> <code>IwmRsnCodeSrbMode</code> <b>Meaning:</b> Caller is in SRB mode. <b>Action:</b> Avoid requesting this function while in SRB mode.
8	xxxx0802	<b>Equate Symbol:</b> <code>IwmRsnCodeXmemUserKeyTkn</code> <b>Meaning:</b> Caller is in cross-memory mode while the token was requested in user key. <b>Action:</b> Avoid requesting this function while in cross-memory mode.

Table 76. Return and Reason Codes for the IWM4MCRE Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0803	<b>Equate Symbol:</b> IwmRsnCodeDisabled <b>Meaning:</b> Caller is disabled. <b>Action:</b> Avoid requesting this function while disabled.
8	xxxx0804	<b>Equate Symbol:</b> IwmRsnCodeLocked <b>Meaning:</b> Caller is locked. <b>Action:</b> Avoid requesting this function while locked.
8	xxxx080B	<b>Equate Symbol:</b> IwmRsnCodeBadPI <b>Meaning:</b> Error accessing parameter list. <b>Action:</b> Check for possible storage overlay.
8	xxxx0810	<b>Equate Symbol:</b> IwmRsnCodeEutFrr <b>Meaning:</b> Caller has EUT FRR established. <b>Action:</b> Avoid requesting this function with an EUT FRR set.
8	xxxx0821	<b>Equate Symbol:</b> IwmRsnCodeBadConn <b>Meaning:</b> Input connect token does not pass validity checking. <b>Action:</b> Check for possible storage overlay.
8	xxxx0823	<b>Equate Symbol:</b> IwmRsnCodeDatoff <b>Meaning:</b> Caller invoked service while DATOFF <b>Action:</b> Avoid requesting this function in this environment.
8	xxxx0824	<b>Equate Symbol:</b> IwmRsnCodeAmode24 <b>Meaning:</b> Caller invoked service but was in 24 bit addressing mode. <b>Action:</b> Request this function only when you are in 31 bit addressing mode.
8	xxxx0825	<b>Equate Symbol:</b> IwmRsnCodeAscModeNotPrimary <b>Meaning:</b> Caller invoked service but was not DAT on Primary ASC mode. <b>Action:</b> Avoid requesting this function in this environment.
8	xxxx0826	<b>Equate Symbol:</b> IwmRsnCodeTaskTerm <b>Meaning:</b> Caller invoked service while task termination is in progress for the TCB associated with the owner. <b>Action:</b> Avoid requesting this function in this environment.
8	xxxx0827	<b>Equate Symbol:</b> IwmRsnCodeRsvdNot0 <b>Meaning:</b> Reserved field in parameter list was non-zero. <b>Action:</b> Check for possible storage overlay of the parameter list.
8	xxxx0828	<b>Equate Symbol:</b> IwmRsnCodeBadVersion <b>Meaning:</b> Version number in parameter list is not valid. <b>Action:</b> Check for possible storage overlay of the parameter list.

## IWM4MCRE Macro

Table 76. Return and Reason Codes for the IWM4MCRE Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0829	<p><b>Equate Symbol:</b> IwmRsnCodeBadOptions</p> <p><b>Meaning:</b> Parameter list omits required parameters or supplies mutually exclusive parameters or provides data associated with options not selected.</p> <p><b>Action:</b> Check for possible storage overlay of the parameter list.</p>
8	xxxx0844	<p><b>Equate Symbol:</b> IwmRsnCodeBadMonTknListLen</p> <p><b>Meaning:</b> The storage area length specified on the MONTKN_LISTLEN parameter is not large enough to contain the data being returned. No data is returned.</p> <p><b>Action:</b> Invoke the function with an output area sufficient to receive the data.</p>
C	—	<p><b>Equate Symbol:</b> IwmRetCodeEnvError</p> <p><b>Meaning:</b> Environmental error.</p>
C	xxxx0C01	<p><b>Equate Symbol:</b> IwmRsnCodeNoStg</p> <p><b>Meaning:</b> Storage is not available for the request.</p> <p><b>Action:</b> There is a storage shortage. The function may work successfully at a later time.</p>
C	xxxx0C09	<p><b>Equate Symbol:</b> IwmRsnCodeNoResmgr</p> <p><b>Meaning:</b> Resource manager could not be established.</p> <p><b>Action:</b> No action required. This condition may be due to a storage shortage condition.</p>
10	—	<p><b>Equate Symbol:</b> IwmRetCodeCompError</p> <p><b>Meaning:</b> Component error.</p> <p><b>Action:</b> No action required. The function may be successful if invoked again.</p>

---

## Chapter 72. IWM4MDRG – Deregister a Resource From Monitoring

The purpose of the IWM4MDRG service is to deregister a resource from monitoring which was previously registered via IWM4MREG. The service allows the caller to identify a resource which is no longer involved in delays to work requests. Thus, the system may no longer alter the size of the resource to balance associated delays.

The system implicitly deregisters a resource due to repetitive errors in calling exits associated with the resource. In this case, the invocation to deregister finds that the associated resource token is invalid and returns with a warning return code. The same return code will be returned in all cases where the resource token is invalid.

---

### Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Supervisor state or program key mask (PKM) allowing keys 0-7.
<b>Dispatchable unit mode:</b>	Task
<b>Cross memory mode:</b>	PASN=HASN=SASN Current HASN must match HASN at the time that IWM4MREG was used to register the resource.
<b>AMODE:</b>	31- or 64-bit
<b>ASC mode:</b>	Primary
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks may be held.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

---

### Programming Requirements

1. The macro IWMYCON must be included to use this macro.
2. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
3. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, IWMRSNCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.
4. All character data input is assumed to be left justified and padded with blanks on the right, as needed, to fill in the specified number of bytes.

---

### Restrictions

- NO FRRs may be established.
- This service should not be invoked from an address space resource manager, because those are dispatched from Master's address space which will not match the space which invoked the registration service. The system will take care of the resources associated with registration when the owning address space terminates.

## Input Register Information

Before issuing the IWM4MDRG macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

## Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Reason code if GR15 return code is non-zero
1	Used as work register by the system
2-13	Unchanged
14	Used as work registers by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

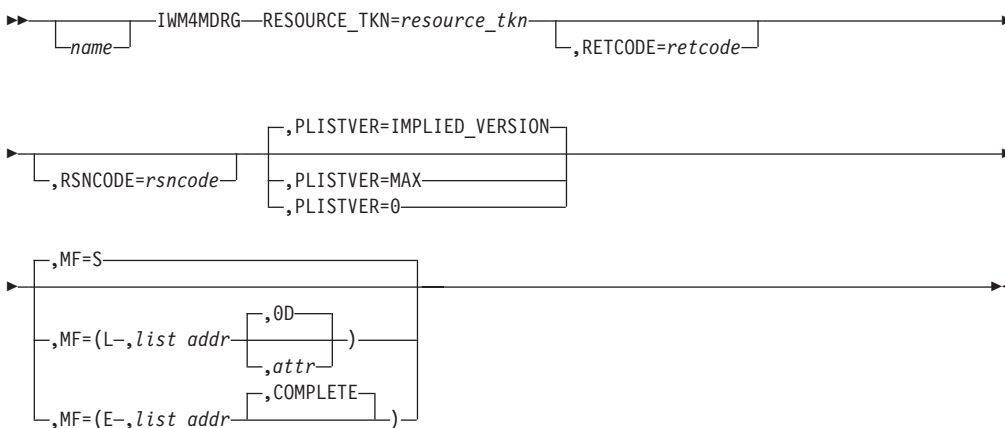
Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

## Performance Implications

None.

## Syntax

### main diagram



## Parameters

The parameters are explained as follows:



*name*

An optional symbol, starting in column 1, that is the name on the IWM4MDRG macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

**,MF=S**

**,MF=(L,*list addr*)**

**,MF=(L,*list addr,attr*)**

**,MF=(L,*list addr,0D*)**

**,MF=(E,*list addr*)**

**,MF=(E,*list addr,COMPLETE*)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

*,list addr*

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

*,attr*

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

**,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

**,PLISTVER=IMPLIED\_VERSION**

**,PLISTVER=MAX**

**,PLISTVER=0**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that

## IWM4MDRG Macro

the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form, when both are assembled with the same level of the system. In this way, MAX ensures that the parameter list does not overwrite nearby storage.

- 0, if you use the currently available parameters.

**To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX
- A decimal value of 0

**RESOURCE\_TKN=***resource\_tkn*

A required input parameter, which contains the associated WLM resource token which is returned by the resource monitoring registration service (IWM4MREG).

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

**,RETCODE=***retcode*

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,RSNCODE=***rsncode*

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

---

## ABEND Codes

None.

---

## Return and Reason Codes

When the IWM4MDRG macro returns control to your program:

- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, if you coded RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 77. Return and Reason Codes for the IWM4MDRG Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> IwmRetCodeOk <b>Meaning:</b> Successful completion. <b>Action:</b> None required.
4	—	<b>Equate Symbol:</b> IwmRetCodeWarning <b>Meaning:</b> Successful completion, unusual conditions noted.

Table 77. Return and Reason Codes for the IWM4MDRG Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
4	xxxx041D	<p><b>Equate Symbol:</b> lwmRsnCodeBadResTkn</p> <p><b>Meaning:</b> The input resource token is not valid. The system may have deregistered due to errors associated with an exit.</p> <p><b>Action:</b> Verify that the resource token passed has the intended value.</p>
8	—	<p><b>Equate Symbol:</b> lwmRetCodeInvocError</p> <p><b>Meaning:</b> Invalid invocation environment or parameters.</p>
8	xxxx0801	<p><b>Equate Symbol:</b> lwmRsnCodeSRBMode</p> <p><b>Meaning:</b> Caller is in SRB mode.</p> <p><b>Action:</b> Avoid requesting this function in SRB mode.</p>
8	xxxx0803	<p><b>Equate Symbol:</b> lwmRsnCodeDisabled</p> <p><b>Meaning:</b> Caller is disabled.</p> <p><b>Action:</b> Avoid requesting this function while disabled.</p>
8	xxxx0804	<p><b>Equate Symbol:</b> lwmRsnCodeLocked</p> <p><b>Meaning:</b> Caller is locked.</p> <p><b>Action:</b> Avoid requesting this function while locked.</p>
8	xxxx080B	<p><b>Equate Symbol:</b> lwmRsnCodeBadPI</p> <p><b>Meaning:</b> Error accessing parameter list.</p> <p><b>Action:</b> Check for possible storage overlay.</p>
8	xxxx0810	<p><b>Equate Symbol:</b> lwmRsnCodeEutFrr</p> <p><b>Meaning:</b> Caller has EUT FRR established.</p> <p><b>Action:</b> Avoid requesting this function with an EUT FRR.</p>
8	xxxx0823	<p><b>Equate Symbol:</b> lwmRsnCodeDatoff</p> <p><b>Meaning:</b> Caller invoked service while DATOFF</p> <p><b>Action:</b> Avoid requesting this function in this environment.</p>
8	xxxx0824	<p><b>Equate Symbol:</b> lwmRsnCodeAmode24</p> <p><b>Meaning:</b> Caller invoked service but was in 24-bit addressing mode.</p> <p><b>Action:</b> Request this function only when you are in 31-bit or in 64-bit addressing mode.</p>
8	xxxx0825	<p><b>Equate Symbol:</b> lwmRsnCodeAscModeNotPrimary</p> <p><b>Meaning:</b> Caller invoked service but was not DAT on Primary ASC mode.</p> <p><b>Action:</b> Avoid requesting this function in this environment.</p>
8	xxxx0828	<p><b>Equate Symbol:</b> lwmRsnCodeBadVersion</p> <p><b>Meaning:</b> Version number or version length field in parameter list is not valid.</p> <p><b>Action:</b> Check for possible overlay of the parameter list.</p>

## IWM4MDRG Macro

Table 77. Return and Reason Codes for the IWM4MDRG Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx082F	<p><b>Equate Symbol:</b> IwmRsnCodeWrongHome</p> <p><b>Meaning:</b> Caller invoked service from the wrong home space</p> <p><b>Action:</b> Invoke the service from the owning address space.</p>
8	xxxx0841	<p><b>Equate Symbol:</b> IwmRsnCodeXmemMode</p> <p><b>Meaning:</b> Caller invoked service but was in cross-memory mode.</p> <p><b>Action:</b> Avoid requesting this function in cross-memory mode.</p>
10	—	<p><b>Equate Symbol:</b> IwmRetCodeCompError</p> <p><b>Meaning:</b> Component error.</p> <p><b>Action:</b> Contact your system programmer.</p>

### Example

To deregister a resource from monitoring, specify:

```

IWM4MDRG RESOURCE_TKN=RSCTOKEN,          X
          RETCODE=RC,RSNCODE=RSN
*
* Storage areas
*
RSCTOKEN DS    CL8           WLM resource token
RC       DS    F             Return code
RSN      DS    F             Reason code

```

---

## Chapter 73. IWM4MINI – Initialize Monitoring Environment

IWM4MINI allows the caller to supply MVS with some or all of the work request attributes needed for the monitoring environment. The attributes include userid, transaction name, transaction class, source LU, and LU 6.2 token.

There are two types of monitoring environments available, management monitoring environments and report-only monitoring environments. Management monitoring environments provide both performance management and performance reporting. Report-only monitoring environments can be used for performance reporting only.

Use the REPORTONLY=YES parameter to specify the monitoring environment will be used for reporting purposes only.

If you are invoking IWM4MINI with the REPORTONLY=YES parameter, ASSOCIATE=ENCLAVE or ASSOCIATE=ADDRESS\_SPACE must be specified to associate the monitoring environment with an enclave or an address space.

For management monitoring environments, if possible, invoke IWM4MINI immediately following IWMCLSFY, and pass the service class for the work request. Without the associated service class in the monitoring environment, delay information cannot be accumulated and reported accurately.

IWM4MINI can be issued multiple times for the same work request. The first time you invoke IWM4MINI for a work request, you must specify MODE=RESET, otherwise the previous work request's attributes are associated with this work request. Any subsequent time you invoke IWM4MINI from the same address space for the same monitoring token for the same work request, specify MODE=RETAIN. If the caller subsystem work manager consists of multiple address spaces (with multiple monitoring tokens), the first time IWM4MINI is invoked in each address space for a given work request must specify MODE=RESET. Any subsequent invocations for the same work request should specify MODE=RETAIN.

If you are invoking IWM4MINI multiple times for the same work request, only one of the invocations should specify EXSTARTTIME=exstarttime. It is up to you to decide at which point in the subsystem work manager's processing you consider the real execution start time.

Optionally, with this macro, you can use the OWNER\_TOKEN and OWNER\_DATA parameters to specify a token for the user/owner of the monitoring environment for your own use.

### Information

Note that this service was previously called IWMMINIT for 31-bit address space only (see "IWMMINIT – Initialize Monitoring Environment" on page 832).

---

## Environment

The requirements for the caller are:

**Minimum authorization:** Either problem state or supervisor state. PSW key must either be 0 or match the value supplied on IWM4MCRE.

## IWM4MINI Macro

<b>Dispatchable unit mode:</b>	Task or SRB
<b>Cross memory mode:</b>	Any PASN, any HASN, any SASN If the key specified on IWM4MCRE was a user key (8-F), then primary addressability must be the same as when IWM4MCRE was invoked.
<b>AMODE:</b>	31- or 64-bit
<b>ASC mode:</b>	Primary Any P,S,H.
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	Locked or unlocked
<b>Control parameters:</b>	Control parameters must be in the primary address space.

---

## Programming Requirements

1. The macro IWMYCON must be included to use this macro.
2. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
3. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, IWMRSNCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.

---

## Restrictions

1. All parameter areas must reside in current primary, except that the TCB (if specified) must reside in current home.
2. Caller is responsible for error recovery.
3. Only limited checking is done against the input monitoring token.
4. If the key specified on IWM4MCRE was a user key (8-F), then the primary addressability must exist to the performance block IWM4MCRE obtained. This condition is satisfied by ensuring that current primary matches primary at the time that IWM4MCRE was invoked. If this service is invoked in a subspace, the condition may be satisfied by ensuring that the performance block is shared with the base space.
5. The caller must serialize to prevent any delay monitoring services from being invoked concurrently for the environment represented by the monitoring token.
6. The REPORTONLY and ASSOCIATE keywords may only be used on systems running z/OS R2 or higher.
7. The BPMGMTONLY keyword may only be used on systems running z/OS R8 or higher.
8. This macro supports multiple versions. Some keywords are only supported by certain versions. Refer to the PLISTVER parameter description for further information.

---

## Input Register Information

Before issuing the IWM4MINI macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

---

## Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Reason code if GR15 return code is non-zero
1	Used as work registers by the system
2-13	Unchanged
14	Used as work registers by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

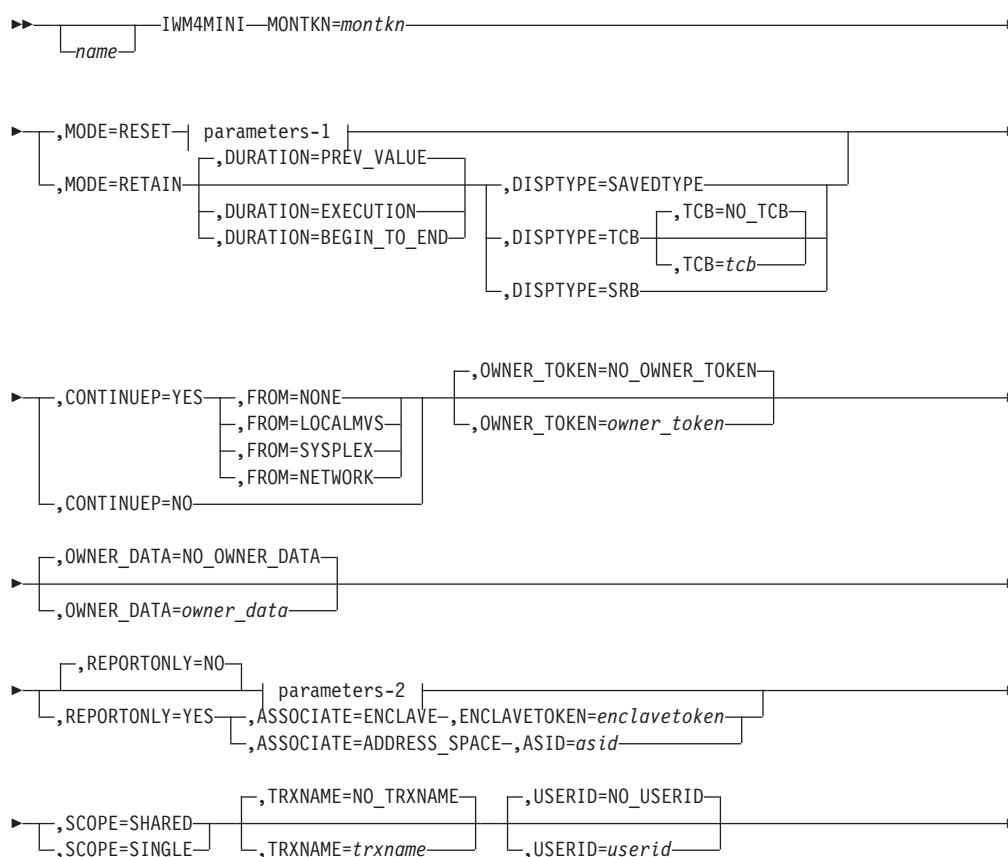
Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

## Performance Implications

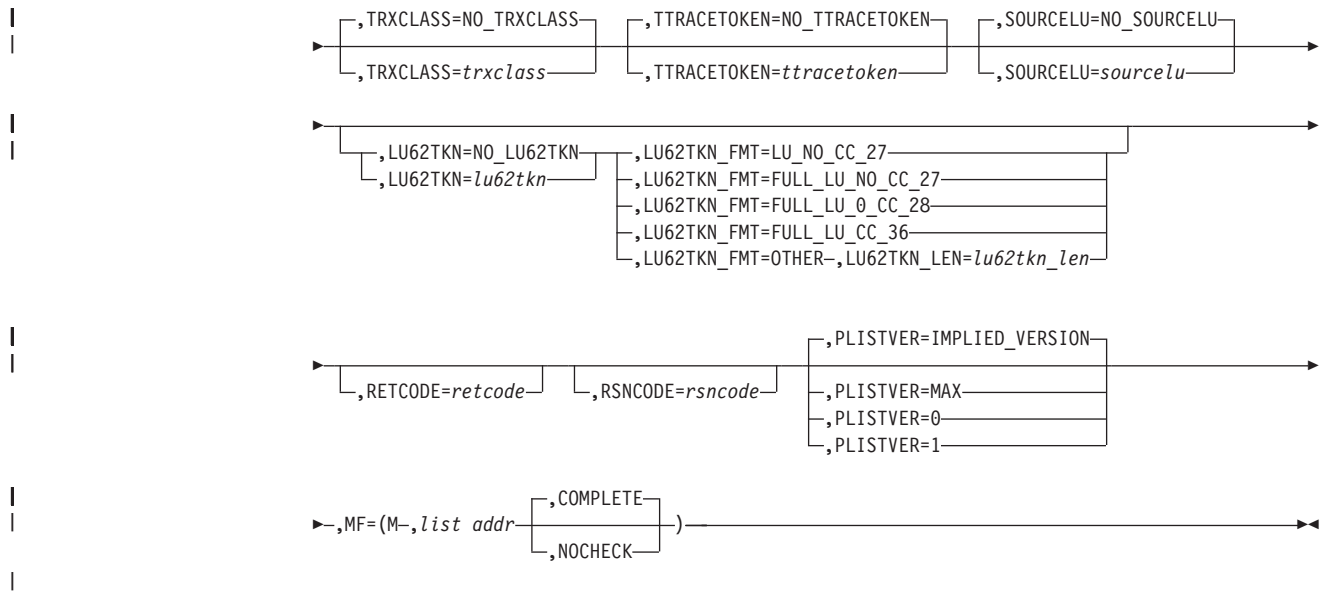
None.

## Syntax

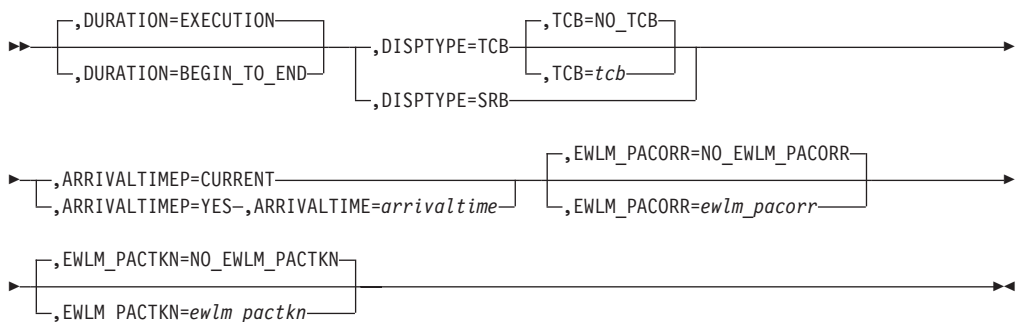
### main diagram



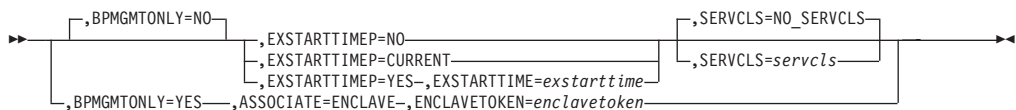
## IWM4MINI Macro



### parameters-1



### parameters-2



## Parameters

The parameters are explained as follows:

### *name*

An optional symbol, starting in column 1, that is the name on the IWM4MINI macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

### **,ARRIVALTIME=arrivaltime**

When ARRIVALTIMEP=YES and MODE=RESET are specified, a required input parameter, which contains the work arrival time in STCK format.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 64 bit field.

### **,ARRIVALTIMEP=CURRENT**



**,ARRIVALTIMEP=YES**

When MODE=RESET is specified, a required parameter, which indicates whether the work arrival time is passed.

**,ARRIVALTIMEP=CURRENT**

indicates that the current time should be supplied by the service.

**,ARRIVALTIMEP=NO**

indicates that the work arrival time is passed.

**,ASID=***asid*

When ASSOCIATE=ADDRESS\_SPACE and REPORTONLY=YES are specified, a required input parameter which contains the address space ID.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 16-bit field.

**,ASSOCIATE=ENCLAVE****,ASSOCIATE=ADDRESS\_SPACE**

When REPORTONLY=YES is specified, a required parameter, which indicates whether the monitoring environment should be associated to an enclave or an address space.

**,ASSOCIATE=ENCLAVE**

indicates that the monitoring environment should be associated to an enclave.

**,ASSOCIATE=ADDRESS\_SPACE**

indicates that the monitoring environment should be associated to an address space.

**,ASSOCIATE=ENCLAVE**

When BPMGMTONLY=YES and REPORTONLY=NO are specified, a required parameter, which indicates whether the monitoring environment should be associated only to an enclave

**,ASSOCIATE=ENCLAVE**

indicates that the monitoring environment should be associated to an enclave.

**,BPMGMTONLY=NO****,BPMGMTONLY=** $\overline{\text{YES}}$ 

When REPORTONLY=NO is specified, an optional parameter, which indicates whether the monitoring environment is for bufferpool management purposes only (YES) or not (NO). The default is BPMGMTONLY=NO.

**,BPMGMTONLY=NO**

indicates that the monitoring environment is not for bufferpool management purposes only.

**,BPMGMTONLY=YES**

indicates that the monitoring environment is for bufferpool management purposes only.

**,CONTINUEP=YES****,CONTINUEP=NO**

A required parameter, which indicates whether it is known (YES) or not (NO) that there exists another monitoring environment for this same work request.

**,CONTINUEP=YES**

indicates that the existence of a prior monitoring environment for the work request is known.

### **,CONTINUEP=NO**

indicates that it is not known whether there exists a prior monitoring environment for the work request. If MODE(RESET) is specified, no status is saved. If MODE(RETAIN) is specified, the existing status is preserved.

### **,DISPTYPE=TCB**

### **,DISPTYPE=SRB**

When MODE=RESET is specified, a required parameter, which describes the nature of the MVS dispatchable units which participate in processing work requests associated with the delay monitoring environment established by this service.

### **,DISPTYPE=TCB**

indicates that work requests run in TCB mode under a TCB within the current home address space. Note that in cross-memory mode, this may be different from the current primary address space.

### **,DISPTYPE=SRB**

indicates that work requests run in SRB mode within the current home address space.

### **,DISPTYPE=SAVEDTYPE**

### **,DISPTYPE=TCB**

### **,DISPTYPE=SRB**

When MODE=RETAIN is specified, a required parameter, which describes the nature of the MVS dispatchable units which participate in processing work requests associated with the delay monitoring environment established by this service.

### **,DISPTYPE=SAVEDTYPE**

indicates that the information saved when MODE(RESET) was used is still applicable.

### **,DISPTYPE=TCB**

indicates that work requests run in TCB mode under a TCB within the current home address space. Note that in cross-memory mode, this may be different from the current primary address space.

### **,DISPTYPE=SRB**

indicates that work requests run in SRB mode within the current home address space.

### **,DURATION=EXECUTION**

### **,DURATION=BEGIN\_TO\_END**

When MODE=RESET is specified, an optional parameter, which indicates the duration of the work request over which the delays are to be represented. The default is DURATION=EXECUTION.

### **,DURATION=EXECUTION**

indicates that the monitoring environment will reflect delays from the point where an application or transaction program is given control, i.e. the execution phase. Typically a monitoring environment with this scope would be passed to IWMMNTFY to pass the execution time for the work request.

### **,DURATION=BEGIN\_TO\_END**

indicates that the monitoring environment will reflect delays from the arrival of the work request into the MVS sysplex until its completion. Ordinarily use of this option would be in close proximity to the time when the work request is classified. Typically a monitoring environment with this duration would be passed to IWMRPT to report the total elapsed time for the work request.

**,DURATION=PREV\_VALUE****,DURATION=EXECUTION****,DURATION=BEGIN\_TO\_END**

When MODE=RETAIN is specified, an optional parameter, which indicates the duration of the work request over which the delays are to be represented. The default is DURATION=PREV\_VALUE.

**,DURATION=PREV\_VALUE**

indicates that the duration for delays has been specified on a previous invocation.

**,DURATION=EXECUTION**

indicates that the monitoring environment will reflect delays from the point where an application or transaction program is given control, for example, the execution phase. Typically a monitoring environment with this duration would be passed to IWMMNTFY to pass the execution time for the work request.

**,DURATION=BEGIN\_TO\_END**

indicates that the monitoring environment will reflect delays from the arrival of the work request into the MVS sysplex until its completion. Ordinarily use of this option would be in close proximity to the time when the work request is classified. Typically, a monitoring environment with this duration is passed to IWMRPT to report the total elapsed time for the work request.

**,ENCLAVETOKEN=*enclavetoken***

When ASSOCIATE=ENCLAVE and REPORTONLY=YES are specified, a required input parameter, which contains the enclave token.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 64-bit field.

**,EWLM\_PACORR=*ewlm\_pacorr*****,EWLM\_PACORR=NO\_EWLM\_PACORR**

When MODE=RESET is specified, an optional input parameter, which contains the cross-platform enterprise workload management (EWLM) parent correlator associated with the work request.

**Note:**

- If EWLM\_PACORR is specified and the correlator does not contain a valid ARM correlator, reason code `IwmRsnCodeInvalidEWLMCorr` is returned to the caller. Refer to Table 78 on page 624 for further information. If the correlator is a valid ARM correlator, but cannot be understood by EWLM (no EWLM format), it is ignored.
- The parameters EWLM\_PACORR and EWLM\_PACTKN are mutually exclusive.

The default is NO\_EWLM\_PACORR. It indicates that no EWLM parent correlator is passed.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a character field.

**,EWLM\_PACTKN=*ewlm\_pactkn*****,EWLM\_PACTKN=NO\_EWLM\_PACTKN**

When MODE=RESET is specified, an optional input parameter, which contains the cross-platform Enterprise Workload Management (EWLM) parent correlator token associated with the work request.

## IWM4MINI Macro

I If EWLM\_PACTKN is specified and the correlator token does not contain a valid  
I correlator token, reason code IwmRsnCodeInvalidEWLMCorr is returned to the  
I caller. Refer to Table 78 on page 624 for further information. The default is  
I NO\_EWLM\_PACTKN. It indicates that no EWLM correlator token is passed.

I **To code:** Specify the RS-type address, or address in register (2)-(12), of a  
I character field.

### **,EXSTARTTIME=exstarttime**

When EXSTARTTIMEP=YES, BPMGMTONLY=NO and REPORTONLY=NO are specified, a required input parameter, which contains the start execution time in STCK format.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 64-bit field.

### **,EXSTARTTIMEP=NO**

### **,EXSTARTTIMEP=CURRENT**

### **,EXSTARTTIMEP=YES**

I When BPMGMTONLY=NO and REPORTONLY=NO are specified, a required  
I parameter, which indicates whether the execution start time value is passed.

### **,EXSTARTTIMEP=NO**

I indicates that the execution start time value is not passed.

I If MODE(RETAIN) is specified, EXSTARTTIMEP(NO) preserves the existing  
I execution start time, if any.

### **,EXSTARTTIMEP=CURRENT**

I indicates that the current time should be supplied by the service.

### **,EXSTARTTIMEP=YES**

I indicates that the start execution time value is passed.

### **,FROM=NONE**

### **,FROM=LOCALMVS**

### **,FROM=SYSPLEX**

### **,FROM=NETWORK**

When CONTINUEP=YES is specified, a required parameter.

### **,FROM=NONE**

indicates that there is no other environment.

### **,FROM=LOCALMVS**

indicates that such an environment should exist on the current MVS.

### **,FROM=SYSPLEX**

indicates that such an environment should exist in the current sysplex, but is not expected to be on the current MVS image.

### **,FROM=NETWORK**

indicates that such an environment may exist, but is not expected to be in the current sysplex.

### **,LU62TKN=lu62tkn**

### **,LU62TKN=NO\_LU62TKN**

An optional input parameter, which contains LU 6.2 token for the work request. This is not an SNA term, but it is comprised of the following fields which are defined by SNA for the FMH5.

- Logical Unit of Work Identifier length byte, in binary, which may have the values 0 or 10-26 decimal (inclusive)
- Logical Unit of Work Identifier

- Length byte for the network qualified LU name, in binary, which may have the values 1-17 decimal (inclusive)
- Network qualified LU network name (1-17 bytes)
- Logical Unit of Work Instance Number, in binary (6 bytes)
- Logical Unit of Work Sequence Number, in binary (2 bytes)
- Conversation Correlator Field (0 to 9 bytes)
  - Length byte for the Conversation Correlator, in binary, which may have the values 0-8 decimal (inclusive)
  - Conversation Correlator of the sending transaction (1-8 bytes)

The Conversation Correlator Field (which includes its length byte) may be dropped when its length byte is 0. The default is NO\_LU62TKN. indicates that no LU 6.2 token was passed.

If MODE(RETAIN) is specified, NO\_LU62TKN will preserve the existing LU6.2 token, if any.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a character field.

**,LU62TKN\_FMT=LU\_NO\_CC\_27**

**,LU62TKN\_FMT=FULL\_LU\_NO\_CC\_27**

**,LU62TKN\_FMT=FULL\_LU\_0\_CC\_28**

**,LU62TKN\_FMT=FULL\_LU\_CC\_36**

**,LU62TKN\_FMT=OTHER**

When LU62TKN=*lu62tkn* is specified, a required parameter, which indicates the format/length of the LU 6.2 token.

**,LU62TKN\_FMT=LU\_NO\_CC\_27**

indicates that a fixed length token of 27 bytes is provided, with no conversation correlator (not even its length byte). The LU name may be 1-17 bytes. Bytes at the end of the token are padded with hexadecimal zeros, if necessary, to form a full 27 bytes.

**,LU62TKN\_FMT=FULL\_LU\_NO\_CC\_27**

indicates that the fully qualified LU name (17 bytes) is used, but no conversation correlator (not even its length byte) is provided. This format is architected to be 27 bytes long.

**,LU62TKN\_FMT=FULL\_LU\_0\_CC\_28**

indicates that the fully qualified LU name (17 bytes) is used, and the conversation correlator length byte is present and has the value 0. This format is architected to be 28 bytes long.

**,LU62TKN\_FMT=FULL\_LU\_CC\_36**

indicates that the fully qualified LU name (17 bytes) is used, and the conversation correlator is provided with a length of 8 (maximum allowed). This format is architected to be 36 bytes long.

**,LU62TKN\_FMT=OTHER**

indicates that the format of the LU 6.2 token is different from those specified by the remaining keywords.

**,LU62TKN\_LEN=*lu62tkn\_len***

When LU62TKN\_FMT=OTHER and LU62TKN=*lu62tkn* are specified, a required input parameter, which contains the length of the LU62 token. Valid values are in the range 1-36 decimal (inclusive).

**To code:** Specify the RS-type address, or address in register (2)-(12), of an one-byte field.

**,MODE=RESET**

### **,MODE=RETAIN**

A required parameter, which indicates how previous attributes established for a monitoring environment should be treated. This does not refer to (or include) attributes established in IWM4MCRE.

### **,MODE=RESET**

indicates that previous attributes should be discarded.

### **,MODE=RETAIN**

indicates that previous attributes should be retained unless explicitly specified.

### **MONTKN=*montkn***

A required input parameter, which contains the delay monitoring token

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32-bit field.

### **,OWNER\_DATA=*owner\_data***

### **,OWNER\_DATA=NO\_OWNER\_DATA**

An optional input parameter, which contains data maintained by the user/owner of the monitoring environment. The format is undefined to MVS. The default is NO\_OWNER\_DATA which indicates that no owner data is provided.

If MODE(RETAIN) is specified, NO\_OWNER\_DATA will preserve the existing owner data, if any.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32 bit field.

### **,OWNER\_TOKEN=*owner\_token***

### **,OWNER\_TOKEN=NO\_OWNER\_TOKEN**

An optional input parameter, which contains a token maintained by the user or owner of the monitoring environment. The format is undefined to MVS. The default is NO\_OWNER\_TOKEN which indicates that no owner token is provided.

If MODE(RETAIN) is specified, NO\_OWNER\_TOKEN preserves the existing owner token, if any.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32-bit field.

### **,PLISTVER=IMPLIED\_VERSION**

### **,PLISTVER=MAX**

### **,PLISTVER=0**

### **,PLISTVER=1**

An optional input parameter that specifies the version of the macro. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want to indicate the latest version currently possible.
- **0**, which supports all parameters except those specifically referenced in higher versions.
- **1**, which supports both the following parameters and those from version 0:

BPMGMTONLY

EWLM\_PACORR

EWLM\_PACTKN

**To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX
- A decimal value of 0, or 1

**REPORTONLY=NO****REPORTONLY=YES**

An optional parameter, which indicates whether the monitoring environment is for reporting purposes only (YES) or not (NO). The default is REPORTONLY=NO.

**,REPORTONLY=NO**

indicates that the monitoring environment is not for reporting purposes only.

**,REPORTONLY=YES**

indicates that the monitoring environment is for reporting purposes only.

**,RETCODE=*retcode***

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,RSNCODE=*rsncode***

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,SCOPE=SHARED****,SCOPE=SINGLE**

A required parameter, which indicates the scope of work passed.

**,SCOPE=SHARED**

indicates that multiple work requests, possibly from different service classes, could be described.

**,SCOPE=SINGLE**

indicates that only a single work request is described.

**,SERVCLS=*servcls*****,SERVCLS=NO\_SERVCLS**

When **BPMGMTONLY=NO** and **REPORTONLY=NO** are specified, an optional input parameter, which contains the service class token. The default is **NO\_SERVCLS** which indicates that no service class token was passed.

If **MODE(RETAIN)** is specified, **NO\_SERVCLS** preserves the existing service class token, if any.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32-bit field.

**,SOURCELU=*sourcelu*****,SOURCELU=NO\_SOURCELU**

An optional input parameter, which contains the LU name associated with the requestor. This may be the fully qualified **NETID.LUNAME**, for example, network name (1-8 bytes), followed by a period, followed by the LU name for the requestor (1-8 bytes). It may also be the 1-8 byte local LU name, with no network qualifier. The **SOURCELU** field may be from 1-17 characters. In the assembler form, the macro determines the length of the field as follows:

1. If the field is specified by register notation, it will be assumed to be 17 characters (padded with blanks) and a full 17 characters will be copied.
2. If the field is specified using an RS form name, then the length will be determined using the L' assembler function. When the length is less than 17

## IWM4MINI Macro

characters, the macro will pad with blanks. When the length is greater than or equal to 17 characters, the macro copies the first 17 bytes.

In the PL/AS form, the rules for the PL/AS compiler determines when to pad with blanks, for example, less than 17 characters implies padding, 17 or more implies a 17 character copy.

This is intended to be the same value as used in IWMCLSFY, and may be distinct from the LU name contained within the LU 6.2 token. For environments where the LU name may be available on some, but not all flows, provision of a data area initialized to all blanks is equivalent to specification of NO\_SOURCELU when MODE(RESET) is specified. Providing an area of all blanks when MODE(RETAIN) is specified will cause that to be used. The default is NO\_SOURCELU which indicates that no source LU name was passed.

If MODE(RETAIN) is specified, NO\_SOURCELU preserves the existing source LU name, if any.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a character field.

**,TCB=*tcb***

**,TCB=NO\_TCB**

When DISPTYPE=TCB and MODE=RESET are specified, an optional input parameter, which defines the TCB within the current home address space which will serve the work request. Note that this name is not the pointer to the TCB, but the name of the data area containing the TCB. A typical invocation might replace xTCB with TCB.

Generally, the input TCB specified should be the TCB under which the work request (for, example, a transaction program) runs and under which the delay information is recorded (in spite of the fact that task switches may occur). The default is NO\_TCB which indicates that no TCB is currently associated with the monitoring environment for this work request.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a character field.

**,TCB=*tcb***

**,TCB=NO\_TCB**

When DISPTYPE=TCB and MODE=RETAIN are specified, an optional input parameter, which defines the TCB within the current home address space which will serve the work request. Note that this name is not the pointer to the TCB, but the name of the data area containing the TCB. A typical invocation might replace xTCB with TCB.

Ordinarily the input TCB specified should be the TCB under which the work request (e.g. transaction program) runs and under which the delay information is recorded (in spite of the fact that task switches may occur). The default is NO\_TCB which indicates that no TCB is currently associated with the monitoring environment for this work request.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a character field.

**,TRXCLASS=*trxclass***

**,TRXCLASS=NO\_TRXCLASS**

An optional input parameter, which contains a class name within a subsystem. For environments where the transaction class is available on some, but not all flows, provision of a data area initialized to all blanks is equivalent to



specification of NO\_TRXCLASS when MODE(RESET) is specified. Providing an area of all blanks when MODE(RETAIN) is specified will cause that to be used. The default is NO\_TRXCLASS. indicates that no class name was passed.

If MODE(RETAIN) is specified, NO\_TRXCLASS will preserve the existing transaction class, if any.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

**,TRXNAME=***trxname*

**,TRXNAME=NO\_TRXNAME**

An optional input parameter, which contains the transaction name. For environments where the transaction name is available on some, but not all flows, provision of a data area initialized to all blanks is equivalent to specification of NO\_TRXNAME when MODE(RESET) is specified. Providing an area of all blanks when MODE(RETAIN) is specified will cause that to be used. The default is NO\_TRXNAME. indicates that no transaction name was passed.

If MODE(RETAIN) is specified, NO\_TRXNAME will preserve the existing transaction name, if any.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

**,TTRACETOKEN=***ttracetoken*

**,TTRACETOKEN=NO\_TTRACETOKEN**

An optional input parameter, which contains the transaction trace token. The default is NO\_TTRACETOKEN which indicates that no transaction trace token was passed.

If MODE(RETAIN) is specified, NO\_TTRACETOKEN preserves the existing transaction trace token, if any.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32-character field.

**,USERID=***userid*

**,USERID=NO\_USERID**

An optional input parameter, which contains the local userid associated with the work request. For environments where the user id is available on some, but not all flows, provision of a data area initialized to all blanks is equivalent to specification of NO\_USERID when MODE(RESET) is specified. Providing an area of all blanks when MODE(RETAIN) is specified will cause that to be used. The default is NO\_USERID. indicates that no userid was passed.

If MODE(RETAIN) is specified, NO\_USERID preserves the existing user id, if any.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

## ABEND Codes

None.

## Return and Reason Codes

When the IWM4MINI macro returns control to your program:

- GPR 15 (and *retcode*, when you code RETCODE) contains a return code.

## IWM4MINI Macro

- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, when you code RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 78. Return and Reason Codes for the IWM4MINI Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<code>IwmRetCodeOk</code> : Successful completion.
4	—	<code>IwmRetCodeWarning</code> : Successful completion, unusual conditions noted.
4	xxxx0402	<b>Equate Symbol:</b> <code>IwmRsnCodeNoMonEnv</code> <b>Meaning:</b> Monitoring token indicates that no monitoring environment exists.
8	—	<code>IwmRetCodeInvocError</code> : Invalid invocation environment or parameters
8	xxxx081E	<b>Equate Symbol:</b> <code>IwmRsnCodeBadLU62TknLen</code> <b>Meaning:</b> The length byte of the LU62 token has an invalid value. Only values 1-36 (decimal) are valid.
8	xxxx0820	<b>Equate Symbol:</b> <code>IwmRsnCodeBadMonEnv</code> : <b>Meaning:</b> Monitoring environment does not pass short form verification.
8	xxxx0894	<b>Equate Symbol:</b> <code>IwmRsnCodeInvalidEWLMCorr</code> <b>Meaning:</b> Passed correlator information ( <code>EWLM_PACORR</code> or <code>EWLM_PACTKN</code> ) does not pass validity checking, that means: the architected ARM correlator length field in the first two Bytes of the correlator (token) is either less than 4 ('0004'x) or greater than 512 ('0200'x). <b>Action:</b> Check the specification of the correlator information.
8	xxxx0895	<b>Equate Symbol:</b> <code>IwmRsnCodeEWLMServiceNotEnabled</code> <b>Meaning:</b> Service is not enabled because the Monitoring Environment was not created with <code>EWLM=YES</code> (either on <code>IWM4CON</code> or <code>IWM4MCRE</code> ). <b>Action:</b> Specify the parameter <code>EWLM_PACORR</code> or <code>EWLM_PACTKN</code> only when the Monitoring Environment was created with <code>EWLM=YES</code> (either on <code>IWM4CON</code> or <code>IWM4MCRE</code> ).
C	—	<code>IwmRetCodeEnvError</code> : Environmental error
C	xxxx0C07	<b>Equate Symbol:</b> <code>IwmRsnCodeNoArrTime</code> : <b>Meaning:</b> No arrival time was supplied to the service and <code>STCK</code> gave a non-zero condition code.
C	xxxx0C08	<b>Equate Symbol:</b> <code>IwmRsnCodeNoExTime</code> : <b>Meaning:</b> No execution start time was supplied to the service and <code>STCK</code> gave a non-zero condition code.

---

## Example

```
IWM4MINI MONTKN=(R9),ARRIVALTIMEP=YES,  
ARRIVALTIME=(R3),EXSTARTTIMEP=YES,  
EXSTARTTIME=(R4),DISPTYPE=TCB,TCB=(R7),  
SCOPE=SINGLE,TRXNAME=WLTRXNAME,SOURCELU=SOURCELU,  
CONTINUEP=YES,LU62TKN_FMT=OTHER,LU62TKN_LEN=LU62TKNLEN,  
LU62TKN=LU62TKN1,MODE=RESET,FROM=NONE,  
REPORTONLY=NO,RETCODE=RCODE,RSNCODE=RSN
```



---

## Chapter 74. IWM4MREG –Register a Resource For Monitoring

The purpose of the IWM4MREG service is to register a resource for delay monitoring. The service allows the caller to identify a resource which may be involved in delays to work requests. The caller's home address space is assumed to be the owner of the resource to be monitored. The system may decide to increase or decrease the size of the resource to balance the associated delays.

The resource to be monitored may require the caller to provide one or more exits to interact with the system. These exits are called in the same system key and non-cross memory environment as the program which registers the resource. The exit always receives control in AMODE31, regardless of the mode in which IWM4MREG was invoked.

The system implicitly deregisters a resource due to repetitive errors in calling exits associated with the resource. In this case, the invocation to deregister finds that the associated resource token is invalid and returns with a warning return code.

The system also cleans up its resources associated with the registration when the address space which owns the resource goes through address space termination. Therefore, no deregistration is required or honored for this case.

---

### Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Supervisor state or program key mask (PKM) allowing keys 0-7. The caller's PSW key must be in the range 0-7.
<b>Dispatchable unit mode:</b>	Task
<b>Cross-memory mode:</b>	PASN=HASN=SASN
<b>AMODE:</b>	31- or 64-bit
<b>ASC mode:</b>	Primary
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks may be held.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

---

### Programming Requirements

1. The macro IWMYCON must be included to use this macro.
2. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
3. Note that the high-order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, IWMRSNCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.
4. All character data input is assumed to be left justified and padded with blanks on the right, as needed, to fill in the specified number of bytes.

## Restrictions

NO FRRs may be established.

## Input Register Information

Before issuing the IWM4MREG macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

## Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Reason code if GR15 return code is non-zero
1	Used as work register by the system
2-13	Unchanged
14	Used as work registers by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

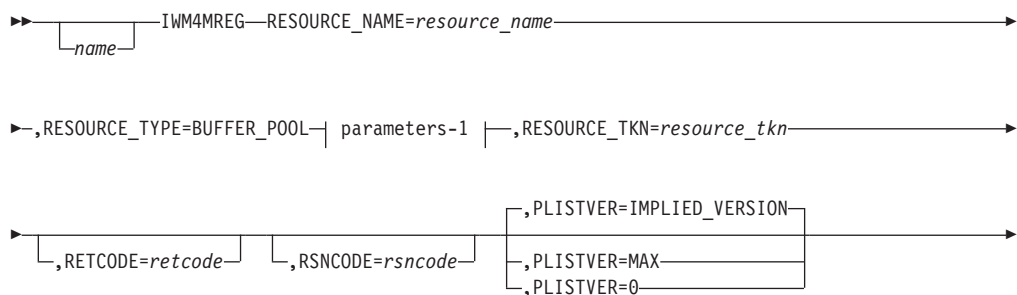
Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

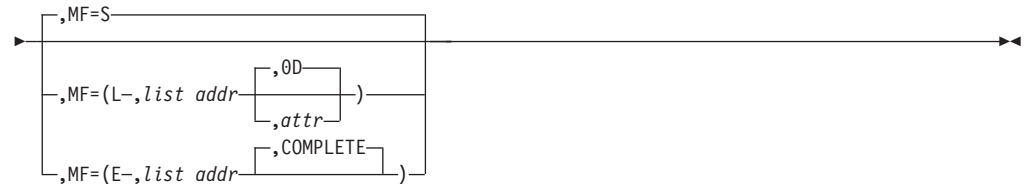
## Performance Implications

None.

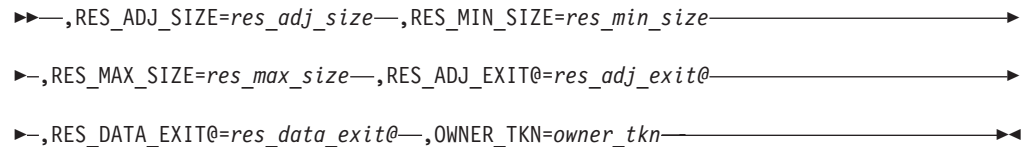
## Syntax

### main diagram





### parameters-1



## Parameters

The parameters are explained as follows:

### *name*

An optional symbol, starting in column 1, that is the name on the IWM4MREG macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

### **,MF=S**

**,MF=(L,*list addr*)**

**,MF=(L,*list addr*,*attr*)**

**,MF=(L,*list addr*,**0D**)**

**,MF=(E,*list addr*)**

**,MF=(E,*list addr*,**COMPLETE**)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

### *,list addr*

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

### *,attr*

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

### **,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

### **,OWNER\_TKN=*owner\_tkn***

When RESOURCE\_TYPE=BUFFER\_POOL is specified, a required input parameter, which contains any data associated with the resource that may be useful later when the resource adjustment exit or the resource data collection exit is called. The format is undefined to MVS.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

### **,PLISTVER=IMPLIED\_VERSION**

### **,PLISTVER=MAX**

### **,PLISTVER=0**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form, when both are assembled with the same level of the system. In this way, MAX ensures that the parameter list does not overwrite nearby storage.

- **0**, if you use the currently available parameters.

**To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX
- A decimal value of 0

### **,RES\_ADJ\_EXIT@=*res\_adj\_exit@***

When RESOURCE\_TYPE=BUFFER\_POOL is specified, a required input parameter that contains the address of the resource adjustment exit to be invoked when the system wishes to rebalance resource usage. This exit is called with the same non cross-memory environment and PSW key as when IWM4MREG is invoked.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a pointer field.

### **,RES\_ADJ\_SIZE=*res\_adj\_size***

When RESOURCE\_TYPE=BUFFER\_POOL is specified, a required input parameter, which contains the minimum size (in 4K pages) by which the specified resource can be adjusted. For a bufferpool, this corresponds to a product external or the size of a cell.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.



**,RES\_DATA\_EXIT@=*res\_data\_exit@***

When RESOURCE\_TYPE=BUFFER\_POOL is specified, a required input parameter that contains the address of the resource data collection exit to be invoked when the system checks how physical resources relate to effectiveness of the given resource. This exit is called with the same non cross-memory environment and PSW key as when IWM4MREG is invoked.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a pointer field.

**,RES\_MAX\_SIZE=*res\_max\_size***

When RESOURCE\_TYPE=BUFFER\_POOL is specified, a required input parameter, which contains the maximum size (in 4K pages) associated with the specified resource.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

**,RES\_MIN\_SIZE=*res\_min\_size***

When RESOURCE\_TYPE=BUFFER\_POOL is specified, a required input parameter, which contains the minimum size (in 4K pages) associated with the specified resource.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

**RESOURCE\_NAME=*resource\_name***

A required input parameter, which contains the resource name associated with the resource to be registered. The value should be padded on the right with blanks for any unused characters.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 16-character field.

**RESOURCE\_TKN=*resource\_tkn***

A required input parameter, which contains the associated WLM resource token which is returned by the resource monitoring registration service (IWM4MREG).

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

**,RESOURCE\_TYPE=BUFFER\_POOL**

A required parameter, which indicates the type of resource being registered.

**,RESOURCE\_TYPE=BUFFER\_POOL**

indicates that a bufferpool is being registered.

**,RETCODE=*retcode***

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,RSNCODE=*rsncode***

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

---

## ABEND Codes

None.

## Return and Reason Codes

When the IWM4MREG macro returns control to your program:

- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, if you coded RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 79. Return and Reason Codes for the IWM4MREG Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> <code>IwmRetCodeOk</code> <b>Meaning:</b> Successful completion. <b>Action:</b> None required.
8	—	<b>Equate Symbol:</b> <code>IwmRetCodeInvocError</code> <b>Meaning:</b> Invalid invocation environment or parameters.
8	xxxx0801	<b>Equate Symbol:</b> <code>IwmRsnCodeSRBMode</code> <b>Meaning:</b> Caller is in SRB mode. <b>Action:</b> Avoid requesting this function in SRB mode.
8	xxxx0803	<b>Equate Symbol:</b> <code>IwmRsnCodeDisabled</code> <b>Meaning:</b> Caller is disabled. <b>Action:</b> Avoid requesting this function while disabled.
8	xxxx0804	<b>Equate Symbol:</b> <code>IwmRsnCodeLocked</code> <b>Meaning:</b> Caller is locked. <b>Action:</b> Avoid requesting this function while locked.
8	xxxx080B	<b>Equate Symbol:</b> <code>IwmRsnCodeBadPl</code> <b>Meaning:</b> Error accessing parameter list. <b>Action:</b> Check for possible storage overlay.
8	xxxx0810	<b>Equate Symbol:</b> <code>IwmRsnCodeEutFrr</code> <b>Meaning:</b> Caller has EUT FRR established. <b>Action:</b> Avoid requesting this function with an EUT FRR.
8	xxxx0823	<b>Equate Symbol:</b> <code>IwmRsnCodeDatoff</code> <b>Meaning:</b> Caller invoked service while DATOFF <b>Action:</b> Avoid requesting this function in this environment.
8	xxxx0824	<b>Equate Symbol:</b> <code>IwmRsnCodeAmode24</code> <b>Meaning:</b> Caller invoked service but was in 24 bit addressing mode. <b>Action:</b> Request this function only when you are in 31 bit addressing mode.

Table 79. Return and Reason Codes for the IWM4MREG Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0825	<b>Equate Symbol:</b> IwmRsnCodeAscModeNotPrimary  <b>Meaning:</b> Caller invoked service but was not DAT on Primary ASC mode.  <b>Action:</b> Avoid requesting this function in this environment.
8	xxxx0828	<b>Equate Symbol:</b> IwmRsnCodeBadVersion  <b>Meaning:</b> Version number or version length field in parameter list is not valid.  <b>Action:</b> Check for possible overlay of the parameter list.
8	xxxx0841	<b>Equate Symbol:</b> IwmRsnCodeXmemMode  <b>Meaning:</b> Caller invoked service but was in cross-memory mode.  <b>Action:</b> Avoid requesting this function in cross-memory mode.
8	xxxx0846	<b>Equate Symbol:</b> IwmRsnCodeNoUserKeyReg  <b>Meaning:</b> Caller invoked service but was in user key.  <b>Action:</b> Request this function in system key (0-7).
8	xxxx08A1	<b>Equate Symbol:</b> IwmRsnCodeBadBPMinMaxSize  <b>Meaning:</b> Maximum resource size is lower than the minimum size.  <b>Action:</b> Specify a maximum size value at least as high as the minim size value
C	—	<b>Equate Symbol:</b> IwmRetCodeEnvError: <b>Meaning:</b> Environmental error. <b>xxxx0C01</b> IwmRsnCodeNoStg: Storage is not available for the request.
10	—	<b>Equate Symbol:</b> IwmRetCodeCompError  <b>Meaning:</b> Component error.  <b>Action:</b> Contact your system programmer.

## Example

To register a resource for delay monitoring, specify:

```

IWM4MREG RESOURCE_NAME=RSCNAME,           X
          RESOURCE_TYPE=BUFFER_POOL,       X
          RES_ADJ_SIZE=RESADJSIZE,         X
          RES_MIN_SIZE=RESMINSIZE,         X
          RES_MAX_SIZE=RESMAXSIZE,         X
          RES_ADJ_EXIT@=ADJEXIT@,         X
          RES_DATA_EXIT@=DATAEXIT@,        X
          OWNER_TKN=OWNERTKN,             X
          RESOURCE_TKN=RSCTOKEN, RETCODE=RC, RSNCODE=RSN

```

\*

\* Storage areas

\*

```

RSCNAME DS CL16           Contains the resource name
RESADJSIZE DS FL4         contains the minimum size by
                           which the specified resource

```

\*

## IWM4MREG Macro

	*			can be adjusted
	RESMIN SIZE DS	FL4		contains the minimum size
	*			associated with the specified
	*			resource
	RESMAX SIZE DS	FL4		contains the maximum size
	*			associated with the specified
	*			resource
	ADJEXIT@ DS	AL4		contains the address of the
	*			Resource Adjustment Exit to be
	*			invoked when the system wishes
	*			to rebalance resource usage
	DATAEXIT@ DS	AL4		contains the address of the
	*			Resource Data Collection Exit
	*			to be invoked when the system
	*			wishes to understand how
	*			physical resources relate to
	*			effectiveness of the given
	*			resource
	OWNERTKN DS	CL8		Contains data maintained by
	*			the user
	RSCTOKEN DS	CL8		WLM resource token
	RC DS	F		Return code
	RSN DS	F		Reason code

---

## Chapter 75. IWM4QDE – Deleting a Request from the Queue for An Execution Address Space

This service deletes a work request that was previously inserted using the IWM4SSL service, if it has not been selected using the IWM4SSL service.

### Information

Note that this service was previously called IWMQDEL for 31-bit address space only (see “IWMQDEL – Deleting a Request from the Queue for An Execution Address Space” on page 846).

---

## Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Supervisor state or program key mask (PKM) allowing keys 0-7.
<b>Dispatchable unit mode:</b>	Task or SRB
<b>Cross memory mode:</b>	Any HASN, any SASN. PASN must be the address space which connected to WLM (i.e. the address space that was home when IWM4CON was issued for Queue_Manager=Yes, or Server_Manager=Yes with Server_Type=Queue).
<b>AMODE:</b>	31- or 64-bit
<b>ASC mode:</b>	Primary
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks may be held.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

---

## Programming Requirements

1. The macro CVT must be included to use this macro.
2. The macro IWMYCON must be included to use this macro.
3. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
4. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, IWMRSNCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.

---

## Restrictions

None.

---

## Input Register Information

Before issuing the IWM4QDE macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.



the use of WLM Work Queuing services as returned by IWM4CON (specifying Queue\_Manager=Yes, or Server\_Manager=Yes with Server\_Type=Queue).

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32 bit field.

**,MF=S**

**,MF=(L,list addr)**

**,MF=(L,list addr,attr)**

**,MF=(L,list addr,0D)**

**,MF=(E,list addr)**

**,MF=(E,list addr,COMPLETE)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

*,list addr*

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

*,attr*

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

**,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

**,PLISTVER=IMPLIED\_VERSION**

**,PLISTVER=MAX**

**,PLISTVER=0**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that

## IWM4QDE Macro

the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form; in this way, MAX ensures that the parameter list does not overwrite nearby storage.

- 0, if you use the currently available parameters.

**To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX
- A decimal value of 0

**,RETCODE=retcode**

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,RSNCODE=rsncode**

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,WLMWUTKN=wlmwutkn**

A required input parameter, specifying the work unit to be deleted. This token must be a token that was returned on a prior IWM4QIN request.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 16-character field.

---

## ABEND Codes

None.

---

## Return and Reason Codes

When the IWM4QDE macro returns control to your program:

- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, if you coded RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 80. Return and Reason Codes for the IWM4QDE Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> IwmRetCodeOk <b>Meaning:</b> Successful completion. <b>Action:</b> None required.
4	—	<b>Equate Symbol:</b> IwmRetCodeWarning <b>Meaning:</b> Successful completion, unusual conditions noted.
4	xxxx0408	<b>Equate Symbol:</b> IwmRsnCodeWorkNotFound <b>Meaning:</b> No work matching the input search criteria was found. <b>Action:</b> None required.



Table 80. Return and Reason Codes for the IWM4QDE Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	—	<b>Equate Symbol:</b> IwmRetCodeInvocError <b>Meaning:</b> Invalid invocation environment or parameters.
8	xxxx0803	<b>Equate Symbol:</b> IwmRsnCodeDisabled <b>Meaning:</b> Caller is disabled. <b>Action:</b> Avoid requesting this function while disabled.
8	xxxx0804	<b>Equate Symbol:</b> IwmRsnCodeLocked <b>Meaning:</b> Caller is locked. <b>Action:</b> Avoid requesting this function while locked.
8	xxxx080B	<b>Equate Symbol:</b> IwmRsnCodeBadPI <b>Meaning:</b> Error accessing parameter list. <b>Action:</b> Check for possible storage overlay.
8	xxxx0821	<b>Equate Symbol:</b> IwmRsnCodeBadConn <b>Meaning:</b> Input connect token does not pass validity checking. <b>Action:</b> Make sure to use the connect token returned by the IWM4CON service requesting Queue_Manager=Yes, or Server_Manager=Yes with Server_Type=Queue.
8	xxxx0823	<b>Equate Symbol:</b> IwmRsnCodeDatoff <b>Meaning:</b> Caller invoked service while DATOFF <b>Action:</b> Avoid requesting this function in this environment.
8	xxxx0824	<b>Equate Symbol:</b> IwmRsnCodeAmode24 <b>Meaning:</b> Caller invoked service but was in 24 bit addressing mode. <b>Action:</b> Request this function only when you are in 31 bit addressing mode.
8	xxxx0825	<b>Equate Symbol:</b> IwmRsnCodeAscModeNotPrimary <b>Meaning:</b> Caller invoked service but was not DAT on Primary ASC mod <b>Action:</b> Avoid requesting this function in this environment.
8	xxxx0827	<b>Equate Symbol:</b> IwmRsnCodeRsvdNot0 <b>Meaning:</b> Reserved field in parameter list was non-zero. <b>Action:</b> Check for use of keywords that are not supported by the MVS release on which the program is running.
8	xxxx0828	<b>Equate Symbol:</b> IwmRsnCodeBadVersion <b>Meaning:</b> Version number in parameter list is not valid. <b>Action:</b> Check for possible storage overlay of the parameter list.

## IWM4QDE Macro

Table 80. Return and Reason Codes for the IWM4QDE Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx083F	<p><b>Equate Symbol:</b> IwmRsnCodePrimaryNotOwnConn</p> <p><b>Meaning:</b> Primary address space does not own the passed connect token.</p> <p><b>Action:</b> Avoid requesting this function while primary address space does not own the connect token.</p>
8	xxxx0840	<p><b>Equate Symbol:</b> IwmRsnCodeServiceNotEnabled</p> <p><b>Meaning:</b> Caller's space connection is not enabled for this service.</p> <p><b>Action:</b> Make sure that Queue_Manager=Yes, or Server_Manager=Yes with Server_Type=Queue is specified on the IWM4CON request to enable this service.</p>
8	xxxx0848	<p><b>Equate Symbol:</b> IwmRsnCodeBadWorkUnitToken</p> <p><b>Meaning:</b> The work unit token is not valid.</p> <p><b>Action:</b> Check the specification of the WLMWUTKN parameter.</p>
10	—	<p><b>Equate Symbol:</b> IwmRetCodeCompError</p> <p><b>Meaning:</b> Component error.</p> <p><b>Action:</b> Contact your system programmer.</p>

## Example

To delete a work request from the WLM queue manager queues, specify:

```

IWM4QDE CONNTKN=CONNTOKEN, X
        WLMWUTKN=WLMWUTKN, RETCODE=RC, RSNCODE=RSN
*
* Storage areas
*
CONNTOKEN DS  FL4           Contains the connect token
*                               associated with the use of WLM
*                               Queuing services as returned by
*                               IWM4CON
*                               (specifying QUEUE_MANAGER=YES
*                               or SERVER_MANAGER=YES
*                               SERVER_TYPE=QUEUE
WLMWUTKN DS  CL16           Work unit token
RC        DS  F             Return code
RSN       DS  F             Reason code

```

---

## Chapter 76. IWM4QIN – Inserting a Request Onto the Queue for An Execution Address space

The IWM4QIN service inserts a work request onto workload management queues so its execution in a server address space can be managed by WLM.

Before using this service, the caller must connect to WLM using the IWM4CON service, specifying Queue\_Manager=Yes, or Server\_Manager=Yes with Server\_Type=Queue.

The IWM4QIN service requires the use of enclaves to manage the performance goals and reporting of work. It requires the use of application environments to associate types of work requests with servers capable of processing them.

### Information

Note that this service was previously called IWMQINS for 31-bit address space only (see “IWMQINS – Inserting a Request Onto the Queue for An Execution Address space” on page 852).

---

## Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Supervisor state or program key mask (PKM) allowing keys 0-7.
<b>Dispatchable unit mode:</b>	Task or SRB
<b>Cross memory mode:</b>	Any HASN, any SASN. PASN must be the address space which connected to WLM (i.e. the address space that was home when IWM4CON was issued for Queue_Manager=Yes, or Server_Manager=Yes with Server_Type=Queue).
<b>AMODE:</b>	31- or 64-bit
<b>ASC mode:</b>	Primary
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks may be held.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

---

## Programming Requirements

1. The macro CVT must be included to use this macro.
2. The macro IWMYCON must be included to use this macro.
3. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
4. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, IWMSRNCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.

## IWM4QIN Macro

5. All character data, unless otherwise specified, is assumed to be left justified and padded with blanks on the right, as needed, to occupy the specified number of bytes.

---

## Restrictions

None.

---

## Input Register Information

Before issuing the IWM4QIN macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

---

## Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Reason code if GR15 return code is non-zero
1	Used as work registers by the system
2-13	Unchanged
14	Used as work registers by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

---

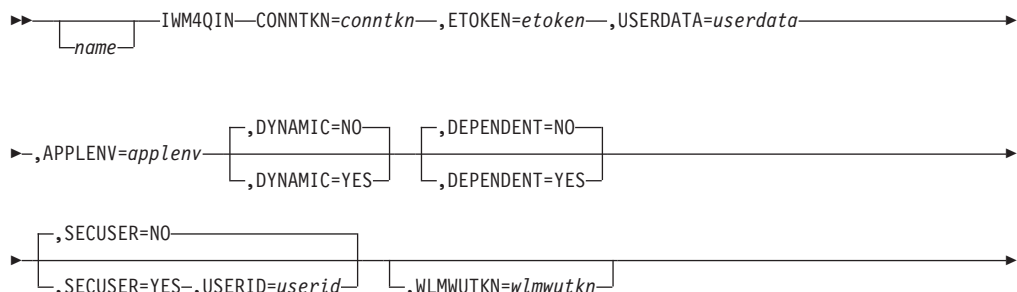
## Performance Implications

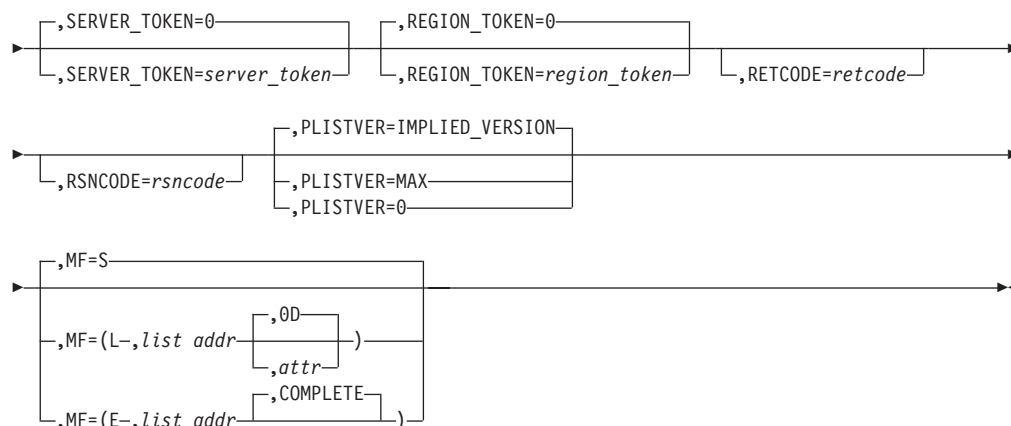
None.

---

## Syntax

### main diagram





## Parameters

The parameters are explained as follows:

### *name*

An optional symbol, starting in column 1, that is the name on the IWM4QIN macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

### **,APPLENV=applenv**

A required input parameter, which contains an application environment name. An application environment is defined in the workload manager service definition and instructs WLM how to create a server address space.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32-character field.

### **CONNTKN=conntkn**

A required input parameter, which contains the connect token returned by the IWM4CON macro.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32 bit field.

### **,DEPENDENT=NO**

### **,DEPENDENT=YES**

An optional parameter indicating whether the insert is for a dependent or a standard request. The default is DEPENDENT=NO.

### **,DEPENDENT=NO**

The request is for an independent (standard) work request (default).

### **,DEPENDENT=YES**

The insert is for a dependent work request which is required by already active server tasks to complete their processing. The request is prioritized above requests which are not marked as dependent.

### **,DYNAMIC=NO**

### **,DYNAMIC=YES**

An optional parameter indicating whether the insert is for a dynamic or static application environment. The default is DYNAMIC=NO.

### **,DYNAMIC=NO**

The server manager connects to a static application environment according to the WLM service definition. This is the default.

### **,DYNAMIC=YES**

The server manager connects to a dynamic application environment according to a prior definition via IWM4AEDF service.

### **,ETOKEN=*etoken***

A required input parameter, which contains the enclave token associated with the work request. An enclave token is obtained using either the IWM4ECRE or IWMESQRY macro.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

### **,MF=S**

**,MF=(L,*list addr*)**

**,MF=(L,*list addr,attr*)**

**,MF=(L,*list addr,0D*)**

**,MF=(E,*list addr*)**

**,MF=(E,*list addr,COMPLETE*)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

**,*list addr***

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

**,*attr***

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

### **,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

### **,PLISTVER=IMPLIED\_VERSION**

**,PLISTVER=MAX**

**,PLISTVER=0**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.

- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify `PLISTVER=MAX` on the list form of the macro. Specifying `MAX` ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form, when both are assembled with the same level of the system. In this way, `MAX` ensures that the parameter list does not overwrite nearby storage.

- **0**, which supports all parameters except those specifically referenced in higher versions.

**To code:** Specify one of the following:

- `IMPLIED_VERSION`
- `MAX`
- A decimal value of 0

**,REGION\_TOKEN=region\_token**

**,REGION\_TOKEN=0**

An optional input parameter, which contains a region token returned by the `IWM4SSL` macro. Use `REGION_TOKEN` to queue a work request to a specific server region. Such a work request is considered to be part of a set of work requests which all need access to the same status information which is kept in the virtual storage of the server region.

The following qualifications apply when specifying a region token:

- The application is responsible for passing the region token to the queueing manager so that it can insert the work request to the region.
- WLM has to know that temporal affinities for work requests to a specific server region exist in order not to stop the server region. The application must use the `IWM4TAF` macro to tell WLM when a temporal affinity starts and when it ends.

Coding `REGION_TOKEN=0` is equivalent to omitting the `REGION_TOKEN` keyword. The default is 0.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 16-character field.

**,RETCODE=retcode**

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,RSNCODE=rsncode**

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,SECUSER=NO**

**,SECUSER=YES**

An optional parameter, which specifies whether the security environment of the user should be associated with the request at run time. The default is `SECUSER=NO`.

**,SECUSER=NO**

No security environment to be established.

## IWM4QIN Macro

### **,SECUSER=YES**

Use the specified userid to establish a security environment.

### **,SERVER\_TOKEN=server\_token**

### **,SERVER\_TOKEN=0**

An optional input parameter, which contains a server token returned by the IWM4SSL macro. Use SERVER\_TOKEN to queue a secondary work request to the same server task that selected a prior work request. A secondary work request is considered to be an extension of the prior work request.

The following qualifications apply when specifying a server token:

- The server task is responsible for passing the server token to the queueing manager so that it can insert a secondary work request.
- Coordination is required between the queueing manager and the server task so that the server task knows when to expect secondary work requests. The server task uses the IWM4SSM macro to select secondary work requests. It must select all secondary work requests before it can resume normal selection using IWM4SSL.
- The same application environment and enclave token passed for the original work request must be passed for each secondary work request.
- A secondary work request cannot be deleted using the IWM4QDE macro. IWM4QIN does not return a work unit token (WLMWUTKN).
- The SECUSER keyword is ignored.

Coding SERVER\_TOKEN=0 is equivalent to omitting the SERVER\_TOKEN keyword. The default is 0.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32-character field.

### **,USERDATA=userdata**

A required input parameter, which contains data to pass to the server address space. This user data is returned to the caller of the IWM4SSL or IWM4SSM macro. The format is undefined to MVS.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 16-character field.

### **,USERID=userid**

When SECUSER=YES is specified, a required input parameter, which contains the requester's userid.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

### **,WLMWUTKN=wlmwutkn**

An optional output parameter, which will receive the work unit token. This token can be passed to the IWM4QDE service to delete the work request.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 16-character field.

---

## ABEND Codes

None.



## Return and Reason Codes

When the IWM4QIN macro returns control to your program:

- GPR 15 (and *retcode*, when you code RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, when you code RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 81. Return and Reason Codes for the IWM4QIN Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> <code>IwmRetCodeOk</code> <b>Meaning:</b> Successful completion. <b>Action:</b> None required.
4	—	<b>Equate Symbol:</b> <code>IwmRetCodeWarning</code> <b>Meaning:</b> Successful completion, unusual conditions noted.
4	xxxx042E	<b>Equate Symbol:</b> <code>IwmRsnCodeServerNotFound</code> <b>Meaning:</b> The server token does not identify an existing server task. The server task may have terminated since the token was obtained. <b>Action:</b> If the server task has not terminated, check that the correct token is specified.
4	xxxx043A	<b>Equate Symbol:</b> <code>IwmRsnCodeRegionNotFound</code> <b>Meaning:</b> The region token does not identify a valid server region. <b>Action:</b> Please specify the correct region token.
8	—	<b>Equate Symbol:</b> <code>IwmRetCodeInvocError</code> <b>Meaning:</b> Invalid invocation environment or parameters.
8	xxxx0803	<b>Equate Symbol:</b> <code>IwmRsnCodeDisabled</code> <b>Meaning:</b> Caller is disabled. <b>Action:</b> Avoid requesting this function while disabled.
8	xxxx0804	<b>Equate Symbol:</b> <code>IwmRsnCodeLocked</code> <b>Meaning:</b> Caller is locked. <b>Action:</b> Avoid requesting this function while locked.
8	xxxx080B	<b>Equate Symbol:</b> <code>IwmRsnCodeBadPI</code> <b>Meaning:</b> Error accessing parameter list. <b>Action:</b> Check for possible storage overlay.
8	xxxx0821	<b>Equate Symbol:</b> <code>IwmRsnCodeBadConn</code> <b>Meaning:</b> Input connect token does not pass validity checking. <b>Action:</b> Make sure to use the connect token returned by the IWM4CON service requesting <code>Queue_Manager=Yes</code> , or <code>Server_Manager=Yes</code> with <code>Server_Type=Queue</code> .

Table 81. Return and Reason Codes for the IWM4QIN Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0823	<p><b>Equate Symbol:</b> IwmRsnCodeDatoff</p> <p><b>Meaning:</b> Caller invoked service while DATOFF</p> <p><b>Action:</b> Avoid requesting this function in this environment.</p>
8	xxxx0824	<p><b>Equate Symbol:</b> IwmRsnCodeAmode24</p> <p><b>Meaning:</b> Caller invoked service but was in 24 bit addressing mode.</p> <p><b>Action:</b> Request this function only when you are in 31 bit addressing mode.</p>
8	xxxx0825	<p><b>Equate Symbol:</b> IwmRsnCodeAscModeNotPrimary</p> <p><b>Meaning:</b> Caller invoked service but was not in primary ASC mode.</p> <p><b>Action:</b> Avoid requesting this function in this environment.</p>
8	xxxx0827	<p><b>Equate Symbol:</b> IwmRsnCodeRsvdNot0</p> <p><b>Meaning:</b> Reserved field in parameter list was non-zero.</p> <p><b>Action:</b> Check for use of keywords that are not supported by the MVS release on which the program is running.</p>
8	xxxx0828	<p><b>Equate Symbol:</b> IwmRsnCodeBadVersion</p> <p><b>Meaning:</b> Version number in parameter list is not valid.</p> <p><b>Action:</b> Check for possible storage overlay of the parameter list.</p>
8	xxxx083A	<p><b>Equate Symbol:</b> IwmRsnCodeBadEnclave</p> <p><b>Meaning:</b> Enclave token in parameter list is not valid.</p> <p><b>Action:</b> Check the specification of the ETOKEN parameter.</p>
8	xxxx083F	<p><b>Equate Symbol:</b> IwmRsnCodePrimaryNotOwnConn</p> <p><b>Meaning:</b> Primary address space does not own the passed connect token.</p> <p><b>Action:</b> Ensure that the primary address space has previously connected to WLM using the IWM4CON macro. Ensure that the connect token returned by the IWM4CON macro is passed to the IWM4QIN macro.</p>
8	xxxx0840	<p><b>Equate Symbol:</b> IwmRsnCodeServiceNotEnabled</p> <p><b>Meaning:</b> Caller's space connection is not enabled for this service</p> <p><b>Action:</b> Make sure that Queue_Manager=Yes, or Server_Manager=Yes with Server_Type=Queue is specified on the IWM4CON request to enable this service.</p>
8	xxxx0842	<p><b>Equate Symbol:</b> IwmRsnCodeNoWLMConnect</p> <p><b>Meaning:</b> Caller's space disconnected from WLM during processing of the insert request.</p> <p><b>Action:</b> None.</p>

Table 81. Return and Reason Codes for the IWM4QIN Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0845	<p><b>Equate Symbol:</b> IwmRsnCodeWrongEnclave</p> <p><b>Meaning:</b> The caller tried to queue a secondary work request to a specific server task using the SERVER_TOKEN parameter. The caller's enclave token does not match the enclave token of the last work request selected by the server task.</p> <p><b>Action:</b> Check that the correct enclave token was specified. Check that the server task is invoking the IWM4SSL and IWM4SSM macros in the correct sequence.</p>
8	xxxx0892	<p><b>Equate Symbol:</b> IwmRsnCodeDupANameInsert</p> <p><b>Meaning:</b> The caller tried to insert to an application environment with a duplicate name within the same node concurrently.</p> <p><b>Action:</b> Make sure not to insert requests to a dynamic and static application environment within the same node concurrently.</p>
C	—	<p><b>Equate Symbol:</b> IwmRetCodeEnvError</p> <p><b>Meaning:</b> Environmental error.</p>
C	xxxx0C01	<p><b>Equate Symbol:</b> IwmRsnCodeNoStg</p> <p><b>Meaning:</b> Storage is not available for the request.</p> <p><b>Action:</b> Contact your system programmer. There is a common storage shortage.</p>
C	xxxx0C16	<p><b>Equate Symbol:</b> IwmRsnCodeServerUnavail</p> <p><b>Meaning:</b> A server cannot be started to process the work request. This is probably caused by one of the following:</p> <ol style="list-style-type: none"> <li>1. An error in the JCL procedure used to start the server address space.</li> <li>2. Repeated, unexpected terminations of the server address space.</li> </ol> <p>In either of these cases, workload management stops the application environment. A DISPLAY WLM command shows this state as INTERNALLY STOPPED.</p> <p><b>Action:</b> Look at the system log to determine what caused the error:</p> <ol style="list-style-type: none"> <li>1. If it is a JCL error, correct the error in the procedure.</li> <li>2. If it is repeated terminations of the server address space, correct the application error causing the termination.</li> </ol> <p>In either case, the server environment can then be resumed using the VARY operator command: V WLM,APPLENV=nnn,RESUME where nnn is the applicable application environment name.</p> <p><b>Note:</b> A re-IPL of some or all of the systems in the sysplex does not reset the stopped state of the application environment. The VARY command is the only way to resume the environment.</p>

Table 81. Return and Reason Codes for the IWM4QIN Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
C	xxxx0C1A	<p><b>Equate Symbol:</b> IwmRsnCodeApplNotDefined</p> <p><b>Meaning:</b> The application environment name is not defined in the active WLM policy.</p> <p><b>Action:</b> Check whether the correct application environment name is being used. If so, a service administrator must define the application environment in the WLM service definition.</p>
C	xxxx0C1B	<p><b>Equate Symbol:</b> IwmRsnCodeApplNotSST</p> <p><b>Meaning:</b> The application environment name is defined for use by a different subsystem type in the active WLM policy.</p> <p><b>Action:</b> Check whether the correct application environment name is being used. If so, a service administrator must change the application environment in the WLM service definition to specify the correct subsystem type.</p>
C	xxxx0C1D	<p><b>Equate Symbol:</b> IwmRsnCodeQMGrNotActive</p> <p><b>Meaning:</b> The required Queue Manager is not active.</p> <p><b>Action:</b> The Queue Manager with the same subsystem type and name as the server must be started and connected to workload management before the request can be honored.</p>
C	xxxx0C22	<p><b>Equate Symbol:</b> IwmRsnCodeApplEnvQuiesced</p> <p><b>Meaning:</b> For server applications connecting to WLM with subsystem type IWEB only: The application environment has been quiesced. The work request is not inserted to the WLM work queue.</p> <p><b>Action:</b> Resume the application environment.</p>
C	xxxx0C40	<p><b>Equate Symbol:</b> IwmRsnCodeNoSafCheckPossible</p> <p><b>Meaning:</b> MLS is active but a security check could not be performed probably for one of the following reasons:</p> <ul style="list-style-type: none"> <li>• No security decision could be made. The RACF router was not loaded; the request, resource, subsystem combination could not be found in the RACF ROUTER table..</li> <li>• A resource or class name is not defined to RACF or the class has not been RAClisted.</li> <li>• The class was RAClisted, but the data space cannot be accessed due to an ALESERV failure.</li> <li>• The class was RAClisted, but the data space has been deleted.</li> <li>• No security decision could be made. The RACF router was not loaded, the request ,resource, subsystem combination could not be found in the RACF ROUTER table.</li> </ul> <p><b>Action:</b> Contact your RACF Security Administrator. Check if RACF is properly installed, configured and tuned. Correct the eventual problems.</p>

Table 81. Return and Reason Codes for the IWM4QIN Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
C	xxxx0C41	<p><b>Equate Symbol:</b> IwmRsnCodeSafCheckFailed</p> <p><b>Meaning:</b> MLS is active. Queue Manager and Server Manager are not authorized to communicate.</p> <p><b>Action:</b> Normally none. If QM and SM really must communicate, contact your RACF Security Administrator. Set the appropriate Security Labels.</p>
C	xxxx0C42	<p><b>Equate Symbol:</b> IwmRsnCodeAletError</p> <p><b>Meaning:</b> Error while accessing access list with ALESERV probably because of one of the following:</p> <ul style="list-style-type: none"> <li>• The current access list cannot be expanded. There are no free access list entries and the maximum size has been reached.</li> <li>• ALESERV could not obtain storage for an expanded access list.</li> </ul> <p><b>Action:</b> Delete unused entries and reissue the request in first case. Free some storage and retry the request in second case. Contact your System Programmer if none works.</p>
10	—	<p><b>Equate Symbol:</b> IwmRetCodeCompError</p> <p><b>Meaning:</b> Component error.</p> <p><b>Action:</b> Contact your system programmer.</p>

## Example

To insert a work request onto the WLM queue manager queues, specify:

```

IWM4QIN  CONNTKN=CONNTOKEN,ETOKEN=ENCTOKEN,      X
         USERDATA=USERDATA,APPLENV=APPLENV,SECUSER=NO, X
         WLMWUTKN=WLMWUTKN,RETCODE=RC,RSNCODE=RSN
*
* Storage areas
*
CONNTOKEN DS   FL4           Contains the connect token
*                               associated with the use of WLM
*                               Queuing services as returned by
*                               IWM4CON
*                               (specifying QUEUE_MANAGER=YES
*                               or SERVER_MANAGER=YES
*                               SERVER_TYPE=QUEUE
ENCTOKEN DS   CL8           Contains the Enclave token
*                               associated with the work
*                               request as returned by IWM4ECRE
USERDATA DS   CL16          Contains data maintained by the
*                               user
APPLENV DS    CL32          Contains the application
*                               environment name
WLMWUTKN DS   CL16          Work unit token
RC DS        F              Return code
RSN DS        F              Reason code

```

## IWM4QIN Macro

---

## Chapter 77. IWM4SLI – Application Environment Limit Service

The IWM4SLI service should be used to tell WLM the total number of server instances which are supported by the application. WLM will ensure that no more server instances will be started in the system.

In addition the caller can define a minimum number of servers which should be made available by WLM regardless of whether work is available to execute or not. If the user defines multiple service classes to give the work of the application different service goals, the caller can define that the minimum number of servers is spread across these service classes to ensure that servers are available for all work executed by the application.

The caller must have previously connected to WLM using the IWM4CON service specifying SERVER\_MANAGER=YES and SERVER\_TYPE=QUEUE. It is recommended to use the IWM4SLI service directly after IWM4CON. If any server uses this service to define limits, the limits apply for all servers of the application environment regardless of whether other servers use the service or not.

If a server defines new limits during execution, WLM attempts to meet the new limit definitions as soon as possible. If the maximum limit for servers is reduced during execution it is not predictable when WLM is able to meet the new maximum definition. This depends highly on the execution time of the running work requests. Therefore changing the limits during execution should be used very carefully and primarily during times of low application utilization.

### Information

Note that this service was previously called IWMSLIM for 31-bit address space only (see “IWMSLIM – Application Environment Limit Service” on page 864).

---

## Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Problem state. Any PSW key
<b>Dispatchable unit mode:</b>	Task
<b>Cross memory mode:</b>	PASN=HASN=SASN
<b>AMODE:</b>	31- or 64-bit
<b>ASC mode:</b>	Primary
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks may be held.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

---

## Programming Requirements

1. Make sure no EUT FRRs are established.
2. The macro CVT must be included to use this macro.
3. The macro IWMYCON must be included to use this macro.
4. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.

## IWM4SLI Macro

5. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, IWMRSNCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.

---

## Restrictions

1. This macro may not be used during task/address space termination.
2. Only a single invocation is allowed to be active for a given address space at any given time.
3. Before using this macro the caller must connect to WLM via IWM4CON Server\_Manager=YES, Server\_Type=Queue.
4. The macro must be used directly after using IWM4CON.

---

## Input Register Information

Before issuing the IWM4SLI macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

---

## Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Reason code if GR15 return code is non-zero
1	Used as work registers by the system
2-13	Unchanged
14	Used as work registers by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

---

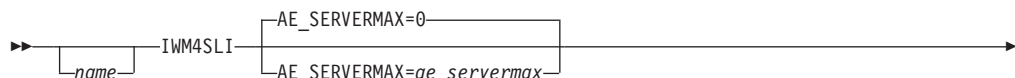
## Performance Implications

None.

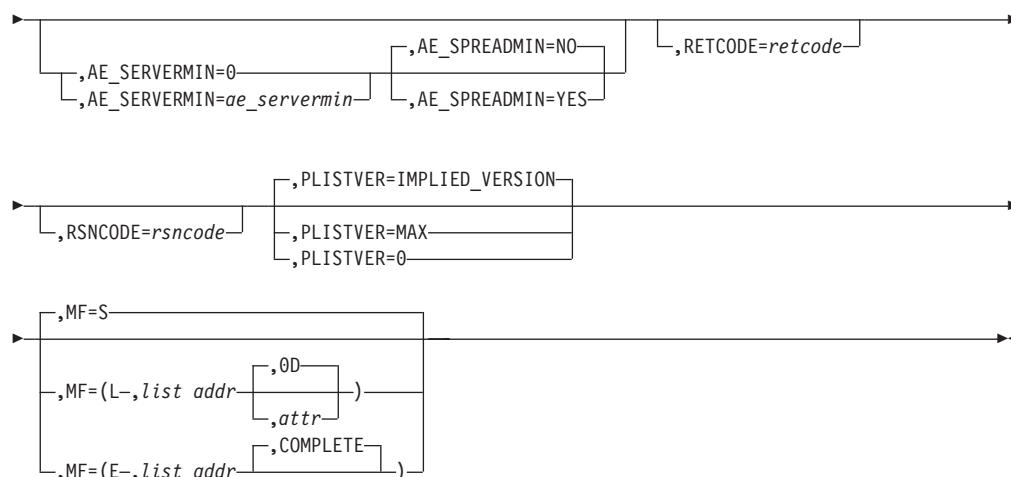
---

## Syntax

main diagram







## Parameters

The parameters are explained as follows:

### *name*

An optional symbol, starting in column 1, that is the name on the IWM4SLI macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

### **AE\_SERVERMAX=ae\_servermax**

#### **AE\_SERVERMAX=0**

An optional input parameter, which indicates the architectural limit for the total number of server instances which can run concurrently across the application environment for a given subsystem type and subsystem name.

This parameter represents a physical limit, such as the maximum number of available connections to a back-end subsystem. WLM will not start more than this number of server instances, even if goals cannot be met because of the limit. This value should be an integral multiple of the PARALLEL\_EU value defined on the IWM4CON service. If AE\_SERVERMAX is not an even multiple of PARALLEL\_EU, WLM will round this value down to the next integral multiple.

The maximum limit and the number of service classes to execute work requests should be defined carefully. If the number of service classes exceeds the quotient of AE\_SERVERMAX divided by PARALLEL\_EU WLM cannot start enough server address spaces to execute the work requests for all service classes. The default is 0, indicating that no maximum limit has been specified

**To code:** Specify the RS-type address of a halfword field.

### **,AE\_SERVERMIN=ae\_servermin**

#### **,AE\_SERVERMIN=0**

An optional input parameter, which indicates the minimum number of servers which should be up and running at all times.

This parameter can be used to tell WLM that a certain amount of server tasks should always be kept available to select work. This value should be an integral multiple of the PARALLEL\_EU value defined on IWM4CON service. If AE\_SERVERMIN is not an even multiple of PARALLEL\_EU, WLM will round this value down to the next integral multiple. The default is 0, indicating that no limit has been specified

**To code:** Specify the RS-type address of a halfword field.

**,AE\_SPREADADMIN=NO****,AE\_SPREADADMIN=YES**

When AE\_SERVERMIN=*ae\_servermin* is specified, an optional parameter, which indicates whether WLM will distribute the minimum number of servers as evenly as possible across the service classes being used to process the work requests. The default is AE\_SPREADADMIN=NO.

**,AE\_SPREADADMIN=NO**

The server tasks specified in AE\_SERVERMIN will be distributed to service classes as needed in order to meet goals.

**,AE\_SPREADADMIN=YES**

The server tasks specified in AE\_SERVERMIN will be distributed as evenly as possible to all service classes being used to execute work requests.

**,MF=S****,MF=(L,*list addr*)****,MF=(L,*list addr,attr*)****,MF=(L,*list addr,OD*)****,MF=(E,*list addr*)****,MF=(E,*list addr,COMPLETE*)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

**,*list addr***

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

**,*attr***

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

**,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

**,PLISTVER=IMPLIED\_VERSION****,PLISTVER=MAX****,PLISTVER=0**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form, when both are assembled with the same level of the system. In this way, MAX ensures that the parameter list does not overwrite nearby storage.

- **0**, if you use the currently available parameters.

**To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX
- A decimal value of 0

**,RETCODE=retcode**

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,RSNCODE=rsncode**

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

---

## ABEND Codes

None.

---

## Return and Reason Codes

When the IWM4SLI macro returns control to your program:

- GPR 15 (and *retcode*, when you code RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, when you code RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 82. Return and Reason Codes for the IWM4SLI Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> IwmRetCodeOk <b>Meaning:</b> Successful completion. <b>Action:</b> None required.
8	—	<b>Equate Symbol:</b> IwmRetCodeInvocError <b>Meaning:</b> Invalid invocation environment or parameters.

Table 82. Return and Reason Codes for the IWM4SLI Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0801	<b>Equate Symbol:</b> IwmRsnCodeSrbMode <b>Meaning:</b> Caller is in SRB mode. <b>Action:</b> Avoid requesting this function while in SRB mode.
8	xxxx0803	<b>Equate Symbol:</b> IwmRsnCodeDisabled <b>Meaning:</b> Caller is disabled. <b>Action:</b> Avoid requesting this function while disabled.
8	xxxx0804	<b>Equate Symbol:</b> IwmRsnCodeLocked <b>Meaning:</b> Caller is locked. <b>Action:</b> Avoid requesting this function while locked.
8	xxxx080B	<b>Equate Symbol:</b> IwmRsnCodeBadPI <b>Meaning:</b> Error accessing parameter list. <b>Action:</b> Check for possible storage overlay.
8	xxxx0824	<b>Equate Symbol:</b> IwmRsnCodeAmode24 <b>Meaning:</b> Caller invoked service but was in 24 bit addressing mode. <b>Action:</b> Request this function only when you are in 31 bit addressing mode.
8	xxxx0825	<b>Equate Symbol:</b> IwmRsnCodeAscModeNotPrimary <b>Meaning:</b> Caller invoked service but was not in primary ASC mode. <b>Action:</b> Avoid requesting this function in this environment.
8	xxxx0827	<b>Equate Symbol:</b> IwmRsnCodeRsvdNot0 <b>Meaning:</b> Reserved field in parameter list was not zero. <b>Action:</b> Check for use of keywords that are not supported by the MVS release on which the program is running.
8	xxxx0828	<b>Equate Symbol:</b> IwmRsnCodeBadVersion <b>Meaning:</b> Version number in parameter list is not valid. <b>Action:</b> Check for possible storage overlay of the parameter list.
8	xxxx0840	<b>Equate Symbol:</b> IwmRsnCodeServiceNotEnabled <b>Meaning:</b> Caller's space connection is not enabled for this service <b>Action:</b> Make sure that SERVER_MANAGER=YES and SERVER_TYPE=QUEUE is specified on the IWM4CON request to enable this service.
8	xxxx0841	<b>Equate Symbol:</b> IwmRsnCodeXmemMode <b>Meaning:</b> Caller is in cross-memory mode. <b>Action:</b> Request this function only when you are not in cross-memory mode.

Table 82. Return and Reason Codes for the IWM4SLI Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0842	<p><b>Equate Symbol:</b> IwmRsnCodeNoWLMConnect</p> <p><b>Meaning:</b> Caller's space is not connected to WLM.</p> <p><b>Action:</b> Invoke the IWM4CON macro before invoking this macro.</p>
8	xxxx087D	<p><b>Equate Symbol:</b> IWMRSNCODEBADNUMAESRVMAX</p> <p><b>Meaning:</b> The server maximum value is incorrect.</p> <p><b>Action:</b> Make sure that the maximum value is greater than the minimum value and that is greater than the parallel_eu value.</p>
10	—	<p><b>Equate Symbol:</b> IwmRetCodeCompError</p> <p><b>Meaning:</b> Component error.</p> <p><b>Action:</b> Contact your system programmer.</p>

---

## Example

To define application limits to WLM, specify the following:

```

IWM4CON WORK_MANAGER=YES,
         SERVER_MANAGER=YES,
         PARALLEL_EU=EUNITS,
         SERVER_TYPE=QUEUE,
         CONNTKN=CTKN,
         CONNTKNKEY=PSWKEY,
         RETCODE=RC,
         RSNCODE=RSN

IWM4SLI AE_SERVERMAX=MAXSRVS,
         AE_SERVERMIN=MINSRVS,
         RETCODE=RC,
         RSNCODE=RSN

*
* Storage areas
*
EUNITS  DS    F           Number of Tasks which will be started
*                               per address space.
MAXSRVS DS    H           Maximum Number of Servers supported
*                               by the application.
MINSRVS DS    H           Minimum number of servers which should
*                               be up and running all time
CTKN    DS    FL4         Connect Token
RC      DS    F           Return code
RSN     DS    F           Reason code

```



---

## Chapter 78. IWM4SRSC – Obtain Server-Specific Routing Information

The IWM4SRSC service provides information about how well a server is suitable to receive work from a WLM point-of-view. The IWM4SRSC service allows to check a specific server before routing work to it from WLM. Thus, the information obtained can be used for making balanced routing decisions, for example, with programs like Sysplex Distributor, and its exploiters, for example, TCP/IP.

The input to the IWM4SRSC service is the STOKEN of an address space. The output is an indicator, of how well the address space itself, the transactions or enclaves – if it is a registered transaction server, an enclave server or an enclave owner – are performing relative to their WLM goal and to the displaceable capacity for its WLM importance on that system.

The service returns an indicator that can be used for load balancing by comparing it to calls of this service for other servers.

The indicator output is a *weight*. This weight is calculated based on the combination of the following four factors:

- The first factor is how well this server, or the transactions or enclaves it is related to, fulfill their goals.
- The second factor is how much other work with lower importance can be displaced, if it receives more work to handle on this system.
- The third factor is the abnormal termination factor. This depends on the ratio of abnormal terminations to normal terminations as reported by the IWMRPT service. If no terminations were reported by IWMRPT, this factor is neutral (=1).
- The fourth factor is the health factor of this server. It is dependent on the health indicator which was reported to WLM for this server by the IWM4HLTH service or by IWMSRSG. If no health indicator was reported, this factor is also neutral.

These four factors are combined to create the output weight as a number.

A scenario where TCP/IP communicates on each system with WLM to obtain information about the servers which receive work is described in the following.

TCP/IP recognizes the server address spaces when they open a port. It invokes the IWM4SRSC service to WLM with an identification of the address space (STOKEN). WLM then finds out whether that address space is a registered WLM server address space or whether it creates WLM transactions which can execute in other server address spaces. The following possibilities are considered:

- The address space does not create any enclaves and is not a server address space registered to WLM. The FTP daemon is such an address space, for example. In this case WLM uses the service class the address space has been classified to.
- The address space is not a registered server address space but it creates independent enclaves which are processed by other address spaces on that system. In this case WLM has to use the service classes for the enclaves which are owned by this address space.

**Note:** There can be multiple service classes which are associated with enclaves. In this case the service class with the highest transaction rate is used.

## IWM4SRSC Macro

- The address space is a server address space from the point-of-view of WLM. In this case it is either an enclave or a transaction server, for example, CICS or IMS. In this case WLM uses the service classes the enclaves or CICS or IMS transactions are classified to.

---

## Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Problem state. Any PSW key
<b>Dispatchable unit mode:</b>	Task or SRB
<b>Cross memory mode:</b>	Any PASN, any HASN, any SASN
<b>AMODE:</b>	31- or 64-bit
<b>ASC mode:</b>	Primary
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks may be held. FRRs may be established.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

---

## Programming Requirements

1. The macro CVT must be included to use this macro.
2. The macro IWMYCON must be included to use this macro.
3. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
4. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, IWMRSNCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.
5. All character data, unless otherwise specified, is assumed to be left-justified and padded with blanks on the right, as needed, to occupy the specified number of bytes.

---

## Restrictions

- This macro may only be used on systems running z/OS V1R7 or higher.
- This macro supports multiple versions. Some keywords are only supported by certain versions. Refer to the PLISTVER parameter description for further information.

---

## Input Register Information

Before issuing the IWM4SRSC macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

---

## Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Reason code if GR15 return code is non-zero



1	Used as work registers by the system
2-13	Unchanged
14	Used as work registers by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

---

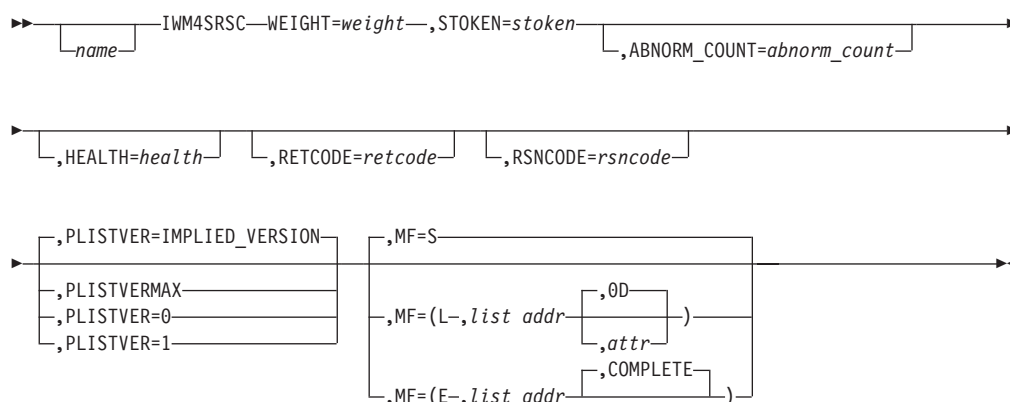
## Performance Implications

None.

---

## Syntax

### main diagram




---

## Parameters

The parameters are explained as follows:

### *name*

An optional symbol, starting in column 1, that is the name on the IWM4SRSC macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

### **,ABNORM\_COUNT=abnorm\_count**

An optional output parameter, which contains the number of abnormal terminations per 1000 total terminations, if available. If no abnormal terminations were reported to WLM, this value is zero. This is independent of the reason why no report was issued - whether no abnormal terminations occurred or whether the subsystem of the server is not enabled to report them to WLM.

## IWM4SRSC Macro

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

**,HEALTH=health**

An optional output parameter, which contains the health indicator of this server. This is a value between 0 and 100, which was reported to WLM either by the IWM4HLTH or the IWMSRSRG service. If no health indicator was reported, the value is 100.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

**,MF=S**

**,MF=(L,list addr)**

**,MF=(L,list addr,attr)**

**,MF=(L,list addr,0D)**

**,MF=(E,list addr)**

**,MF=(E,list addr,COMPLETE)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

**,list addr**

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

**,attr**

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

**,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

**,PLISTVER=IMPLIED\_VERSION**

**,PLISTVER=MAX**

**,PLISTVER=0**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.

- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify `PLISTVER=MAX` on the list form of the macro. Specifying `MAX` ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form, when both are assembled with the same level of the system. In this way, `MAX` ensures that the parameter list does not overwrite nearby storage.

- **0**, which supports all parameters except those specifically referenced in higher versions.
- **1**, which supports both the following parameters and those from version 0:

`ABNORM_COUNT`                      `HEALTH`

**To code:** Specify one of the following:

- `IMPLIED_VERSION`
- `MAX`
- A decimal value of 0, or 1

**,RETCODE=***retcode*

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,RSNCODE=***rsncode*

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,STOKEN=***stoken*

A required input parameter, which contains the space token of the server.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 64-bit field.

**WEIGHT=***weight*

A required output parameter, which contains the weight of how well the server is performing.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

---

## ABEND Codes

None.

---

## Return and Reason Codes

When the IWM4SRSC macro returns control to your program:

- GPR 15 (and *retcode*, when you code `RETCODE`) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, when you code `RSNCODE`) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 83. Return and Reason Codes for the IWM4SRSC Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> IwmRetCodeOk <b>Meaning:</b> Successful completion. <b>Action:</b> None required.
4	—	<b>Equate Symbol:</b> IwmRetCodeWarning <b>Meaning:</b> Successful completion, unusual conditions noted.
8	—	<b>Equate Symbol:</b> IwmRetCodeInvocError <b>Meaning:</b> Invalid invocation environment or parameters.
8	xxxx0803	<b>Equate Symbol:</b> IwmRsnCodeDisabled <b>Meaning:</b> Caller is disabled. <b>Action:</b> Avoid requesting this function while disabled.
8	xxxx0804	<b>Equate Symbol:</b> IwmRsnCodeLocked <b>Meaning:</b> Caller is locked. <b>Action:</b> Avoid requesting this function while locked.
8	xxxx0807	<b>Equate Symbol:</b> IwmRsnCodeBadSTOKEN <b>Meaning:</b> Bad STOKEN passed. <b>Action:</b> Check for possible storage overlay.
8	xxxx080B	<b>Equate Symbol:</b> IwmRsnCodeBadPI <b>Meaning:</b> Error accessing parameter list. <b>Action:</b> Check for possible storage overlay.
8	xxxx0823	<b>Equate Symbol:</b> IwmRsnCodeDatoff <b>Meaning:</b> Caller invoked service while DATOFF <b>Action:</b> Avoid requesting this function in this environment.
8	xxxx0824	<b>Equate Symbol:</b> IwmRsnCodeAmode24 <b>Meaning:</b> Caller invoked service but was in 24 bit addressing mode. <b>Action:</b> Request this function only when you are in 31 bit addressing mode.
8	xxxx0825	<b>Equate Symbol:</b> IwmRsnCodeAscModeNotPrimary <b>Meaning:</b> Caller invoked service but was not DAT on Primary ASC mode. <b>Action:</b> Avoid requesting this function in this environment.
8	xxxx0828	<b>Equate Symbol:</b> IwmRsnCodeBadVersion <b>Meaning:</b> Version number in parameter list or version length field is not valid. <b>Action:</b> Check for possible storage overlay of the parameter list.
C	—	<b>Equate Symbol:</b> IwmRetCodeEnvError <b>Meaning:</b> Environmental error.

Table 83. Return and Reason Codes for the IWM4SRSC Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
C	xxxx0C01	<b>Equate Symbol:</b> IwmRsnCodeNoStg <b>Meaning:</b> Storage is not available for the request. <b>Action:</b> There is a storage shortage. The function may work successfully at a later time.
10	—	<b>Equate Symbol:</b> IwmRetCodeCompError <b>Meaning:</b> Component error. <b>Action:</b> No action required. The function may be successful if invoked again.

## Example

To get the recommended values for a particular server, specify:

```
IWM4SRSC STOKEN=STKN,  
          WEIGHT=WGHT,  
          ABNORM_COUNT=ABCNT,  
          HEALTH=HLTH,  
          RETCODE=RC,RSNCODE=RSN  
  
*  
* Storage areas  
*  
STKN    DS    CL8           Contains the STOKEN  
*                               associated with the address  
*                               space  
WGHT    DS    F            Field to receive the weight  
ABCNT   DS    F            Field to receive the Abnormal Count  
HLTH    DS    F            Field to receive the Health Indicator  
RC      DS    F            Return code  
RSN     DS    F            Reason code
```

---

## Chapter 79. IWM4SSL – Selecting a Request from a Caller’s Work Manager Queue

The IWM4SSL service selects the next work request from the queue associated with the caller’s application environment. The caller must have previously connected to WLM using the IWM4CON service specifying `SERVER_MANAGER=YES`.

If there are no queued work requests waiting for selection the calling task will be suspended, pending arrival of work to do. The caller cannot rely upon asynchronous exits receiving control while the task is suspended.

After a work request is selected, the caller uses the IWM4STBG and IWM4STEN services to indicate the start and end of processing of the request.

### Information

Note that this service was previously called IWMSSSEL for 31-bit address space only (see “IWMSSSEL – Selecting a Request from a Caller’s Work Manager Queue” on page 871).

---

## Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Problem state. Any PSW key
<b>Dispatchable unit mode:</b>	Task
<b>Cross memory mode:</b>	PASN=HASN=SASN
<b>AMODE:</b>	31- or 64-bit
<b>ASC mode:</b>	Primary
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks may be held.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

---

## Programming Requirements

1. The macro CVT must be included to use this macro.
2. The macro IWMYCON must be included to use this macro.
3. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
4. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, `IWMRSNCODE_MASK_CONST` defined in IWMYCON, may be used for this purpose.

---

## Restrictions

None.

## Input Register Information

Before issuing the IWM4SSL macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

## Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Reason code if GR15 return code is non-zero
1	Used as work registers by the system
2-13	Unchanged
14	Used as work registers by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

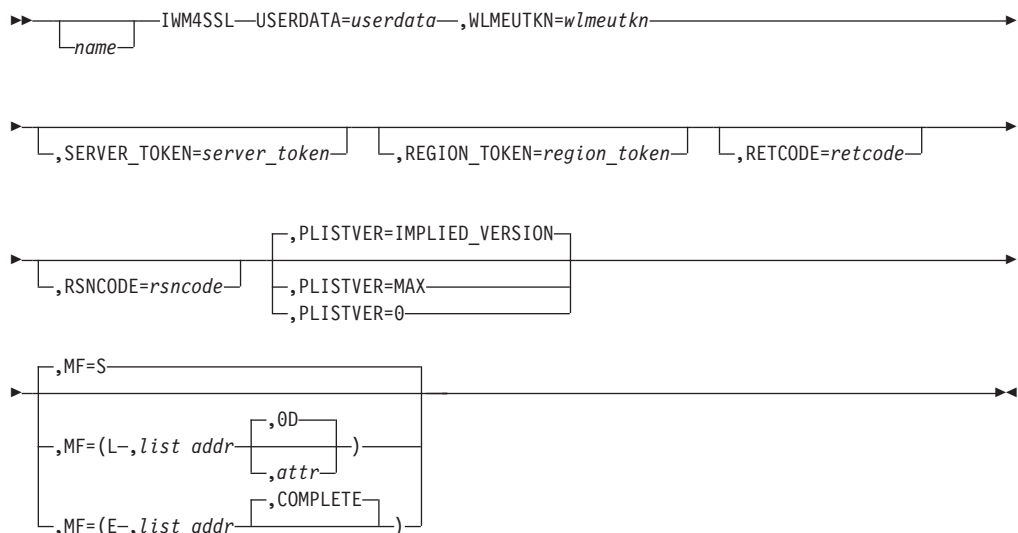
Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

## Performance Implications

None.

## Syntax

### main diagram





## Parameters

The parameters are explained as follows:

*name*

An optional symbol, starting in column 1, that is the name on the IWM4SSL macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

**,MF=S**

**,MF=(L,*list addr*)**

**,MF=(L,*list addr,attr*)**

**,MF=(L,*list addr,0D*)**

**,MF=(E,*list addr*)**

**,MF=(E,*list addr,COMPLETE*)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

*,list addr*

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

*,attr*

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

**,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

**,PLISTVER=IMPLIED\_VERSION**

**,PLISTVER=MAX**

**,PLISTVER=0**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.

- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify `PLISTVER=MAX` on the list form of the macro. Specifying `MAX` ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form, when both are assembled with the same level of the system. In this way, `MAX` ensures that the parameter list does not overwrite nearby storage.

- **0**, which supports all parameters except those specifically referenced in higher versions.

**To code:** Specify one of the following:

- `IMPLIED_VERSION`
- `MAX`
- A decimal value of 0

### **,REGION\_TOKEN=***region\_token*

An optional output parameter, which contains a region token. A queueing manager can use the region token to queue work requests to a specific server region. These work requests are considered to belong to a set of work requests all needing access to same status information which exists only in the virtual storage of the server region. They are selected using the `IWM4SSL` macro. It is assumed that the application uses the service `IWM4TAF` to tell WLM when the temporary affinity to the defined server region begins and ends.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

### **,RETCODE=***retcode*

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

### **,RSNCODE=***rsncode*

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a

### **,SERVER\_TOKEN=***server\_token*

An optional output parameter, which contains a server token. A queueing manager can use the server token to queue secondary work requests to this server task. Secondary work requests are considered to be extensions of the work request selected by `IWM4SSL`. They are selected using the `IWM4SSM` macro. See the `IWM4SSM` macro for more information.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32-character field.

### **USERDATA=***userdata*

A required output parameter, which contains the user data previously passed to WLM via `IWM4QIN`.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 16-character field.

### **,WLMEUTKN=***wlmeutkn*

A required output parameter, which will receive the execution unit token. This token must be passed on subsequent `IWM4STBG` and `IWM4STEN` requests.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

---

## ABEND Codes

None.

---

## Return and Reason Codes

When the IWM4SSL macro returns control to your program:

- GPR 15 (and *retcode*, when you code RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, when you code RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 84. Return and Reason Codes for the IWM4SSL Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> <code>IwmRetCodeOk</code> <b>Meaning:</b> Successful completion. <b>Action:</b> None required.
8	—	<b>Equate Symbol:</b> <code>IwmRetCodeInvocError</code> <b>Meaning:</b> Invalid invocation environment or parameters.
8	xxxx0801	<b>Equate Symbol:</b> <code>IwmRsnCodeSrbMode</code> <b>Meaning:</b> Caller is in SRB mode. <b>Action:</b> Avoid requesting this function while in SRB mode.
8	xxxx0803	<b>Equate Symbol:</b> <code>IwmRsnCodeDisabled</code> <b>Meaning:</b> Caller is disabled. <b>Action:</b> Avoid requesting this function while disabled.
8	xxxx0804	<b>Equate Symbol:</b> <code>IwmRsnCodeLocked</code> <b>Meaning:</b> Caller is locked. <b>Action:</b> Avoid requesting this function while locked.
8	xxxx080B	<b>Equate Symbol:</b> <code>IwmRsnCodeBadPI</code> <b>Meaning:</b> Error accessing parameter list. <b>Action:</b> Check for possible storage overlay.
8	xxxx0823	<b>Equate Symbol:</b> <code>IwmRsnCodeDatoff</code> <b>Meaning:</b> Caller invoked service while DATOFF <b>Action:</b> Avoid requesting this function in this environment.
8	xxxx0824	<b>Equate Symbol:</b> <code>IwmRsnCodeAmode24</code> <b>Meaning:</b> Caller invoked service but was in 24 bit addressing mode. <b>Action:</b> Request this function only when you are in 31 bit addressing mode.

Table 84. Return and Reason Codes for the IWM4SSL Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0825	<p><b>Equate Symbol:</b> IwmRsnCodeAscModeNotPrimary</p> <p><b>Meaning:</b> Caller invoked service but was not in primary ASC mode.</p> <p><b>Action:</b> Avoid requesting this function in this environment.</p>
8	xxxx0827	<p><b>Equate Symbol:</b> IwmRsnCodeRsvdNot0</p> <p><b>Meaning:</b> Reserved field in parameter list was non-zero.</p> <p><b>Action:</b> Check for possible storage overlay of the parameter list.</p>
8	xxxx0828	<p><b>Equate Symbol:</b> IwmRsnCodeBadVersion</p> <p><b>Meaning:</b> Version number in parameter list is not valid.</p> <p><b>Action:</b> Check for possible storage overlay of the parameter list.</p>
8	xxxx0840	<p><b>Equate Symbol:</b> IwmRsnCodeServiceNotEnabled</p> <p><b>Meaning:</b> Caller's space connection is not enabled for this service</p> <p><b>Action:</b> Make sure that SERVER_MANAGER=YES is specified on the IWM4CON request to enable this service.</p>
8	xxxx0841	<p><b>Equate Symbol:</b> IwmRsnCodeXmemMode</p> <p><b>Meaning:</b> Caller is in cross-memory mode.</p> <p><b>Action:</b> Request this function only when you are not in cross-memory mode.</p>
8	xxxx0842	<p><b>Equate Symbol:</b> IwmRsnCodeNoWLMConnect</p> <p><b>Meaning:</b> Caller's space is not connected to WLM.</p> <p><b>Action:</b> Invoke the IWM4CON macro before invoking this macro.</p>
8	xxxx0854	<p><b>Equate Symbol:</b> IwmRsnCodeTooManySelect</p> <p><b>Meaning:</b> The caller is attempting to select more work units than it has tasks to execute the work.</p> <p><b>Action:</b> Wait until an execution task has issued IWM4STEN before attempting to select more work units.</p>
8	xxxx0864	<p><b>Equate Symbol:</b> IwmRsnCodeSecondaryWorkExists</p> <p><b>Meaning:</b> There are secondary work requests queued to this server task. The caller was expected to process them using IWM4SSM before calling IWM4SSL.</p> <p><b>Action:</b> Select all secondary work requests before issuing IWM4SSL.</p>
C	—	<p><b>Equate Symbol:</b> IwmRetCodeEnvError</p> <p><b>Meaning:</b> Environmental error.</p>
C	xxxx0C01	<p><b>Equate Symbol:</b> IwmRsnCodeNoStg</p> <p><b>Meaning:</b> Storage is not available for the request.</p> <p><b>Action:</b> Caller must disconnect by invoking the IWM4DIS macro.</p>

Table 84. Return and Reason Codes for the IWM4SSL Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
C	xxxx0C14	<p><b>Equate Symbol:</b> IwmRsnCodeNoWorkShutDown</p> <p><b>Meaning:</b> No work selected. Caller is to shutdown.</p> <p><b>Action:</b> Caller must disconnect by invoking the IWM4DIS macro.</p>
C	xxxx0C3B	<p><b>Equate Symbol:</b> IwmRsnCodeStopTask</p> <p><b>Meaning:</b> WLM decided to stop the server instance.</p> <p><b>Action:</b> Calling task must shutdown, but server address space must remain active.</p>
10	—	<p><b>Equate Symbol:</b> IwmRetCodeCompError</p> <p><b>Meaning:</b> Component error.</p> <p><b>Action:</b> Contact your system programmer.</p>

---

## Example

To select a work request from the WLM queue manager queues, specify:

```

IWM4SSL USERDATA=USERDATA,          X
        WLMEUTKN=WLMEUTKN, RETCODE=RC, RSNCODE=RSN
*
* Storage areas
*
USERDATA DS    CL16          Contains the user-defined data
*                               that was passed to IWM4QIN
WLMEUTKN DS    CL8           Work unit token that must be
*                               passed to IWM4STBG and IWMSTEND
RC        DS    F            Return code
RSN       DS    F            Reason code

```

## IWM4SSL Macro

---

## Chapter 80. IWM4SSM – WLM Server Select Secondary Service

The IWM4SSM service selects the next secondary work request from the queue associated with the caller's server task.

If there are no queued secondary work requests waiting for selection the calling task will be suspended, pending arrival of work to do. The caller cannot rely upon asynchronous exits receiving control while the task is suspended.

Secondary work requests are considered to be extensions of an original work request selected using IWM4SSL. The caller must invoke WLM services in the following sequence:

- The caller invokes the IWM4SSL macro to select an initial work request. IWM4SSL returns a token identifying the server task. The caller is responsible for passing the server token to the queueing manager so that it can insert secondary work requests.
- The caller invokes the IWM4STBG macro to establish an environment for processing the work request selected using IWM4SSL. This environment also covers all secondary work requests.
- The caller invokes the IWM4SSM macro to select each secondary work request. The queueing manager is responsible for indicating the last secondary work request so that the server task knows when not to try to select another one.
- After the last secondary work request has been processed, the caller invokes the IWM4STEN macro to remove the environment created by IWM4STBG.
- The caller invokes IWM4SSL to select a new primary work request, and repeats the above flow.

In the above flow, IWM4SSL, IWM4STBG, IWM4SSM, and IWM4STEN must be invoked from the same task.

### Information

Note that this service was previously called IWMSSSEM for 31-bit address space only (see "IWMSSSEM – WLM Server Select Secondary Service" on page 877).

---

## Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Problem state. Any PSW key
<b>Dispatchable unit mode:</b>	Task
<b>Cross memory mode:</b>	PASN=HASN=SASN
<b>AMODE:</b>	31- or 64-bit
<b>ASC mode:</b>	Primary
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks may be held.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

---

## Programming Requirements

1. The macro CVT must be included to use this macro.
2. The macro IWMYCON must be included to use this macro.
3. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
4. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, IWMSRNCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.

---

## Restrictions

None.

---

## Input Register Information

Before issuing the IWM4SSM macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

---

## Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Reason code if GR15 return code is non-zero
1	Used as work registers by the system
2-13	Unchanged
14	Used as work registers by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

---

## Performance Implications

None.

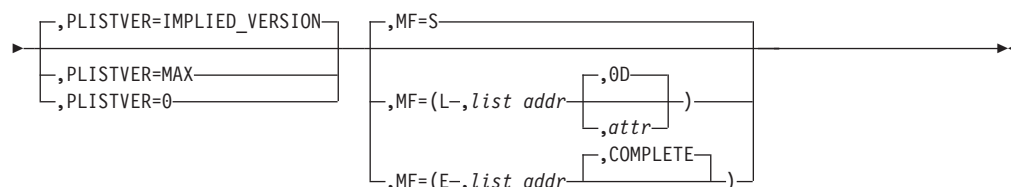
---

## Syntax

### main diagram

▶ `[name] IWM4SSM USERDATA=userdata [, RETCODE=retcode] [, RSNCODE=rsncode]` ▶





## Parameters

The parameters are explained as follows:

### *name*

An optional symbol, starting in column 1, that is the name on the IWM4SSM macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

### **,MF=S**

### **,MF=(L,*list addr*)**

### **,MF=(L,*list addr*,*attr*)**

### **,MF=(L,*list addr*,**OD**)**

### **,MF=(E,*list addr*)**

### **,MF=(E,*list addr*,**COMPLETE**)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

### *,list addr*

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

### *,attr*

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

### **,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

### **,PLISTVER=IMPLIED\_VERSION**

### **,PLISTVER=MAX**

### **,PLISTVER=0**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When

## IWM4SSM Macro

using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form, when both are assembled with the same level of the system. In this way, MAX ensures that the parameter list does not overwrite nearby storage.

- **0**, if you use the currently available parameters.

**To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX
- A decimal value of 0

**,RETCODE=retcode**

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,RSNCODE=rsncode**

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**USERDATA=userdata**

A required output parameter, which contains the user data previously passed to WLM via IWM4QIN.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 16-character field.

---

## ABEND Codes

None.

---

## Return and Reason Codes

When the IWM4SSM macro returns control to your program:

- GPR 15 (and *retcode*, when you code RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, when you code RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 85. Return and Reason Codes for the IWM4SSM Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> IwmRetCodeOk <b>Meaning:</b> Successful completion. <b>Action:</b> None required.
8	—	<b>Equate Symbol:</b> IwmRetCodeInvocError <b>Meaning:</b> Invalid invocation environment or parameters.
8	xxxx0801	<b>Equate Symbol:</b> IwmRsnCodeSrbMode <b>Meaning:</b> Caller is in SRB mode. <b>Action:</b> Avoid requesting this function while in SRB mode.
8	xxxx0803	<b>Equate Symbol:</b> IwmRsnCodeDisabled <b>Meaning:</b> Caller is disabled. <b>Action:</b> Avoid requesting this function while disabled.
8	xxxx0804	<b>Equate Symbol:</b> IwmRsnCodeLocked <b>Meaning:</b> Caller is locked. <b>Action:</b> Avoid requesting this function while locked.
8	xxxx080B	<b>Equate Symbol:</b> IwmRsnCodeBadPI <b>Meaning:</b> Error accessing parameter list. <b>Action:</b> Check for possible storage overlay.
8	xxxx0824	<b>Equate Symbol:</b> IwmRsnCodeAmode24 <b>Meaning:</b> Caller invoked service but was in 24 bit addressing mode. <b>Action:</b> Request this function only when you are in 31 bit addressing mode.
8	xxxx0825	<b>Equate Symbol:</b> IwmRsnCodeAscModeNotPrimary <b>Meaning:</b> Caller invoked service but was not in primary ASC mode. <b>Action:</b> Avoid requesting this function in this environment.
8	xxxx0827	<b>Equate Symbol:</b> IwmRsnCodeRsvdNot0 <b>Meaning:</b> Reserved field in parameter list was not zero. <b>Action:</b> Check for use of keywords that are not supported by the MVS release on which the program is running.
8	xxxx0828	<b>Equate Symbol:</b> IwmRsnCodeBadVersion <b>Meaning:</b> Version number in parameter list is not valid. <b>Action:</b> Check for possible storage overlay of the parameter list.
8	xxxx0840	<b>Equate Symbol:</b> IwmRsnCodeServiceNotEnabled <b>Meaning:</b> Caller's space connection is not enabled for this service <b>Action:</b> Make sure that SERVER_MANAGER=YES is specified on the IWM4CON request to enable this service.

## IWM4SSM Macro

Table 85. Return and Reason Codes for the IWM4SSM Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0841	<b>Equate Symbol:</b> IwmRsnCodeXmemMode <b>Meaning:</b> Caller is in cross-memory mode. <b>Action:</b> Request this function only when you are not in cross-memory mode.
8	xxxx0842	<b>Equate Symbol:</b> IwmRsnCodeNoWLMConnect <b>Meaning:</b> Caller's space is not connected to WLM. <b>Action:</b> Invoke the IWM4CON macro before invoking this macro.
8	xxxx0862	<b>Equate Symbol:</b> IwmRsnCodeNoPriorSelect <b>Meaning:</b> The caller has not previously selected work using the IWM4SSL macro. <b>Action:</b> Invoke the IWM4SSL macro before invoking this macro.
8	xxxx0863	<b>Equate Symbol:</b> IwmRsnCodeNoExecEnv <b>Meaning:</b> The caller has not established an execution environment using IWM4STBG. <b>Action:</b> Invoke the IWM4STBG macro before invoking this macro.
C	—	<b>Equate Symbol:</b> IwmRetCodeEnvError <b>Meaning:</b> Environmental error.
C	xxxx0C14	<b>Equate Symbol:</b> IwmRsnCodeNoWorkShutDown <b>Meaning:</b> No work selected. Caller is to shutdown. <b>Action:</b> Caller must disconnect by invoking the IW DISC macro.
10	—	<b>Equate Symbol:</b> IwmRetCodeCompError <b>Meaning:</b> Component error. <b>Action:</b> Contact your system programmer.

## Example

To select a secondary work request from the WLM queue manager queues, specify:

```

IWM4SSM USERDATA=USERDATA,RETCODE=RC,RSNCODE=RSN
*
* Storage areas
*
USERDATA DS    CL16           Contains the user-defined data
*                               that was passed to IWM4QIN
RC        DS    F             Return code
RSN       DS    F             Reason code

```

---

## Chapter 81. IWM4STBG – Beginning a Request from a Caller’s Work Manager Queue

IWM4STBG establishes the environment to process a work request that was previously selected using IWM4SSL. The caller must invoke IWM4STBG from the task in the server address space that will process the request. IWM4STBG establishes a business unit-of-work relationship by joining the caller’s task to the enclave associated with the request. IWM4STBG creates a security environment if there is a userid associated with the request previously selected.

Use IWM4STBG together with IWM4STEN to begin and end the processing of a work request. A task can process only one work request at a time.

Note that a task may only join an Enclave if it is not already part of an Enclave. In particular, a subtask which inherited the Enclave attribute from its mother task (which may happen either as a result of the mother task issuing IWMEJOIN or IWM4STBG) is not allowed to use IWMEJOIN to explicitly join an Enclave. This restriction is independent of whether the specified Enclave is the same Enclave as it is in, or a different Enclave from the one it is in. Such a subtask which inherited the Enclave attribute is also not allowed to use IWMELEAV to explicitly leave the Enclave. The subtask would only leave the Enclave upon its own (task) termination or when the Enclave is deleted (IWM4EDEL). Also, a task which successfully establishes a Begin environment (IWM4STBG) may not invoke Enclave Join, nor is the task allowed to use Enclave Leave while this Begin environment exists.

### Information

Note that this service was previously called IWMSTBGN for 31-bit address space only (see “IWMSTBGN – Beginning a Request from a Caller’s Work Manager Queue” on page 884).

---

## Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Problem state. Any PSW key
<b>Dispatchable unit mode:</b>	Task
<b>Cross memory mode:</b>	PASN=HASN=SASN
<b>AMODE:</b>	31- or 64-bit
<b>ASC mode:</b>	Primary
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks may be held.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

---

## Programming Requirements

1. The macro CVT must be included to use this macro.
2. The macro IWMYCON must be included to use this macro.
3. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.

## IWM4STBG Macro

- Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, `IWMRSNCODE_MASK_CONST` defined in `IWMYCON`, may be used for this purpose.

---

## Restrictions

The caller cannot have an EUT FRR established.

---

## Input Register Information

Before issuing the IWM4STBG macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

---

## Output Register Information

When control returns to the caller, the GPRs contain:

<b>Register</b>	<b>Contents</b>
<b>0</b>	Reason code if GR15 return code is non-zero
<b>1</b>	Used as work register by the system
<b>2-13</b>	Unchanged
<b>14</b>	Used as work registers by the system
<b>15</b>	Return code

When control returns to the caller, the ARs contain:

<b>Register</b>	<b>Contents</b>
<b>0-1</b>	Used as work registers by the system
<b>2-13</b>	Unchanged
<b>14-15</b>	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

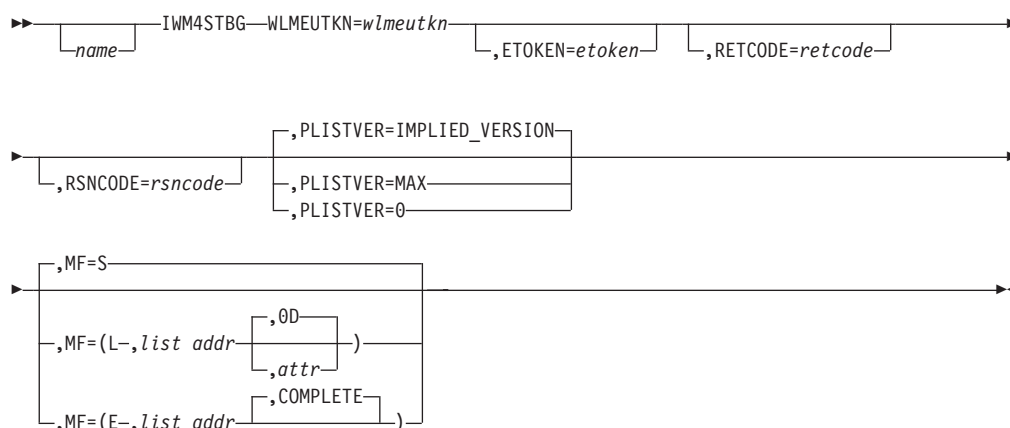
---

## Performance Implications

None.

## Syntax

### main diagram



## Parameters

The parameters are explained as follows:

### *name*

An optional symbol, starting in column 1, that is the name on the IWM4STBG macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

### **,ETOKEN=etoken**

An optional output parameter, which will receive the Enclave token.

**To code:** Specify the RS-type address, or register (2)-(12), of an 8-character field.

### **,MF=S**

#### **,MF=(L,list addr)**

#### **,MF=(L,list addr,attr)**

#### **,MF=(L,list addr,OD)**

#### **,MF=(E,list addr)**

#### **,MF=(E,list addr,COMPLETE)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

**,list *addr***

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

**,attr**

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

**,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

**,PLISTVER=IMPLIED\_VERSION**

**,PLISTVER=MAX**

**,PLISTVER=0**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form; in this way, MAX ensures that the parameter list does not overwrite nearby storage.

- **0**, if you use the currently available parameters.

**To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX
- A decimal value of 0

**,RETCODE=*retcode***

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,RSNCODE=*rsncode***

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**WLMEUTKN=*wlmeutkn***

A required input parameter, execution unit token that was returned by a prior invocation of IWM4SSL.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.



## ABEND Codes

None.

## Return and Reason Codes

When the IWM4STBG macro returns control to your program:

- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, if you coded RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 86. Return and Reason Codes for the IWM4STBG Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> <code>IwmRetCodeOk</code> <b>Meaning:</b> Successful completion. <b>Action:</b> None required.
4	—	<b>Equate Symbol:</b> <code>IwmRetCodeWarning</code> <b>Meaning:</b> Successful completion, unusual conditions noted.
4	xxxx041F	<b>Equate Symbol:</b> <code>IwmRsnCodeExecEnvChanged</code> <b>Meaning:</b> The execution environment has changed while the requested function is in progress. <b>Action:</b> None required.
8	—	<b>Equate Symbol:</b> <code>IwmRetCodeInvocError</code> <b>Meaning:</b> Invalid invocation environment or parameters.
8	xxxx0801	<b>Equate Symbol:</b> <code>IwmRsnCodeSrbMode</code> <b>Meaning:</b> Caller is in SRB mode. <b>Action:</b> Avoid requesting this function while in SRB mode.
8	xxxx0803	<b>Equate Symbol:</b> <code>IwmRsnCodeDisabled</code> <b>Meaning:</b> Caller is disabled. <b>Action:</b> Avoid requesting this function while disabled.
8	xxxx0804	<b>Equate Symbol:</b> <code>IwmRsnCodeLocked</code> <b>Meaning:</b> Caller is locked. <b>Action:</b> Avoid requesting this function while locked.
8	xxxx080B	<b>Equate Symbol:</b> <code>IwmRsnCodeBadPI</code> <b>Meaning:</b> Error accessing parameter list. <b>Action:</b> Check for possible storage overlay.
8	xxxx0810	<b>Equate Symbol:</b> <code>IwmRsnCodeEutFrr</code> <b>Meaning:</b> Caller has EUT FRR established. <b>Action:</b> Avoid requesting this function with an EUT FRR set.

## IWM4STBG Macro

Table 86. Return and Reason Codes for the IWM4STBG Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0823	<p><b>Equate Symbol:</b> IwmRsnCodeDatoff</p> <p><b>Meaning:</b> Caller invoked service while DATOFF</p> <p><b>Action:</b> Avoid requesting this function in this environment.</p>
8	xxxx0824	<p><b>Equate Symbol:</b> IwmRsnCodeAmode24</p> <p><b>Meaning:</b> Caller invoked service but was in 24 bit addressing mode.</p> <p><b>Action:</b> Request this function only when you are in 31 bit addressing mode.</p>
8	xxxx0825	<p><b>Equate Symbol:</b> IwmRsnCodeAscModeNotPrimary</p> <p><b>Meaning:</b> Caller invoked service but was not DAT on Primary ASC mode.</p> <p><b>Action:</b> Avoid requesting this function in this environment.</p>
8	xxxx0826	<p><b>Equate Symbol:</b> IwmRsnCodeTaskTerm</p> <p><b>Meaning:</b> Caller invoked service while task termination is in progress for the task associated with the owner.</p> <p><b>Action:</b> Avoid requesting this function while task termination is in progress.</p>
8	xxxx0828	<p><b>Equate Symbol:</b> IwmRsnCodeBadVersion</p> <p><b>Meaning:</b> Version number in parameter list is not valid or version length field is not valid.</p> <p><b>Action:</b> Check for possible storage overlay of the parameter list.</p>
8	xxxx083A	<p><b>Equate Symbol:</b> IwmRsnCodeBadEnclave</p> <p><b>Meaning:</b> Enclave token does not pass verification.</p> <p><b>Action:</b> Check for possible storage overlay of the Enclave token, or asynchronous events which may have deleted the Enclave.</p>
8	xxxx0840	<p><b>Equate Symbol:</b> IwmRsnCodeServiceNotEnabled</p> <p><b>Meaning:</b> Caller's space connection is not enabled for this service</p> <p><b>Action:</b> Make sure that SERVER_MANAGER=YES is specified on the IWM4CON request to enable this service.</p>
8	xxxx0841	<p><b>Equate Symbol:</b> IwmRsnCodeXmemMode</p> <p><b>Meaning:</b> Caller is in cross-memory mode.</p> <p><b>Action:</b> Request this function only when you are not in cross-memory mode.</p>
8	xxxx0842	<p><b>Equate Symbol:</b> IwmRsnCodeNoWLMConnect</p> <p><b>Meaning:</b> Caller's space is not connected to WLM.</p> <p><b>Action:</b> Invoke the IWM4CON macro before invoking this macro.</p>

Table 86. Return and Reason Codes for the IWM4STBG Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0850	<b>Equate Symbol:</b> IwmRsnCodeBeginEnvOutstanding <b>Meaning:</b> Caller is already operating under an outstanding Begin environment. <b>Action:</b> Avoid requesting this function in this environment.
8	xxxx0851	<b>Equate Symbol:</b> IwmRsnCodeSecEnvOutstanding <b>Meaning:</b> Caller is already operating under an outstanding security environment. <b>Action:</b> Avoid requesting this function while there is a task level security environment outstanding.
8	xxxx0852	<b>Equate Symbol:</b> IwmRsnCodeExecTokenNotCorrect <b>Meaning:</b> The execution unit token does not identify a previously selected work unit. <b>Action:</b> Verify that you have coded the WLMEUTKN parameter correctly.
8	xxxx0857	<b>Equate Symbol:</b> IwmRsnCodeAlreadyInEnclave <b>Meaning:</b> Current dispatchable workunit is already in an Enclave. <b>Action:</b> Avoid requesting this function while the caller is already in an Enclave.
8	xxxx085A	<b>Equate Symbol:</b> IwmRsnCodeSelectedWorkActive <b>Meaning:</b> The selected work element associated with the input execution unit token is already in execution. <b>Action:</b> You may have invoked IWM4STBG from multiple tasks in the server address space passing the same WLMEUTKN. Avoid requesting this function in this environment.
C	—	<b>Equate Symbol:</b> IwmRetCodeEnvError <b>Meaning:</b> Environmental error.
C	xxxx0C17	<b>Equate Symbol:</b> IwmRsnCodeSecEnvCreateFailed <b>Meaning:</b> A user security environment cannot be created. <b>Action:</b> Verify that the userid is defined to RACF or check the SAF installation exit routine to enable the function.
10	—	<b>Equate Symbol:</b> IwmRetCodeCompError <b>Meaning:</b> Component error. <b>Action:</b> Contact your system programmer.

---

## Example

Suppose a work request was selected using IWM4SSL and the execution unit token returned by IWM4SSL is WLMEUTKN.

To establish the environment to process the work request, specify:

```
IWM4STBG WLMEUTKN=WLMEUTKN,RETCODE=RC,RSNCODE=RSN
```

\*

\* Storage areas

## IWM4STBG Macro

*				
WLMEUTKN	DS	CL8		Contains the execution unit token that was returned by IWM4SSL
*				
*				
RC	DS	F		Return code
RSN	DS	F		Reason code

---

## Chapter 82. IWM4STEN – End a Request from a Caller’s Work Manager Queue

IWM4STEN removes the environment which was previously established using IWM4STBG to process a work request. The caller must invoke IWM4STEN from the same task that invoked IWM4STBG. IWM4STEN removes the caller’s task from the enclave associated with the request. IWM4STEN deletes the security environment if one was previously established by IWM4STBG.

### Information

Note that this service was previously called IWMSTEND for 31-bit address space only (see “IWMSTEND – End a Request from a Caller’s Work Manager Queue” on page 891).

---

## Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Problem state. Any PSW key
<b>Dispatchable unit mode:</b>	Task
<b>Cross memory mode:</b>	PASN=HASN=SASN
<b>AMODE:</b>	31- and 64-bit
<b>ASC mode:</b>	Primary
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks may be held.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

---

## Programming Requirements

1. The macro CVT must be included to use this macro.
2. The macro IWMYCON must be included to use this macro.
3. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
4. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, IWMRSNCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.

---

## Restrictions

The caller cannot have an EUT FRR established.

---

## Input Register Information

Before issuing the IWM4STEN macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

## Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Reason code if GR15 return code is non-zero
1	Used as work register by the system
2-13	Unchanged
14	Used as work registers by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

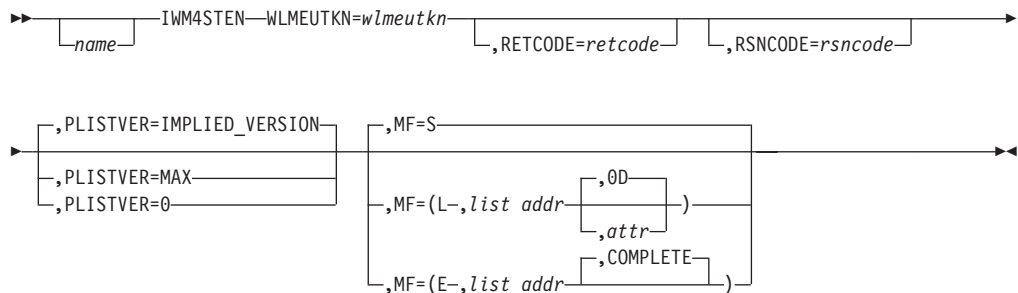
Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

## Performance Implications

None.

## Syntax

### main diagram



## Parameters

The parameters are explained as follows:

### *name*

An optional symbol, starting in column 1, that is the name on the IWM4STEN macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

### **,MF=S**

**,MF=(L,list addr)**

**,MF=(L,list addr,attr)**

**,MF=(L,list addr,0D)**

**,MF=(E,list addr)**

**,MF=(E,*list addr*,COMPLETE)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

**,*list addr***

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

**,*attr***

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

**,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

**,PLISTVER=IMPLIED\_VERSION****,PLISTVER=MAX****,PLISTVER=0**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form; in this way, MAX ensures that the parameter list does not overwrite nearby storage.

- **0**, if you use the currently available parameters.

**To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX
- A decimal value of 0

## IWM4STEN Macro

### **,RETCODE=***retcode*

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

### **,RSNCODE=***rsncode*

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

### **WLMEUTKN=***wlmeutkn*

A required input parameter, execution unit token that was specified on the prior invocation of IWM4STBG.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

---

## ABEND Codes

None.

---

## Return and Reason Codes

When the IWM4STEN macro returns control to your program:

- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, if you coded RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 87. Return and Reason Codes for the IWM4STEN Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> IwmRetCodeOk <b>Meaning:</b> Successful completion. <b>Action:</b> None required.
4	—	<b>Equate Symbol:</b> IwmRetCodeWarning <b>Meaning:</b> Successful completion, unusual conditions noted.
4	xxxx041C	<b>Equate Symbol:</b> IwmRsnCodeNotEnclave <b>Meaning:</b> The current dispatchable workunit is not associated with an Enclave. <b>Action:</b> None required.
4	xxxx041F	<b>Equate Symbol:</b> IwmRsnCodeExecEnvChanged <b>Meaning:</b> The execution environment has changed while the requested function is in progress. <b>Action:</b> None required.



Table 87. Return and Reason Codes for the IWM4STEN Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
4	xxxx042F	<p><b>Equate Symbol:</b> IwmRsnCodeSecondaryWorkDeleted</p> <p><b>Meaning:</b> There were secondary work requests queued to this server task. The caller was expected to process them using IWM4SSM before calling IWM4STEN. The secondary work requests were deleted.</p> <p><b>Action:</b> Select all secondary work requests before issuing IWM4STEN.</p>
8	—	<p><b>Equate Symbol:</b> IwmRetCodeInvocError</p> <p><b>Meaning:</b> Invalid invocation environment or parameters.</p>
8	xxxx0801	<p><b>Equate Symbol:</b> IwmRsnCodeSrbMode</p> <p><b>Meaning:</b> Caller is in SRB mode.</p> <p><b>Action:</b> Avoid requesting this function while in SRB mode.</p>
8	xxxx0803	<p><b>Equate Symbol:</b> IwmRsnCodeDisabled</p> <p><b>Meaning:</b> Caller is disabled.</p> <p><b>Action:</b> Avoid requesting this function while disabled.</p>
8	xxxx0804	<p><b>Equate Symbol:</b> IwmRsnCodeLocked</p> <p><b>Meaning:</b> Caller is locked.</p> <p><b>Action:</b> Avoid requesting this function while locked.</p>
8	xxxx080B	<p><b>Equate Symbol:</b> IwmRsnCodeBadPI</p> <p><b>Meaning:</b> Error accessing parameter list.</p> <p><b>Action:</b> Check for possible storage overlay.</p>
8	xxxx0810	<p><b>Equate Symbol:</b> IwmRsnCodeEutFrr</p> <p><b>Meaning:</b> Caller has EUT FRR established.</p> <p><b>Action:</b> Avoid requesting this function with an EUT FRR set.</p>
8	xxxx0823	<p><b>Equate Symbol:</b> IwmRsnCodeDatoff</p> <p><b>Meaning:</b> Caller invoked service while DATOFF</p> <p><b>Action:</b> Avoid requesting this function in this environment.</p>
8	xxxx0824	<p><b>Equate Symbol:</b> IwmRsnCodeAmode24</p> <p><b>Meaning:</b> Caller invoked service but was in 24 bit addressing mode.</p> <p><b>Action:</b> Request this function only when you are in 31 bit addressing mode.</p>
8	xxxx0825	<p><b>Equate Symbol:</b> IwmRsnCodeAscModeNotPrimary</p> <p><b>Meaning:</b> Caller invoked service but was not DAT on Primary ASC mode.</p> <p><b>Action:</b> Avoid requesting this function in this environment.</p>
8	xxxx0828	<p><b>Equate Symbol:</b> IwmRsnCodeBadVersion</p> <p><b>Meaning:</b> Version number in parameter list is not valid or version length field is not valid.</p> <p><b>Action:</b> Check for possible storage overlay of the parameter list.</p>

## IWM4STEN Macro

Table 87. Return and Reason Codes for the IWM4STEN Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0841	<p><b>Equate Symbol:</b> IwmRsnCodeXmemMode</p> <p><b>Meaning:</b> Caller invoked service but was in cross-memory mode.</p> <p><b>Action:</b> Avoid requesting this function in cross-memory mode.</p>
8	xxxx084F	<p><b>Equate Symbol:</b> IwmRsnCodeWrongExecToken</p> <p><b>Meaning:</b> Current dispatchable workunit is not associated with the input execution unit token.</p> <p><b>Action:</b> Check for possible storage overlay of the execution unit token.</p>
8	xxxx0859	<p><b>Equate Symbol:</b> IwmRsnCodeEnclaveSubTaskExists</p> <p><b>Meaning:</b> The current dispatchable workunit has residual subtasks propagated to the Enclave which are still associated with the Enclave.</p> <p><b>Action:</b> Avoid requesting this function in this environment.</p>
10	—	<p><b>Equate Symbol:</b> IwmRetCodeCompError</p> <p><b>Meaning:</b> Component error.</p> <p><b>Action:</b> Contact your system programmer.</p>

## Example

To remove the environment which was previously established using IWM4STBG, specify:

```

        IWM4STEN WLMEUTKN=WLMEUTKN,RETCODE=RC,RSNCODE=RSN
*
* Storage areas
*
WLMEUTKN DS    CL8           Contains the execution unit
*                               token that was specified on
*                               the prior invocation of
*                               IWM4STBG
RC        DS    F            Return code
RSN       DS    F            Reason code

```

---

## Chapter 83. IWM4TAF — WLM Temporal Affinity Service

The IWM4TAF service should be used to inform WLM when a temporal affinity for a specific server region starts and when it ends. WLM will ensure that server regions will not be terminated as long as temporal affinities exist.

The caller must have previously connected to WLM using the IWM4CON as server or as queue manager.

### Information

Note that this service was previously called IWMTAFF for 31-bit address space only (see “IWMTAFF — WLM Temporal Affinity Service” on page 897).

---

## Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Problem state. Any PSW key
<b>Dispatchable unit mode:</b>	Task
<b>Cross memory mode:</b>	PASN=HASN=SASN
<b>AMODE:</b>	31- and 64-bit
<b>ASC mode:</b>	Primary
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks may be held.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

---

## Programming Requirements

1. Make sure no EUT FRRs are established.
2. The macro CVT must be included to use this macro.
3. The macro IWMYCON must be included to use this macro.
4. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
5. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, IWMSRNCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.

---

## Restrictions

- This macro may not be used during task/address space termination.
- Before using this macro the caller must connect to WLM via IWM4CON Server\_Manager=YES, Server\_Type=Queue or IWM4CON Queue\_Manager=YES.

## Input Register Information

Before issuing the IWM4TAF macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

## Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Reason code if GR15 return code is non-zero
1	Used as work registers by the system
2-13	Unchanged
14	Used as work registers by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

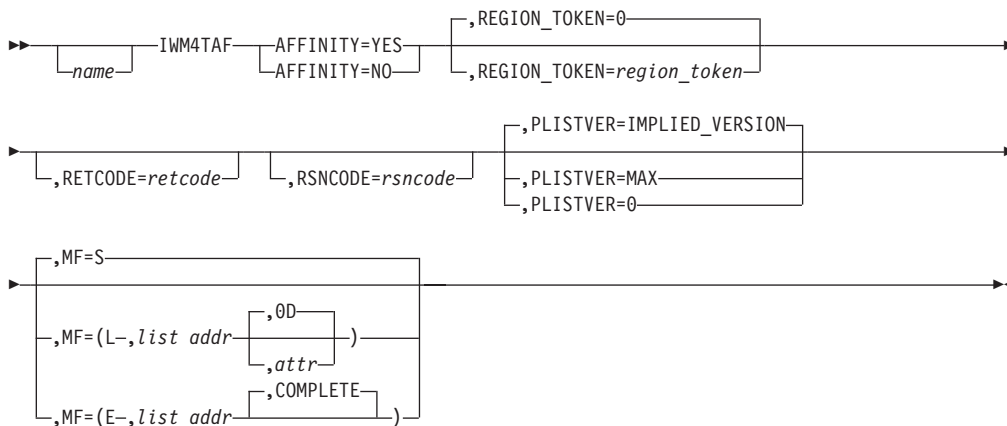
Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

## Performance Implications

None.

## Syntax

### main diagram



## Parameters

The parameters are explained as follows:

*name*

An optional symbol, starting in column 1, that is the name on the IWM4TAF macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

**AFFINITY=YES**

**AFFINITY=NO**

A required parameter indicating whether a temporal affinity begins or ends

**AFFINITY=YES**

A new temporal affinity for the server region begins. WLM will ensure that the server regions is not terminated before all temporal affinity have ended.

**AFFINITY=NO**

A temporal affinity for the server region has ended. WLM will start to terminate server regions if all temporal affinities have ended.

**,MF=S**

**,MF=(L,*list addr*)**

**,MF=(L,*list addr,attr*)**

**,MF=(L,*list addr,OD*)**

**,MF=(E,*list addr*)**

**,MF=(E,*list addr,COMPLETE*)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

*,list addr*

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

*,attr*

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

**,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

**,PLISTVER=IMPLIED\_VERSION**

**,PLISTVER=MAX**

**,PLISTVER=0**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When

## IWM4TAF Macro

using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form, when both are assembled with the same level of the system. In this way, MAX ensures that the parameter list does not overwrite nearby storage.

- **0**, if you use the currently available parameters.

**To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX
- A decimal value of 0

**,REGION\_TOKEN=region\_token**

**,REGION\_TOKEN=0**

An optional input parameter, which contains the region token. The region token is not required if the macro is invoked from the server region for which the temporal affinity should be started or stopped. The region token must be used if the services is used from the queueing manager. The region token is returned by the IWM4SSL macro.

The caller must be supervisor state or have PSW key mask 0-7 authority to use this service with the REGION\_TOKEN parameter.

Coding REGION\_TOKEN=0 is equivalent to omitting the REGION\_TOKEN keyword. The default is 0.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 16-character field.

**,RETCODE=retcode**

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,RSNCODE=rsncode**

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

---

## ABEND Codes

None.

---

## Return and Reason Codes

When the IWM4TAF macro returns control to your program:

- GPR 15 (and *retcode*, when you code RETCODE) contains a return code.

- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, when you code RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 88. Return and Reason Codes for the IWM4TAF Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> <code>IwmRetCodeOk</code> <b>Meaning:</b> Successful completion. <b>Action:</b> None required.
4	—	<b>Equate Symbol:</b> <code>IwmRetCodeWarning</code> <b>Meaning:</b> Successful completion, unusual conditions noted. <b>Action:</b> None required.
4	xxxx0439	<b>Equate Symbol:</b> <code>IwmRsnCodeNoAffinityFound</code> <b>Meaning:</b> The service has been invoked to tell WLM that an existing server region affinity has been terminated but WLM has no affinity defined for this server region. <b>Action:</b> If region token was not specified make sure to use the service properly at the beginning and end of each affinity. If the region token has been defined make sure that it is used for the correct server region.
4	xxxx043A	<b>Equate Symbol:</b> <code>IwmRsnCodeRegionNotFound</code> <b>Meaning:</b> The region token does not identify a valid server region. <b>Action:</b> Please specify the correct region token.
8	—	<b>Equate Symbol:</b> <code>IwmRetCodeInvocError</code> <b>Meaning:</b> Invalid invocation environment or parameters.
8	xxxx0801	<b>Equate Symbol:</b> <code>IwmRsnCodeSrbMode</code> <b>Meaning:</b> Caller is in SRB mode. <b>Action:</b> Avoid requesting this function while in SRB mode.
8	xxxx0803	<b>Equate Symbol:</b> <code>IwmRsnCodeDisabled</code> <b>Meaning:</b> Caller is disabled. <b>Action:</b> Avoid requesting this function while disabled.
8	xxxx0804	<b>Equate Symbol:</b> <code>IwmRsnCodeLocked</code> <b>Meaning:</b> Caller is locked. <b>Action:</b> Avoid requesting this function while locked.
8	xxxx080B	<b>Equate Symbol:</b> <code>IwmRsnCodeBadPI</code> <b>Meaning:</b> Error accessing parameter list. <b>Action:</b> Check for possible storage overlay.

Table 88. Return and Reason Codes for the IWM4TAF Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0824	<p><b>Equate Symbol:</b> IwmRsnCodeAmode24</p> <p><b>Meaning:</b> Caller invoked service but was in 24 bit addressing mode.</p> <p><b>Action:</b> Request this function only when you are in 31 bit addressing mode.</p>
8	xxxx0825	<p><b>Equate Symbol:</b> IwmRsnCodeAscModeNotPrimary</p> <p><b>Meaning:</b> Caller invoked service but was not in primary ASC mode.</p> <p><b>Action:</b> Avoid requesting this function in this environment.</p>
8	xxxx0827	<p><b>Equate Symbol:</b> IwmRsnCodeRsvdNot0</p> <p><b>Meaning:</b> Reserved field in parameter list was not zero.</p> <p><b>Action:</b> Check for use of keywords that are not supported by the MVS release on which the program is running.</p>
8	xxxx0828	<p><b>Equate Symbol:</b> IwmRsnCodeBadVersion</p> <p><b>Meaning:</b> Version number in parameter list is not valid.</p> <p><b>Action:</b> Check for possible storage overlay of the parameter list.</p>
8	xxxx0840	<p><b>Equate Symbol:</b> IwmRsnCodeServiceNotEnabled</p> <p><b>Meaning:</b> Caller's space connection is not enabled for this service</p> <p><b>Action:</b> Make sure that SERVER_MANAGER=YES and SERVER_TYPE=QUEUE is specified on the IWM4CON request to enable this service.</p>
8	xxxx0841	<p><b>Equate Symbol:</b> IwmRsnCodeXmemMode</p> <p><b>Meaning:</b> Caller is in cross-memory mode.</p> <p><b>Action:</b> Request this function only when you are not in cross-memory mode.</p>
8	xxxx0842	<p><b>Equate Symbol:</b> IwmRsnCodeNoWLMConnect</p> <p><b>Meaning:</b> Caller's space is not connected to WLM.</p> <p><b>Action:</b> Invoke the IWM4CON macro before invoking this macro.</p>
8	xxxx084D	<p><b>Equate Symbol:</b> IwmRsnCodeNotAuthConnect</p> <p><b>Meaning:</b> The caller must be supervisor state or have PSW key mask 0-7 authority to use the requested WLM service. This applies only if the caller provides a region token for a server address space for which it wants to set the affinity.</p> <p><b>Action:</b> Avoid requesting this function in this environment.</p>
10	—	<p><b>Equate Symbol:</b> IwmRetCodeCompError</p> <p><b>Meaning:</b> Component error.</p> <p><b>Action:</b> Contact your system programmer.</p>



---

## Example

To start a temporal affinity from the server region, specify:

```
IWM4TAF AFFINITY=YES
        RETCODE=RC,
        RSNCODE=RSN
*
* Storage areas
*
RC      DS    F          Return code
RSN     DS    F          Reason code
```



---

## Part 3. Appendixes



## Appendix A. SMF Type 99 Action Codes

Table 89. SMF Type 99 Action Codes

1	STA_RECOVERY_RETRY	Retry.
2	STA_RECOVERY_PERC	Percolation.
10	RA_AUXP_DEC_MPL	Resource adjustment, too much auxiliary storage paging, decrease mpl.
3	STA_RECOVERY_REDRIIVE_SET	Tell WLM to set to same policy again.
20	RA_AUXP_NO_ACTION	Resource adjustment, too much auxiliary storage paging, no action.
30	RA_MP_NO_ACTION	Resource adjustment, managed paging, no action.
40	RA_OU_DEC_MPL	Resource adjustment, overutilized, decrease mpl.
50	RA_OU_NO_ACTION	Resource adjustment, overutilized, no action.
60	RA_SWAP_FOR_MPL	Resource adjustment, working set management picked this address space to swap out.
70	RA_UP_DECREASE_MPL	Resource adjustment, unmanaged paging decrease mpl.
90	RA_UP_NO_ACTION	Resource adjustment, unmanaged paging no action.
100	RA_UU_INC_MPL	Resource adjustment, underutilized, increase mpl.
105	RA_UU_ADD_SRV_GR	Resource adjustment, underutilized, add server.
106	RA_UU_ADD_SRV_RR	Resource adjustment, underutilized, add server.
107	ADD_SRV_ASSESS	Resource adjustment or discretionary goal management, assess adding server(s).
108	ADD_SRV_ASSESS2	Resource adjustment or discretionary goal management, assess adding server(s).
110	RA_UU_NO_ACTION	Resource adjustment, underutilized, no action.
130	SWAP_DETECTED_WAIT	Detected wait swap.
140	SWAP_EXCHANGE	Exchange swap.
150	SWAP_LONG_WAIT	Long wait swap.
160	SWAP_UNILATERAL	Unilateral swap.

Table 89. SMF Type 99 Action Codes (continued)

170	RA_MON_PAG_COST_HI	Resource adjustment, monitor this space because paging cost is high.
180	RA_MON_POLICY_DIR	Resource adjustment, policy code directed us to monitor this space.
190	RA_UNMON_ALL_P_OK	Resource adjustment, unmonitor this space because the last 10 plotted points were ok.
195	RA_UNMON_NO_CAPT	Resource adjustment, unmonitor this space because insufficient capture time accumulated in last 5 minutes to plot a point.
200	TX_END_UNMON	Unmonitor because of transaction end, initiator detach, or address space termination.
210	NS_STOR_TAR_ACTION	Storage target action, no specific reason.
220	PA_ADD_TRANS_DISP	Add transaction server dynamic internal service period.
222	PA_AS_BET_DISPS	Move address space between server internal service periods.
224	PA_AS_FROM_DISP	Move address space from server internal service period.
226	PA_AS2_TRX_DISP	Move address space to server internal service period.
227	PA_AS2_NONTRX_DISP	Move address space to server internal service period.
230	PA_DELETE_DISP	Delete server internal service period.
232	PA_ADDDISP_MT_EN_Q	Add a non transaction server dynamic internal service period for multi-threaded enclave queue servers.
233	PA_ADD_DISP_MT_EN	Add a non transaction server dynamic internal service period for multi-threaded enclave non_queue servers.
235	PA_ADDDISP_ST_EN_Q	Add a non transaction server dynamic internal service period for single-threaded enclave queue servers.
236	PA_ADD_DISP_ST_EN	Add a non transaction server dynamic internal service period for single-threaded enclave non-queue servers.

Table 89. SMF Type 99 Action Codes (continued)

240	PA_GREC_CAND	Policy adjustment, goal receiver candidate selected.
245	PA_NA_NO_MPL	Policy adjustment, no action taken because period had an mpl out target of 0 and the delay was other than mpl.
246	PA_DRV_PRO_SKIPPED	Policy adjustment, processor action was skipped even though it is required.
250	PA_NA_NO_PROBLEM	Policy adjustment, no action taken because receiver did not have a problem.
251	PA_ADDDISP_SCSP	Add a non-transaction server dynamic internal service period for a single class/single period enclave server.
252	PA_ADDDISP_SCSP_Q	Add a non transaction server dynamic internal service period for a single class/single period enclave and queue server.
253	PA_ADDDISP_SCMP	Add a non transaction server dynamic internal service period for a single class/multi-period enclave server.
254	PA_ADDDISP_SCMP_Q	Add a non transaction server dynamic internal service period for a single class/multi-period enclave and queue server.
255	PA_ADDDISP_MCMP	Add a non transaction server dynamic internal service period for a multi-class/multi-period enclave server.
256	PA_ADDDISP_MCMP_Q	Add a non transaction server dynamic internal service period for a multi-class/multi-period enclave and queue server.
260	PA_NA_UNKNOW_DELAY	Policy adjustment, no action taken because delay is not known.
265	PA_NA_SYSPLEX_ONLY	Policy adjustment, because resource only addressed on sysplex pass and this is the local pass.
270	PA_REC_CAND	Policy adjustment, receiver candidate selected.
280	PA_RREC_CAND	Policy adjustment, resource receiver candidate selected.

Table 89. SMF Type 99 Action Codes (continued)

290	PA_USE_DISC_CENT	Policy adjustment, use discretionary central.
300	PA_USE_DISC_EXP	Policy adjustment, use discretionary expanded.
305	PA_STOR_DONOR	Policy adjustment, storage donor.
306	SH_STOR_DONOR	Shortage, storage donor.
307	SV_STOR_DONOR	Storage donor from server.
308	PA_DONOR_PERIOD	Policy adjustment, donor period.
310	WLM_Q_REQ	WLM queue sysevent issued. Begin/end are not traced.
311	WLM_Q_MISC	WLM queue miscellaneous actions.
315	PA_CPC_MOVE_DOWN	Policy adjustment, period is moved down to make room for CPU critical period.
320	PA_CAL_PI_NO_FOREIGN_FACTOR	Policy adjustment, calculate PI.
500	HSK_FROM_SPC_DP	Housekeeping, move from small processor consumer priority, period is no longer small consumer.
510	HSK_TO_SPC_DP	Housekeeping, move to small processor consumer priority.
520	HSK_XFROM_SPC_DP	Housekeeping, exchange from small processor consumer priority to make room for another small consumer.
525	HSK_UNBUNCH_PRTY	Housekeeping, unbunch priorities.
526	PA_PCC_NO_OCC_PRTY	Policy adjustment, CPU critical, no occupied priority to move blocker to.
527	PA_PCC_NO_UNO_PRTY	Policy adjustment, CPU critical, no unoccupied priority to move blocker to.
528	PA_PCC_BLK_R_MOVED	Policy adjustment, CPU critical, blocker has been moved to a new priority.
529	PA_PCC_BLK_R_VIOLTN	Policy adjustment, CPU critical, blocker violates CPU critical rules at new priority.
530	PA_PMDO_DON	Policy adjustment, assess moving primary processor donor down to occupied priority.



Table 89. SMF Type 99 Action Codes (continued)

531	PA_PCC_DON_VIOLTN	Policy adjustment, moving the donor to the receivers priority violates CPU critical rules.
532	PA_PCC_BLK_R_IS_DON	Policy adjustment, cannot move the blocker up because it is the donor.
533	PA_PCC_BLK_R_IS_REC	Policy adjustment, cannot move the blocker down because it is the receiver.
534	PA_PCC_BLK_R_NETVAL	Policy adjustment, moving blocker down fails net value check.
540	PA_PMDU_DON	Policy adjustment, assess moving primary processor donor down to unoccupied priority.
550	PA_PMD_DON_NETVAL	Policy adjustment, processor move down, rejected for no net value, donor trace, affected by resource donor.
552	PA_PMD_DON_NVL_SD	Policy adjustment, processor move down, rejected for no net value, donor trace, affected by secondary donor.
560	PA_PMD_GDON_NETVAL	Policy adjustment, processor move down, rejected for no net value, goal donor trace, affected by resource donor.
562	PA_PMD_GDON_NVL_SD	Policy adjustment, processor move down, rejected for no net value, goal donor trace, affected by secondary donor.
565	PA_PMD_GREC_NETVAL	Policy adjustment, processor move down, rejected for no net value, goal receiver trace, affected by resource donor.
567	PA_PMD_GREC_NVL_SD	Policy adjustment, processor move down, rejected for no net value, goal receiver trace, affected by secondary donor.
570	PA_PMD_RDON_NETVAL	Policy adjustment, processor move down, rejected for no net value, resource donor trace, affected by resource donor.
572	PA_PMD_RDON_NVL_SD	Policy adjustment, processor move down, rejected for no net value, resource donor trace, affected by secondary donor.

Table 89. SMF Type 99 Action Codes (continued)

573	PA_PMD_REC_NETVAL	Policy adjustment, processor move down, rejected for no net value, receiver trace, affected by resource donor.
575	PA_PMD_REC_NVL_SD	Policy adjustment, processor move down, rejected for no net value, receiver trace, affected by secondary donor.
576	PA_PMD_RREC_NETVAL	Policy adjustment, processor move down, rejected for no net value, resource receiver trace, affected by resource donor.
578	PA_PMD_RREC_NVL_SD	Policy adjustment, processor move down, rejected for no net value, resource receiver trace, affected by secondary donor.
580	PA_PMD_SEC_DON	Policy adjustment, assess moving secondary processor donor down.
590	PA_PMU_DON_NETVAL	Policy adjustment, processor move up, rejected for no net value, donor trace.
595	PA_PMU_DON_SEC_REC	Policy adjustment, processor assess moving donor up as secondary receiver.
600	PA_PMU_GDON_NETVAL	Policy adjustment, processor move up, rejected for no net value, goal donor trace.
605	PA_PMU_GREC_NETVAL	Policy adjustment, processor move up, rejected for no net value, goal receiver trace.
610	PA_PMU_RDON_NETVAL	Policy adjustment, processor move up, rejected for no net value, resource donor trace.
613	PA_PMU_REC_NETVAL	Policy adjustment, processor move up, rejected for no net value, receiver trace.
616	PA_PMU_RREC_NETVAL	Policy adjustment, processor move up, rejected for no net value, resource donor trace.
620	PA_PMUO_REC	Policy adjustment, assess moving primary processor receiver up to occupied priority.
630	PA_PMUUA_REC	Policy adjustment, assess moving primary processor receiver up to unoccupied priority above donor.

Table 89. SMF Type 99 Action Codes (continued)

635	PA_PMUUB_REC	Policy adjustment, assess moving primary processor receiver up to unoccupied priority between donor and receiver's current priorities.
640	PA_PMU_SEC_REC	Policy adjustment, assess moving secondary processor receiver up.
650	PA_PMU_TO_SPC_DP	Policy adjustment, move up to small processor consumer priority.
651	PA_PMU_SPC_NXT_DP	Policy adjustment, move up small processor consumer to next priority.
655	PA_PMU_SPC_UP_FAIL	Policy adjustment, moving up small processor consumer failed because the move is blocked by CPU critical period.
660	PA_PRO_DECP_DON	Policy adjustment, decrease priority for donor.
665	PA_PRO_DECP_MPL	Policy adjustment, decrease priority because of an mpl increase.
670	PA_PRO_DECP_SEC	Policy adjustment, decrease priority for secondary donor or receiver.
675	PA_PRO_DECP_BLK	Policy adjustment, decrease priority for blocker period.
690	PA_PRO_DON_DEPEN	Policy adjustment, no further processor action because of donor dependency relationship.
720	PA_PRO_GREC_NETVAL	Policy adjustment, no processor action because insufficient net value, goal receiver trace.
730	PA_PRO_GREC_RECVAL	Policy adjustment, no processor action because insufficient receiver value, goal receiver trace.
740	PA_PRO_INCP_DON	Policy adjustment, increase priority for donor.
750	PA_PRO_INCP_REC	Policy adjustment, increase priority for receiver.
760	PA_PRO_INCP_SEC	Policy adjustment, increase priority for secondary donor or receiver.
770	PA_PRO_INCP_BLK	Policy adjustment, increase priority for blocker period.

Table 89. SMF Type 99 Action Codes (continued)

780	PA_PRO_INCP_SC	Policy adjustment, increase priority for small consumer period.
850	PA_PRO_NA_NO_DONOR	Policy adjustment, no processor action because no donor selected.
870	PA_PRO_NA_SPC_DP	Policy adjustment, no processor action because period is at or just moved from small processor consumer priority.
880	PA_PRO_RDON_CAND	Policy adjustment, processor resource donor candidate selected.
890	PA_PRO_REC_DEPEN	Policy adjustment, no further processor action because of receiver dependency relationship.
900	PA_PRO_REC_NETVAL	Policy adjustment, no processor action because insufficient net value, receiver trace.
910	PA_PRO_REC_RECVAL	Policy adjustment, no processor action because insufficient receiver value, receiver trace.
920	PA_PRO_RREC_NETVAL	Policy adjustment, no processor action because insufficient net value, resource receiver trace.
930	PA_PRO_RREC_RECVAL	Policy adjustment, no processor action because insufficient receiver value, resource receiver trace.
933	PA_PRO_SERVED_GDON	Policy adjustment, served goal donor selected.
936	PA_PRO_SERVED_GREC	Policy adjustment, served goal receiver selected.
938	PA_PRO_TO_SPC_DP	Policy moved to small processor consumer.
939	PA_PRO_SPC_UP_FAIL	Policy adjustment, small processor user move up failed.
940	PA_PRO_UNC_DON	Policy adjustment, unchanged donor.
950	PA_PRO_UNC_REC	Policy adjustment, unchanged receiver.
960	PA_PRO_UNC_SEC_DON	Policy adjustment, unchanged secondary donor.
970	PA_PRO_UNC_SEC_REC	Policy adjustment, unchanged secondary receiver.

Table 89. SMF Type 99 Action Codes (continued)

975	PA_SDO_DONFAIL_SPC	Policy adjustment, select donor failed selecting period as the donor because period is small consumer.
976	PA_SDO_ADD_DGRP	Policy adjustment, move period from one donor group to another, add donor group.
978	PA_SDO_CLR_FLGS	Policy adjustment, select donor, clear cannot donate storage flags.
980	PA_TA_EA_MOV_UBA	Policy adjustment, tuning alias, efficiency-based adjustment, move unbound alias.
981	PA_TA_EA_MOV_BDEV	Policy adjustment, tuning alias, efficiency-based adjustment, move base device.
982	PA_TA_EA_NA_TIME	Policy adjustment, tuning alias, efficiency-based adjustment, no action due to PAV subsystem time since last alias move not exceeding one minute.
983	PA_TA_EA_NA_DONPIO	Policy adjustment, tuning alias, efficiency-based adjustment, donor not selected because donor projected increase in queued I/O requests exceeds threshold.
984	PA_TA_EA_NA_IOSQL	Policy adjustment, tuning alias, efficiency-based adjustment, receiver not selected because of insufficient average queued I/O requests.
987	PA_TA_EA_DON_L1MIN	Policy adjustment, tuning alias, efficiency-based adjustment, donor not selected because donor donated recently.
988	PA_TA_EA_REC_L1MIN	Policy adjustment, tuning alias, efficiency-based adjustment, receiver not selected because receiver was helped recently.
989	PA_TA_EA_NA_CUQDT	Policy adjustment, tuning alias, efficiency-based adjustment, receiver not selected due to excessive control unit queueing delay.

Table 89. SMF Type 99 Action Codes (continued)

990	PA_TA_GA_MOV_UBA	Policy adjustment, tuning alias, goal-based adjustment, move unbound alias.
991	PA_TA_GA_MOV_BDEV	Policy adjustment, tuning alias, goal-based adjustment, move base device.
992	PA_TA_GA_INV_RDEV	Policy adjustment, tuning alias, goal-based adjustment, invalid receiver device.
993	PA_TA_GA_NA_DONPIO	Policy adjustment, tuning alias, goal-based adjustment, donor not selected because donor projected increase in queued I/O requests exceeds threshold.
994	PA_TA_GA_NA_IOSQL	Policy adjustment, tuning alias, goal-based adjustment, receiver not selected because of insufficient average queued I/O requests.
995	PA_TA_GA_DON_L1MIN	Policy adjustment, tuning alias, goal-based adjustment, donor not selected because donor donated recently.
996	PA_TA_GA_REC_L1MIN	Policy adjustment, tuning alias, goal-based adjustment, receiver not selected because receiver was helped recently.
997	PA_TA_RRPATOD	When Tuning alias adjustment was entered.
998	PA_TA_GA_DONGTREC	Policy adjustment, tuning alias, goal-based adjustment, no action due to donor's importance greater than receiver's or donor belongs to a system service class.
999	PA_TA_GA_NA_CUQDT	Policy adjustment, tuning alias, goal-based adjustment, receiver not selected due to excessive control until queueing delay.
1000	PA_TA_EA_PASS_NO	Policy adjustment, tuning alias, efficiency adjustment, pass number.
1900	PA_OC9_suppressed	Policy adjustment.
2010	PA_DEC_PSI_TAR	Policy adjustment, decrease period protective processor storage target for this resource period.

Table 89. SMF Type 99 Action Codes (continued)

2011	PA_DEC_PSI_TAR_GP	Policy adjustment, decrease period protective processor storage target for a resource period associated with this goal period. The goal period is different than the resource period.
2020	PA_INC_PSI_TAR	Policy adjustment, increase period protective processor storage target for this resource period.
2021	PA_INC_PSI_TAR_GR	Policy adjustment, increase period protective processor storage target to the resource receiver associated with this goal receiver. The goal receiver is different than the resource receiver.
2030	PA_PSI_NA_NET_VAL	Policy adjustment, no period protective processor storage action because insufficient net value. Resource receiver trace.
2031	PA_PSI_GREC_NETVAL	Policy adjustment, no period protective processor storage action because insufficient net value. Goal receiver trace. The goal receiver is different than the resource receiver.
2040	PA_PSI_NA_REC_VAL	Policy adjustment, no period protective processor storage action because insufficient receiver value. Goal receiver trace.
2041	PA_PSI_RREC_RECVAL	Policy adjustment, no period protective processor storage action because insufficient receiver value. Resource receiver trace. The resource receiver is different than the goal receiver.
2050	PA_PSI_TAR_UNAB	Policy adjustment, no period protective processor storage action because current target not absorbed.
2060	PA_REM_PSI_TAR	Policy adjustment, remove period protective processor storage target for this resource period.

Table 89. SMF Type 99 Action Codes (continued)

2061	PA_REM_PSI_TAR_GP	Policy adjustment, remove period protective processor storage target for the resource period associated with this goal period. The goal period is different than the resource period.
2070	PLOT_X_REM_PSI_TAR	Plot expansion, remove period protective processor storage target for this resource period.
2071	PLOT_X_REM_PSI_GP	Plot expansion, remove period protective processor storage target for a resource period associated with this goal period. The goal period is different than the resource period.
2080	SH_DEC_PSI_TAR	Storage shortage, decrease period protective processor storage target for this resource period.
2081	SH_DEC_PSI_TAR_GP	Storage shortage, decrease period protective processor storage target for a resource period associated with this goal period. The goal period is different than the resource period.
2090	SH_REM_PSI_TAR	Storage shortage, remove period protective processor storage target for this resource period.
2091	SH_REM_PSI_TAR_GP	Storage shortage, remove period protective processor storage target from a resource period associated with this goal period. The goal period is different than the resource period.
2100	TDH_AS_DEC_PSI_TAR	Time driven housekeeping, decrease period protective processor storage target for this resource period.
2101	TDH_AS_DEC_PSI_GP	Time driven housekeeping, decrease period protective processor storage target for a resource period associated with this goal period. The goal period is different than the resource period.
2110	TDH_AS_REM_PSI_TAR	Time driven housekeeping, remove period protective processor storage target from this resource period.



Table 89. SMF Type 99 Action Codes (continued)

2111	TDH_AS_REM_PSI_GP	Time driven housekeeping, remove period protective processor storage target from a resource period associated with this goal period. The goal period is different than the resource period.
2120	TDH_ME_DEC_PSI_TAR	Time driven minimal effect housekeeping, decrease period protective processor storage target for this resource period.
2121	TDH_ME_DEC_PSI_GP	Time driven minimal effect housekeeping, decrease period protective processor storage target for a resource period associated with this goal period. The goal period is different than the resource period.
2130	TDH_ME_REM_PSI_TAR	Time driven minimal effect housekeeping, remove period protective processor storage target from this resource period.
2131	TDH_ME_REM_PSI_GP	Time driven minimal effect housekeeping, remove period protective processor storage target from a resource period associated with this goal period. The goal period is different than the resource period.
2140	TDH_UA_DEC_PSI_TAR	Time driven unassessable housekeeping, decrease period protective processor storage target for this resource period.
2141	TDH_UA_DEC_PSI_GP	Time driven unassessable housekeeping, decrease period protective processor storage target for a resource period associated with this goal period. The goal period is different than the resource period.
2150	TDH_UA_REM_PSI_TAR	Time driven unassessable housekeeping, remove period protective processor storage target from this resource period.

Table 89. SMF Type 99 Action Codes (continued)

2151	TDH_UA_REM_PSI_GP	Time driven unassessable housekeeping, remove period protective processor storage target from a resource period associated with this goal period. The goal period is different than the resource period.
2160	RV_HSK_INC_PSI_TAR	Reverse housekeeping, increment period protective processor storage target for this resource receiver.
2161	RV_HSK_INC_PSI_GR	Reverse housekeeping, increment period protective processor storage target for a resource receiver associated with this goal receiver. The goal receiver is different than the resource receiver.
2170	WSM_DEC_PSI_TAR	Working set management, decrease period protective processor storage target for this resource period.
2171	WSM_DEC_PSI_TAR_GP	Working set management, decrease period protective processor storage target for a resource period associated with this goal period. The goal period is different than the resource period.
2180	WSM_REM_PSI_TAR	Working set management, remove period protective processor storage target from this resource period.
2181	WSM_REM_PSI_TAR_GP	Working set management, remove period protective processor storage target from a resource period associated with this goal period. The goal period is different than the resource period.
2510	PA_DEC_PRT	Policy adjustment, decrease swap protect time.
2520	PA_INC_PRT	Policy adjustment, increase swap protect time.
2530	PA_PRT_NA_NET_VAL	Policy adjustment, no swap protect time action because insufficient net value.
2540	PA_PRT_NA_REC_VAL	Policy adjustment, no swap protect time action because insufficient receiver value.

Table 89. SMF Type 99 Action Codes (continued)

2550	PA_PRT_NA_SRVR_UD	Policy adjustment, no swap protect time action because period is a server or a universal donor.
2555	PA_PRT_NA_ENCLAVE	Policy adjustment, no swap protect time action because no policy adjustment actions for enclave swap delay.
2560	PA_PRT_NO_WSS	Policy adjustment, no swap protect time action because no average working set size to calculate frame projections.
2570	PA_PRT_TAR_UNAB	Policy adjustment, no swap protect time action because current target not absorbed.
2580	PA_REM_PRT	Policy adjustment, remove swap protect time.
2590	RV_HSK_INC_PRT	Reverse housekeeping increment swap protect time.
2600	SH_DEC_PRT	Storage shortage, decrease swap protect time.
2610	SH_REM_PRT	Storage shortage, remove swap protect time.
2620	TDH_DEC_PRT	Time driven housekeeping, decrease swap protect time.
2630	TDH_REM_PRT	Time driven housekeeping, remove swap protect time.
2640	WSM_DEC_PRT	Working set management, decrease swap protect time.
2650	WSM_REM_PRT	Working set management, remove swap protect time.
3010	PA_CSI_NA_NET_VAL	Policy adjustment, no common area protective processor storage target action because insufficient net value.
3020	PA_CSI_NA_REC_VAL	Policy adjustment, no common area protective processor storage target action because insufficient receiver value.
3030	PA_CSI_TAR_UNAB	Policy adjustment, no common area protective processor storage target action because current target not absorbed.
3040	PA_INC_CSI_TAR	Policy adjustment, increase common area protective processor storage target.

Table 89. SMF Type 99 Action Codes (continued)

3050	TDH_DEC_CSI_TAR	Time driven housekeeping, decrease common area protective processor storage target.
3060	TDH_REM_CSI_TAR	Time driven housekeeping, remove common area protective processor storage target.
3070	PA_INC_XMEM_TAR	Increase protective processor storage target to reduce cross memory paging.
3080	PA_XMEM_NA_NET_VAL	Did not increase protective processor storage target for cross memory paging because of insufficient net value.
3090	PA_XMEM_NA_REC_VAL	Did not increase protective processor storage target for cross memory paging because of insufficient receiver value.
3095	PA_XMEM_NA_SRT	Did not increase protective processor storage target for cross memory paging because address space faulted on was in a service period with short response time goals.
3100	PA_XMEM_TAR_UNAB	Did not increase protective processor storage target for cross memory paging because target was unabsorbed.
3110	TDH_DEC_SSI_TAR	Time driven housekeeping, decrease shared area protective processor storage target.
3120	PA_SHR_TAR_UNAB	Policy adjustment, no shared area protective processor storage target action because current target not absorbed.
3130	PA_SHR_NA_REC_VAL	Policy adjustment, no shared area protective processor storage target action because insufficient receiver value.
3140	PA_SHR_NA_NET_VAL	Policy adjustment, no shared area protective processor storage target action because insufficient net value.
3150	PA_INC_SHR_TAR	Policy adjustment, increase shared area protective processor storage target.

Table 89. SMF Type 99 Action Codes (continued)

3160	PA_DEC_SHR_DEL	Policy adjustment, decrease shared area protective processor storage target by delta in SPTE.
3510	B16M_SHORT_DEC_MPL	Below 16 meg storage shortage, decrease mpl.
3520	PA_DEC_MPL	Policy adjustment, decrease mpl.
3521	PA_DEC_MPL_GP	Policy adjustment, decrease mpl for a resource period that is associated with this goal period. The goal period is different than the resource period.
3530	PA_INC_MPL	Policy adjustment, increase mpl.
3531	PA_INC_MPL_TS	Policy adjustment, increase mpl for transaction servers.
3540	PA_INC_MPL_GR	Policy adjustment, increase mpl for storage managed enclave goal period.
3541	PA_INC_MPL_RR	Policy adjustment, increase mpl for storage managed enclave resource period.
3550	PA_MPL_NA_NET_VAL	Policy adjustment, no mpl action because insufficient net value.
3551	PA_MPL_NETVAL_RR	Policy adjustment, no mpl action because insufficient net value for storage managed enclave server periods.
3552	PA_MPL_NETVAL_GR	Policy adjustment, no mpl action because insufficient net value for storage managed enclave goal periods.
3560	PA_MPL_NA_REC_VAL	Policy adjustment, no mpl action because insufficient receiver value.
3561	PA_MPL_RECVAL_RR	Policy adjustment, no mpl action because insufficient receiver value for storage managed enclave servers.
3562	PA_MPL_RECVAL_GR	Policy adjustment, no mpl action because insufficient receiver value for storage managed enclave goal periods.
3580	PA_MPL_NA_SHORTAGE	Policy adjustment, no mpl action because system is in a storage shortage.

Table 89. SMF Type 99 Action Codes (continued)

3590	PA_SWAP_FOR_MPL	Policy adjustment, mpl assess picked this address space to swap out.
3600	TDH_DEC_MPL	Time driven housekeeping, decrease mpl.
3601	TDH_DEC_MPL_FOR_GR	Time driven housekeeping, decrease mpl for goal receiver.
3602	TDH_DEC_MPL_FOR_RR	Time driven housekeeping, decrease mpl for resource receiver.
3603	TDH_DEC_QMPL_GR	Time driven housekeeping, decrease QMPL for goal receiver.
3604	TDH_DEC_QMPL_RR	Time driven housekeeping, decrease QMPL for resource receiver.
3605	TDH_INC_QMPL_GR	Time-driven housekeeping, increase QMPL for goal receiver.
3606	TDH_INC_QMPL_RR	Time-driven housekeeping, increase QMPL for goal receiver.
3607	TDH_MOD_SERVINST	Time-driven housekeeping, modify number of server instances per address space.
3608	TDH_STRT_MIN_SP	Time-driven housekeeping, modify number of server instances per address space.
3609	TDH_RECOMM_FAILED	Time driven housekeeping, recommendation for additional address spaces or instances failed.
3610	RV_HSK_INC_MPL	Reverse housekeeping, increment mpl.
3613	TDH_DEC_QMOV_GR	Timer-driven housekeeping, decrease QMPL for move.
3614	TDH_DEC_QMOV_RR	Timer-driven housekeeping, decrease QMPL for move.
3615	TDH_DEC_QSWP_GR	Timer-driven housekeeping, decrease QMPL for swap out.
3616	TDH_DEC_QSWP_RR	Timer-driven housekeeping, decrease QMPL for swap out.
3617	TDH_DEC_QSVT_GR	Timer-driven housekeeping, decrease QMPL for service time.
3618	TDH_DEC_QSVT_RR	Timer-driven housekeeping, decrease QMPL for service time.

Table 89. SMF Type 99 Action Codes (continued)

3620	TDH_NA_INI_BAL	Balancing of initiators, no action.
3621	TDH_MPL_VCAL_ERR	Timer driven housekeeping, no decrease of mpl because invalid scope of projected velocity.
3622	TDH_MPL_SVLCAL_ERR	Timer driven housekeeping, no decrease of mpl because invalid scope of projected sysplex velocity.
4010	ESPOL_NSW_LRU	Change non-swap expanded access policy to lru.
4020	ESPOL_NSW_SP_AVAIL	Change non-swap expanded access policy to space available.
4050	ESPOL_SWP_LRU	Change swap expanded access policy to lru.
4060	ESPOL_SWP_SP_AVAIL	Change swap expanded access policy to space available.
4090	HSK_ROLL_EXP_SPA	Housekeep address space to space available because it is rolling expanded storage.
4200	STL_CR_AS_BLW_TRGT	Stealing found a storage critical address space which is below its processor protective target and RSM took more frames as requested, first trace.
4201	STL_CR_AS_BLW_TRG2	Stealing found a storage critical address space which is below its processor protective target and RSM took more frames as requested, second trace.
4202	STL_CR_AS_BLW_TRG3	Stealing found a storage critical address space which is below its processor protective target and RSM took more frames as requested, third trace.
4203	STL_CR_REQ_BLW_PPS	Steal request will bring a storage critical address space below its processor protective target.
4310	STOR_AFCOK_INC	The available frame queue target got increased.
4320	STOR_AFCOK_DEC	The available frame queue target got decreased.
4330	STOR_PRESTEALOK_INC	The presteal target got increased.

Table 89. SMF Type 99 Action Codes (continued)

4340	STOR_PRESTEALOK_DEC	The presteal target got decreased.
4510	ALL_OK_REM_ISI_TAR	All points ok, remove individual protective processor storage target for this address space.
4511	ALL_OK_REM_ISI_GP	All points ok, remove individual protective processor storage target from an address space in a resource period associated with this goal period. The goal period is different than the resource period.
4520	HSK_SL_DEC_ISI_TAR	Slow mode housekeeping, decrement individual protective processor storage target for this address space.
4521	HSK_SL_DEC_ISI_GP	Slow mode housekeeping, decrement individual protective processor storage target space for an address space in a resource period associated with this goal period. The goal period is different than the resource period.
4530	HSK_SL_REM_ISI_TAR	Slow mode housekeeping, remove individual protective processor storage target for this address space.
4531	HSK_SL_REM_ISI_GP	Slow mode housekeeping, remove individual protective processor storage target for an address space in a resource period associated with this goal period. The goal period is different than the resource period.
4540	OK1_INC_ISI_TAR	Ok1 increment individual protective processor storage target for this address space.
4541	OK1_INC_ISI_TAR_GR	Ok1 increment individual protective processor storage target for an address space in a resource receiver associated with this goal receiver. The goal period is different than the resource period.
4550	PA_DEC_ISI_TAR	Policy adjustment, decrease individual protective processor storage target for this address space.



Table 89. SMF Type 99 Action Codes (continued)

4551	PA_DEC_ISI_TAR_GP	Policy adjustment, decrease individual protective processor storage target for an address space in a resource period associated with this goal period. The goal period is different than the resource period.
4560	PA_INC_ISI_TAR	Policy adjustment, increase individual protective processor storage target for this address space.
4561	PA_INC_ISI_TAR_GR	Policy adjustment, increase individual protective processor storage target for an address space in a resource receiver associated with this goal receiver. The goal receiver is different than the resource receiver.
4570	PA_ISI_NA_NET_VAL	Policy adjustment, no individual protective processor storage action because insufficient net value. Resource receiver trace.
4571	PA_ISI_GREC_NETVAL	Policy adjustment, no individual protective processor storage action because insufficient net value. Goal receiver trace.
4580	PA_ISI_NA_REC_VAL	Policy adjustment, no individual protective processor storage action because insufficient receiver value for the goal receiver. Resource receiver trace.
4581	PA_ISI_GREC_RECVAL	Policy adjustment, no individual protective processor storage action because insufficient receiver value for the goal receiver. Goal receiver trace.
4590	PA_REM_ISI_TAR	Policy adjustment, remove individual protective processor storage target for this address space.
4591	PA_REM_ISI_TAR_GP	Policy adjustment, remove individual protective processor storage target from an address space in a resource period associated with this goal period. The goal period is different than the resource period.

Table 89. SMF Type 99 Action Codes (continued)

4592	PA_DEC_ISI_GDON	Goal donor trace when ISI target is reduced.
4600	PLOT_X_REM_ISI_TAR	Plot expansion, remove individual protective processor storage target for this address space.
4601	PLOT_X_REM_ISI_GP	Plot expansion, remove individual protective processor storage target from an address space in a resource period associated with this goal period. The goal period is different than the resource period.
4610	ROLL_EXP_REM_ISI	Remove individual protective processor storage target because address space target is rolling expanded for this address space.
4611	ROLL_EXP_REM_ISIGP	Remove individual protective processor storage target because address space target is rolling expanded for this address space, which is in a resource period associated with this goal period. The goal period is different than the resource period.
4620	RV_HSK_INC_ISI_TAR	Reverse housekeeping, increment individual protective processor storage target for this address space.
4621	RV_HSK_INC_ISI_GR	Reverse housekeeping, increment individual protective processor storage target for an address space in a resource receiver associated with this goal receiver. The goal receiver is different than the resource receiver.
4630	SH_DEC_ISI_TAR	Storage shortage, decrease individual protective processor storage target for this address space.
4631	SH_DEC_ISI_TAR_GP	Storage shortage, decrease individual protective processor storage target for an address space in a resource period associated with this goal period. The goal period is different than the resource period.

Table 89. SMF Type 99 Action Codes (continued)

4640	SH_REM_ISI_TAR	Storage shortage, remove individual protective processor storage target for this address space.
4641	SH_REM_ISI_TAR_GP	Storage shortage, remove individual protective processor storage target from an address space in a resource period associated with this goal period. The goal period is different than the resource period.
4650	TDH_ME_DEC_ISI_TAR	Time driven minimal effect housekeeping, decrease individual protective processor storage target for this address space.
4653	TDH_ME_DEC_ISI_GP	Time driven housekeeping minimal effect on goal period when housekeeping down the individual protective processor storage target for a transaction server period address space.
4660	TDH_ME_REM_ISI_TAR	Time driven minimal effect housekeeping, remove individual protective processor storage target for this address space.
4661	TDH_ME_REM_ISI_GP	Time driven minimal effect housekeeping, remove individual protective processor storage target from an address space in a resource period associated with this goal period. The goal period is different than the resource period.
4670	TDH_UA_DEC_ISI_TAR	Time driven unassessable housekeeping, decrease individual protective processor storage target for this address space.
4671	TDH_UA_DEC_ISI_GP	Time driven unassessable housekeeping, decrease individual protective processor storage target for an address space in a resource period associated with this goal period. The goal period is different than the resource period.

Table 89. SMF Type 99 Action Codes (continued)

4680	TDH_UA_REM_ISI_TAR	Time driven unassessable housekeeping, remove individual protective processor storage target for this address space.
4681	TDH_UA_REM_ISI_GP	Time driven unassessable housekeeping, remove individual protective processor storage target from an address space in a resource period associated with this goal period. The goal period is different than the resource period.
4690	WSM_DEC_ISI_TAR	Working set management, decrease individual protective processor storage target for this address space.
4691	WSM_DEC_ISI_TAR_GP	Working set management, decrease individual protective processor storage target for an address space in a resource period associated with this goal period. The goal period is different than the resource period.
4700	WSM_INC_ISI_TAR	Working set management, increase individual protective processor storage target for this address space.
4701	WSM_INC_ISI_TAR_GR	Working set management, increase individual protective processor storage target for an address space in a resource receiver associated with this goal receiver. The goal receiver is different than the resource receiver.
4710	WSM_REM_ISI_TAR	Working set management, remove individual protective processor storage target for this address space.
4711	WSM_REM_ISI_TAR_GP	Working set management, remove individual protective processor storage target from an address space in a resource period associated with this goal period. The goal period is different than the resource period.
4720	Hsk_cr_inc_ici_tar	Housekeep storage critical address space increment central protective target.

Table 89. SMF Type 99 Action Codes (continued)

4721	Hsk_cr_dec_ici_tar	Housekeep storage critical address space decrement central protective target.
4722	Hsk_cr_inc_ipi_tar	Housekeep storage critical address space increment processor protective target.
4723	Hsk_cr_dec_ipi_tar	Housekeep storage critical address space decrement processor protective target.
4724	Hsk_cr_inc_ici_gp	Housekeep storage critical space increment central protective target. Goal and resource periods are different.
4725	Hsk_cr_dec_ici_gp	Housekeep storage critical space decrement central protective target. Goal and resource periods are different.
4726	Hsk_cr_inc_ipi_gp	Housekeep storage critical space increment processor protective target. Goal and resource periods are different.
4727	Hsk_cr_dec_ipi_gp	Housekeep storage critical space decrement processor protective target. Goal and resource periods are different.
4730	Hsk_cr_rem_ipi_tar	Housekeep storage critical space remove protective processor target.
4740	Chp_cr_inc_ici_tar	Change period increment central protective target.
4743	Chp_cr_inc_ipi_tar	Change period increment processor protective Target.
4744	Chp_cr_inc_ipi_gp	Change period increment processor protective Target, goal and resource periods are different.
4747	inc_ipi_tar_blw_bw	The protective processor target cannot be incremented the new target would be below threshold for storage critical address spaces.
4750	pa_cr_no_action	No storage donation action was taken, because the address space was storage critical.

Table 89. SMF Type 99 Action Codes (continued)

4751	paaup_cr_no_action	No storage donation action was taken, because the address space was storage critical.
4752	palpd_cr_no_action	No storage donation action was taken, the address space was storage critical.
4760	pa_fst_outof_donor	Find_storage has to give up, because it did not find more donors.
4761	pa_fst_action	Find_storage, action bookkept.
4762	pa_fst_begin	Find_storage, begin.
4763	pa_fst_end	Find_storage, end.
4764	pa_fst_parms	Find_storage, trace parameters.
4768	pa_fst_isi_dnval_fd	Find_storage, check donation value failed.
4769	pa_fst_no_isi_sdon	Find_storage, the resource donor is not a WSM individual storage donor.
4770	pa_fst_isi_no_bactn	Find_storage, check donation value failed.
4771	pa_fst_no_bst_5as	Find_storage, no best five individual donor address spaces found.
5500	PA_DCM_INC_TAR	Policy adjustment, increase I/O velocity target. Resource receiver trace.
5501	PA_DCM_NA_NOPROB	Policy adjustment, no action, insufficient delay. Resource receiver trace.
5502	PA_DCM_NA_MAXVEL	Policy adjustment, no action, actual velocity or current target velocity is at maximum. Resource receiver trace.
5503	PA_DCM_NA_MAXTARG	Policy adjustment, no action, new target velocity is at maximum. Resource receiver trace.
5504	PA_DCM_NA_TAR_UNAB	Policy adjustment, no action, current target velocity is not being achieved. Resource receiver trace.
5505	PA_DCM_NA_RECVAL	Policy adjustment, no action, insufficient receiver value. Resource receiver trace.

Table 89. SMF Type 99 Action Codes (continued)

5506	PA_DCM_NA_SVC_INC	Policy adjustment, no action, current target velocity has caused service time to increase. Resource receiver trace.
5507	PA_DCM_NA_IOSCDT	Policy adjustment, no action, service error. Resource receiver trace.
5508	PA_DCM_WLM_HUNG	Policy adjustment, dynamic chpid management, WLM task has not run recently so it may be hung.
5510	PA_DCM_GREC	Policy adjustment, dynamic chpid management, goal receiver trace. The goal receiver is different from the resource receiver.
5515	PA_DCM_NO_SCMT_ROW	Policy adjustment, dynamic chpid management, period is beyond bounds of Service Class Measurement Table.
5516	PA_DCM_DROP_SUBSYS	Policy adjustment, dynamic chpid management, subsystem dropped from tracking either because its no longer eligible for management or there was a problem retrieving current data from IOS.
5517	PA_DCM_NEWSUB_ERR	Policy adjustment, dynamic chpid management, error from IOS service when trying to get data about a new subsystem to track.
5518	PA_DCM_GOALALG_ON	Policy adjustment, dynamic chpid management goal algorithm has been enabled.
5519	PA_DCM_GOALALG_OFF	Policy adjustment, dynamic chpid management goal algorithm has been disabled.
5520	HSK_DCM_BELOW_DEF	Housekeeping, dynamic chpid management, velocity target eliminated because it is below the default velocity.
5521	HSK_DCM_NO_DELAY	Housekeeping, dynamic chpid management, velocity target is reduced or eliminated because no period is experiencing delay.
5522	HSK_DCM_IOSCDT_ERR	Housekeeping, dynamic chpid management, target is reduced or eliminate because no period is experiencing delay.

Table 89. SMF Type 99 Action Codes (continued)

5530	IOV_SUBSYS	Sysevent loViolat, I/O subsystem trace.
5531	IOV_GREC_SYS	Sysevent loViolat, goal receiver, sysplex level trace.
5532	IOV_GREC_LOC	Sysevent loViolat, goal receiver, local system trace.
5533	IOV_GREC_REM	Sysevent loViolat, goal receiver, remote system trace.
5534	IOV_GREC_NETV_SYS	Sysevent loViolat, goal receiver, rejected for net value, sysplex level trace.
5535	IOV_GREC_NETV_LOC	Sysevent loViolat, goal receiver, rejected for net value, local system trace.
5536	IOV_GREC_NETV_REM	Sysevent loViolat, goal receiver, rejected for net value, remote system trace.
5537	IOV_GDON_NETV_SYS	Sysevent loViolat, goal donor, rejected for net value, sysplex level trace.
5538	IOV_GDON_NETV_LOC	Sysevent loViolat, goal donor, rejected for net value, local system trace.
5539	IOV_GDON_NETV_REM	Sysevent loViolat, goal donor, rejected for net value, remote system trace.
5540	IOV_RREC_NETV	Sysevent loViolat, resource receiver, rejected for net value.
5541	IOV_RDON_NETV	Sysevent loViolat, resource donor, rejected for net value.
5542	IOV_GDON_MIMP_SYS	Sysevent loViolat, goal donor, rejected for net value, sysplex level trace.
5543	IOV_GDON_MIMP_LOC	Sysevent loViolat, goal donor, most impacted, local system trace.
5544	IOV_GDON_MIMP_REM	Sysevent loViolat, goal donor, most impacted, remote system trace.
5545	IOV_NORECEIVER	Sysevent loViolat, no receiver found.
5546	IOV_NODONOR	Sysevent loViolat, no donor found.
5547	IOV_RC	Sysevent loViolat, return code.
5548	IOV_IREC_SYS	Sysevent loViolat, internal receiver, sysplex level trace.



Table 89. SMF Type 99 Action Codes (continued)

5549	IOV_IREC_LOC	Sysevent loViolat, internal receiver, local system trace.
5550	IOV_IREC_REM	Sysevent loViolat, internal receiver, remote system trace.
5551	IOV_IDON_SYS	Sysevent loViolat, internal donor, sysplex level trace.
5552	IOV_IDON_LOC	Sysevent loViolat, internal donor, local system trace.
5553	IOV_IDON_REM	Sysevent loViolat, internal donor, remote system trace.
5554	IOV_DEC_TAR	Sysevent loViolat, decrease I/O velocity target for subsystem whose target is violated.
5555	IOV_BAD_SUBSYS	Sysevent loViolat, I/O subsystem trace, bad subsystem index or velocity.
5556	IOV_RDON_MIMP	Sysevent loViolat, resource donor, most impacted.
5557	IOV_ADD_CHPID	Sysevent loViolat, add chpid proposal.
5558	IOV_DELETE_CHPID	Sysevent loViolat, delete chpid proposal.
5559	IOV_AVAILABILITY	Sysevent loViolat, availability need request.
6510	HSK_SL_DEC_ICI_TAR	Slow mode housekeeping, decrement protective central storage target.
6520	HSK_SL_REM_ICI_TAR	Slow mode housekeeping, remove protective central storage target.
6530	OK1_INC_ICI_TAR	Ok1 increment individual protective central storage target.
6540	PA_DEC_ICI_TAR	Policy adjustment, decrease the protective central storage target.
6550	PA_INC_ICI_TAR	Increase protective central storage target.
6560	PA_REM_ICI_TAR	Policy adjustment, remove the protective central storage target.
6570	PLOT_X_REM_ICI_TAR	Remove restrictive processor target for phase change.
6580	SH_DEC_ICI_TAR	Shortage, decrease protective central storage target.
6590	SH_REM_ICI_TAR	Shortage, remove protective central storage target.

Table 89. SMF Type 99 Action Codes (continued)

6600	SWAPIN_DEC_ICI_TAR	Decrease protective central storage target at swap in because we cannot get enough frames to run address space at target.
6610	SWAPIN_REM_ICI_TAR	Remove protective central storage target at swap in because we cannot get enough frames to run address space at target.
6620	WSM_DEC_ICI_TAR	Working set management, decrease protective central storage target.
6630	WSM_INC_ICI_TAR	Working set management, increase protective central storage target.
6640	WSM_REM_ICI_TAR	Working set management, remove protective central storage target.
7110	SWAPIN_REM_RCS_TAR	Remove restrictive central target because we can swap address space in at ok1 point.
7120	SWAPIN_SET_RCS_TAR	Set restrictive central storage target at swap in because we cannot get enough frames to run address space at ok1.
7510	OTL_USE_DISC_CENT	Working set management, out too long use discretionary central.
7520	WSM_DEC_MPL	Working set management, decrease mpl.
7521	WSM_DEC_MPL_GP	Working set management, decrease mpl for a resource period associated with this goal period. The goal period and the resource period are different.
7530	WSM_END_A2B_CNT	Working set management, end a's frames to b central storage interval.
7540	WSM_END_A2B_PSTOR	Working set management, end a's frames to b processor stor interval.
7550	WSM_END_OK1	Working set management, end ok1 interval.
7560	WSM_END_OK1_BY_STL	Working set management, end ok1 interval.
7570	WSM_END_OK1_RUN_OK	Working set management, end ok1 interval.

Table 89. SMF Type 99 Action Codes (continued)

7590	WSM_END_SWAPIN	Working set management, end a's frames to b swapping interval.
7600	WSM_END_TRYLRU	Working set management, end trylrु interval.
7610	WSM_NA_MP1	Working set management, no action was taken in MP1.
7620	WSM_NA_NET_VAL	Working set management, find_storage couldn't find enough storage for the action.
7630	WSM_NA_NPCR_VAL	Working set management, not enough net productive time gain from this action.
7640	WSM_STRT_A2B_CNT	Working set management, start a's frames to b central storage interval.
7650	WSM_STRT_A2B_PSTOR	Working set management, start a's frames to b processor storage interval.
7660	WSM_STRT_OK1	Working set management, start ok1 interval.
7670	WSM_START_OTL_IN	Working set management, start out too long swap in interval.
7690	WSM_STRT_SWAPIN	Working set management, start a's frames to b swapping interval.
7700	WSM_STRT_TRYLRU	Working set management, start trylrु interval.
7710	WSM_USE_DISC_CENT	Working set management, use discretionary central.
7720	WSM_USE_DISC_EXP	Working set management, use discretionary expanded.
8010	PA_CAP_DECS	Decrease cap slices.
8020	PA_CAP_INCS	Increase cap slices.
8025	PA_CAP_GETMAIN	A new CAP pattern is getmained.
8030	PA_DRGROU_P_ADD	Dynamic resource group created.
8040	PA_DRGROU_P_DELETE	Dynamic resource group deleted.
8050	PA_DRGROU_P_MRK_DEL	Dynamic resource group marked for deletion.
8055	PA_DRGROU_P_MRK_ALL	All dynamic resource groups marked for deletion.
8060	PA_DRGROU_P_EXCHG	Dynamic resource group exchanged.

Table 89. SMF Type 99 Action Codes (continued)

8070	PA_DRGROUPE_MAX_INC	Dynamic resource group maximum service.
8075	PA_DRGROUPE_MAX_NI	Dynamic resource group maximum service rate not increased.
8080	PA_DRGROUPE_MAX_DEC	Dynamic resource group maximum service rate decreased.
8090	PA_DRGROUPE_ADD_INT	Dynamic resource group add initiators.
8095	PA_DRGROUPE_TEST	Dynamic resource test trace.
8500	HSK_FROM_SPC_IODP	Housekeeping, move from small I/O consumer priority, period is no longer small consumer.
8510	HSK_TO_SPC_IODP	Housekeeping, move to small I/O consumer priority.
8520	HSK_XFROM_SPC_IODP	Housekeeping, exchange from small I/O consumer priority to make room for another small consumer.
8525	HSK_UNBUNCH_IOPRTY	Housekeeping, unbunch I/O priorities.
8530	PA_IMDO_DON	Policy adjustment, assess moving primary I/O donor down to occupied priority.
8540	PA_IMDU_DON	Policy adjustment, assess moving primary I/O donor down to unoccupied priority.
8550	PA_IMD_DON_NETVAL	Policy adjustment, I/O move down, rejected for no net value, donor trace, affected by resource donor.
8552	PA_IMD_DON_NVL_SD	Policy adjustment, I/O move down, rejected for no net value, donor trace, affected by secondary donor.
8560	PA_IMD_GDON_NETVAL	Policy adjustment, I/O move down, rejected for no net value, goal donor trace, affected by resource donor.
8562	PA_IMD_GDON_NVL_SD	Policy adjustment, I/O move down, rejected for no net value, goal donor trace, affected by secondary donor.
8565	PA_IMD_GREC_NETVAL	Policy adjustment, I/O move down, rejected for no net value, goal receiver trace, affected by resource donor.

Table 89. SMF Type 99 Action Codes (continued)

8567	PA_IMD_GREC_NVL_SD	Policy adjustment, I/O move down, rejected for no net value, goal receiver trace, affected by secondary donor.
8570	PA_IMD_RDON_NETVAL	Policy adjustment, I/O move down, rejected for no net value, resource donor trace, affected by resource donor.
8572	PA_IMD_RDON_NVL_SD	Policy adjustment, I/O move down, rejected for no net value, resource donor trace, affected by secondary donor.
8573	PA_IMD_REC_NETVAL	Policy adjustment, I/O move down, rejected for no net value, receiver trace, affected by resource donor.
8575	PA_IMD_REC_NVL_SD	Policy adjustment, I/O move down, rejected for no net value, receiver trace, affected by secondary donor.
8576	PA_IMD_RREC_NETVAL	Policy adjustment, I/O move down, rejected for no net value, resource receiver trace, affected by resource donor.
8578	PA_IMD_RREC_NVL_SD	Policy adjustment, I/O move down, rejected for no net value, resource receiver trace, affected by secondary donor.
8580	PA_IMD_SEC_DON	Policy adjustment, assess moving secondary I/O donor down.
8590	PA_IMU_DON_NETVAL	Policy adjustment, I/O move up, rejected for no net value, donor trace.
8595	PA_IMU_DON_SEC_REC	Policy adjustment, I/O assess moving donor up as secondary receiver.
8600	PA_IMU_GDON_NETVAL	Policy adjustment, I/O move up, rejected for no net value, goal donor trace.
8605	PA_IMU_GREC_NETVAL	Policy adjustment, I/O move up, rejected for no net value, goal receiver trace.
8610	PA_IMU_RDON_NETVAL	Policy adjustment, I/O move up, rejected for no net value, resource donor trace.
8613	PA_IMU_REC_NETVAL	Policy adjustment, I/O move up, rejected for no net value, receiver trace.

Table 89. SMF Type 99 Action Codes (continued)

8616	PA_IMU_RREC_NETVAL	Policy adjustment, I/O move up, rejected for no net value, resource donor trace.
8620	PA_IMUO_REC	Policy adjustment, assess moving primary I/O receiver up to occupied priority.
8630	PA_IMUUA_REC	Policy adjustment, assess moving I/O processor receiver up to unoccupied priority above donor.
8635	PA_IMUUB_REC	Policy adjustment, assess moving I/O processor receiver up to unoccupied priority between donor and receiver's current priorities.
8640	PA_IMU_SEC_REC	Policy adjustment, assess moving secondary I/O donor up.
8650	PA_IMU_TO_SPC_DP	Policy adjustment, move up to small I/O consumer priority.
8660	PA_IO_DECP_DON	Policy adjustment, decrease priority for donor.
8670	PA_IO_DECP_SEC	Policy adjustment, decrease priority for secondary donor or receiver.
8690	PA_IO_DON_DEPEN	Policy adjustment, no further I/O action because of donor dependency relationship.
8720	PA_IO_GREC_NETVAL	Policy adjustment, no I/O action because insufficient net value, goal receiver trace.
8730	PA_IO_GREC_RECVAL	Policy adjustment, no I/O action because insufficient receiver value, goal receiver trace.
8740	PA_IO_INCP_DON	Policy adjustment, increase priority for donor.
8750	PA_IO_INCP_REC	Policy adjustment, increase priority for receiver.
8760	PA_IO_INCP_SEC	Policy adjustment, increase priority for secondary donor or receiver.
8850	PA_IO_NA_NO_DONOR	Policy adjustment, no processor action because no donor selected.
8870	PA_IO_NA_SPC_DP	Policy adjustment, no I/O action because period is at or just moved from small processor consumer priority.

Table 89. SMF Type 99 Action Codes (continued)

8880	PA_IO_RDON_CAND	Policy adjustment, I/O resource donor candidate selected.
8890	PA_IO_REC_DEPEN	Policy adjustment, no further I/O action because of receiver dependency relationship.
8900	PA_IO_REC_NETVAL	Policy adjustment, no I/O action because insufficient net value, receiver trace.
8910	PA_IO_REC_RECVAL	Policy adjustment, no I/O action because insufficient receiver value, receiver trace.
8920	PA_IO_RREC_NETVAL	Policy adjustment, no I/O action because insufficient net value, resource receiver trace.
8930	PA_IO_RREC_RECVAL	Policy adjustment, no I/O action because insufficient receiver value, resource receiver trace.
8933	PA_IO_SERVED_GDON	Policy adjustment, served goal donor selected.
8936	PA_IO_SERVED_GREC	Policy adjustment, served goal receiver selected.
8938	PA_IO_TO_SPC_DP	Policy moved to small I/O consumer.
8940	PA_IO_UNC_DON	Policy adjustment, unchanged donor.
8950	PA_IO_UNC_REC	Policy adjustment, unchanged receiver.
8960	PA_IO_UNC_SEC_DON	Policy adjustment, unchanged secondary donor.
8970	PA_IO_UNC_SEC_REC	Policy adjustment, unchanged secondary receiver.
8975	PA_IO_NA_TOO_SOON	Policy adjustment, no I/O action, too soon since last change.
8980	PA_IO_NA_NO_CLUST	Policy adjustment, no I/O action, no clusters have been built.
8985	PA_IO_NA_REC_INEL	Policy adjustment, no I/O action, receiver not eligible.
8990	PA_IO_IMPLEMENT	Policy adjustment, implement I/O changes. The changes are only trace on the system that made the change.
9001	PA_BP_DON_PER	Buffer pool owning period, potential donor.

Table 89. SMF Type 99 Action Codes (continued)

9002	PA_BP_DON_PER_VAL	Buffer pool owning period, assess values of potential donation: current buffer pool size, donor period pi delta, receiver period pi delta.
9003	PA_BP_AFF_PER_VAL	Buffer pool affected/sharing period, assess values of potential donation: current buffer pool size, affected period pi delta, receiver period pi delta.
9004	PA_BP_AFF_PER_SUC	Success of buffer pool assessment for affected/sharing period: 1 = successful, 0 = unsuccessful.
9005	PA_BP_ONE_UNSUC	Assessment of buffer pool resource donation for all periods using a common buffer pool had a negative result, the assessment of at least one period using the common buffer pool failed.
9006	PA_BP_ALL_SUC	Assessment of buffer pool resource donation for all periods using a common buffer pool had a positive result (all assessments finished without error).
9010	PA_DEC_BP_TAR	Policy adjustment, decrease bp storage target.
9020	PA_INC_BP_TAR	Policy adjustment, increase bp storage target.
9030	PA_BP_NA_NET_VAL	Policy adjustment, no period bp storage action because insufficient net value.
9040	PA_BP_NA_REC_VAL	Policy adjustment, no period bp storage action because insufficient receiver value.
9050	PA_BP_TAR_UNAB	Policy adjustment, no bp storage action because current target not absorbed.
9060	PA_BP_NA_EXIT_FAIL	Policy adjustment, no bp storage action because exit called failed.
9070	PA_PRIREQ_LONG_DEL	Policy adjustment, priority requests are waiting for longer than 60 seconds.
9071	PA_PRIREQ_NO_CPU	Policy adjustment, could not find CPU for additional server to handle priority requests.



Table 89. SMF Type 99 Action Codes (continued)

9072	PA_PRIREQ_NO_STOR	Policy adjustment, could not find storage for additional server to handle priority requests.
9075	PA_PRIREQ_NA_PEND	Policy adjustment, a previous recommendation to start up a space still exists or the first space has not bound yet.
9079	PA_PRIREQ_START_SA	Policy adjustment, priority request algorithms start a server address space.
9170	WSM_DEC_BP_TAR	Working set management, decrease bp storage target.
9180	PA_QMPL_NA_REC	Qmpl recommendations not allowed.
9190	PA_QMPL_NA_STOR	No qmpl actions taken because a critical shortage condition exists.
9191	PA_QMPL_AUX_STOR	Available auxilliary storage for server spaces.
9195	PA_QMPL_NA_RUA0	No qmpl actions taken because a ready user average is zero.
9200	PA_QMPL_NA_MPL	No qmpl actions taken because a mpl problem exists for this period.
9202	PA_QMPL_NA_IDLE	No qmpl actions taken because already idle inits and there is queued work. Let queued work be picked up by the idle inits before starting more.
9205	PA_QMPL_NA_QUEUE	No qmpl actions taken because the resource period isn't a queue server.
9210	PA_QMPL_NA_PEND	No qmpl actions taken because previous qmpl recommendation(s) exists.
9220	PA_QMPL_NA_UNMGED	No qmpl actions taken because queue is unmanaged.
9230	PA_QMPL_NA_REC_RR	No qmpl actions taken because there is no receiver value.
9240	PA_QMPL_NA_REC_GR	No qmpl actions taken because there is no receiver value.
9245	PA_QMPL_NA_SYSLOC	No qmpl actions taken because there is a better system to start initiators.

Table 89. SMF Type 99 Action Codes (continued)

9246	PA_QMPL_NA_NOSYS	No qmpl actions taken because there is no system to start initiators.
9250	PA_INC_QMPL_GR	Policy adjustment, increase qmpl for queue servers, goal receiver.
9251	PA_DEC_QMPL_GR	Policy adjustment, decrease qmpl for queue servers, goal receiver.
9260	PA_INC_QMPL_RR	Policy adjustment, increase qmpl for queue servers, resource receiver.
9261	PA_DEC_QMPL_RR	Policy adjustment, decrease qmpl for queue servers, resource receiver.
9270	PA_QMPL_NA_NETVAL	Policy adjustment, no qmpl action because insufficient net value for queue servers.
9280	PA_QMPL_NA_NO_REQ	Policy adjustment, no qmpl action because no requests queued.
9285	PA_QMPL_NA_GSMAX	No QMPL actions taken to increase resource group service maximum.
9294	PA_QMPL_LIMIT_AVT	The number of initiators started was limited to not more than number of available address spaces minus 10.
9295	ra_inc_qmpl_aff	Start an initiator for a batch work queue because of a specific affinity requirement.
9296	PA_QMPL_LIMIT_NUM	The number of initiators started was limited to not more than twice the current number of initiators.
9297	PA_QMPL_IMPACT_PER	Period most impacted by starting the initiators on this system.
9298	PA_QMPL_CPU_DON	Period whose CPU assess will be reduced by adding initiators on this system.
9299	PA_QMPL_INC_GSMAX	QMPL actions taken to increase resource group service maximum.
9300	PA_PPP_DECP_DON	Priority of the period that was low importance period because other high importance period was missing local goals.

Table 89. SMF Type 99 Action Codes (continued)

9301	PA_PPP_POT_REC	A period which is missing local goals due to significant CPU delay and detects one or more low importance period to help this period.
9305	PA_LMP_WT_CHANGE	Lpar weight management action taken to increase lpar weight, goal/resource receiver trace.
9306	PA_LMP_GWT_CHANGE	Lpar weight management action taken to increase lpar weight, goal receiver trace.
9307	PA_LMP_RWT_CHANGE	Lpar weight management action taken to increase lpar weight, resource receiver trace.
9308	PA_LMP_DON_NO_CAP	Donor image cannot donate service units required to increase weight of the system.
9309	PA_LMP_DIAG_FAIL	Failed during LPAR weight change processing.
9310	PA_LMP_REC_RECVAL	Lpar weight management, no weight change because of insufficient receiver value, goal/resource receiver trace.
9311	PA_LMP_GREC_RECVAL	Lpar weight management, no weight change because of insufficient receiver value, goal receiver trace.
9312	PA_LMP_RREC_RECVAL	Lpar weight management, no weight change because of insufficient receiver value, resource receiver trace.
9313	PA_LMP_REC_NETVAL	Lpar weight management, no weight change because of insufficient net value, goal/resource receiver trace.
9314	PA_LMP_GREC_NETVAL	Lpar weight management, no weight change because of insufficient net value, goal receiver trace.
9315	PA_LMP_RREC_NETVAL	Lpar weight management, no weight change because of insufficient net value, resource receiver trace.
9316	PA_LMP_DON_NETVAL	Lpar weight management, no weight change because of insufficient net value, goal/resource donor trace.

Table 89. SMF Type 99 Action Codes (continued)

9317	PA_LMP_GDON_NETVAL	Lpar weight management, no weight change because of insufficient net value, goal donor trace.
9318	PA_LMP_RDON_NETVAL	Lpar weight management, no weight change because of insufficient net value, resource donor trace.
9319	PA_LMP_DON_INV	The selected donor can not donate weight because of either the LDE is too old or the image is at minimum weight or the LPAR CPU management is deactivated for the donor partition.
9320	PA_LMP_REC_MAX_WT	The receiver may reach maximum weight if try to increase it's weight.
9321	PA_LMP_REC_TIMEINT	The receiver weight cannot be increased because not enough time is elapsed from last weight change.
9322	PA_LMP_REC_INV	The receiver weight cannot be increased because either failed or LPAR CPU management is deactivated for receiver.
9323	PA_LMP_DON_NETVOK	Lpar weight management, system is candidate for possible weight change because of sufficient net value for this goal/resource donor period.
9324	PA_LMP_GDON_NETVOK	Lpar weight management, system is candidate for possible weight change because of sufficient net value for this goal donor period.
9325	PA_LMP_RDON_NETVOK	Lpar weight management, system is candidate for possible weight change because of sufficient net value for this goal donor period.
9326	PA_CPU_ONLINE_REQ	Not enough CPUs are online. A request is queued invoke reconfig to bring required number of CPU on-line.
9327	PA_CPU_OFFLINE_REQ	More CPUs than necessary are online. A request is queued invoke reconfig to bring a CPU offline.
9328	PA_LMP_DON_CAND	Donor image candidate.

Table 89. SMF Type 99 Action Codes (continued)

9329	PA_LMP_RECVAL_OK	Sufficient receiver value.
9330	PA_LPCAP_PMAW	Logical partition is to be capped with a non-zero pricing management adjustment weight in order to enforce a soft cap.
9331	PA_LPCAP_PATTERN	Logical partition is to have capping turned on and off to enforce a soft cap.
9332	PA_LPCAP_CAP_ON	Logical partition was capped as part of enforcing a soft cap.
9333	PA_LPCAP_CAP_OFF	Logical partition was uncapped as part of enforcing a soft cap.
9334	PA_LPCAP_ON_ERR	An error occurred when trying to cap the logical partition as part of enforcing a soft cap.
9335	PA_LPCAP_OFF_ERR	An error occurred when trying to uncap the logical partition as part of enforcing a soft cap.
9336	PA_LPCAP_NODATA	Unable to cap or uncap the partition on this interval due to failure to obtain current partition data.
9337	PA_LPQUERY_ERR	Failed to obtain current partition data.
9338	PA_LPCAP_CONFIGCAP	WLM was capping the partition to enforce a soft cap. The partition was reconfigured to be capped all the time.
9339	PA_LPCAP_FIX_PMAW	WLM and PR/SM information is not in sync. PR/SM has a different pricing management adjustment weight. WLM information updated to match PR/SM.
9340	PA_LPCAP_FIX_OFF	WLM and PR/SM information is not in sync. According to PR/SM, WLM capping is not on. WLM information updated to match PR/SM.
9341	PA_LPCAP_FIX_ON	WLM and PR/SM information is not in sync. According to PR/SM, WLM capping is on. WLM information updated to match PR/SM.
9342	PA_LMP_GREC_RECOK	Sufficient receiver value - goal receiver.

Table 89. SMF Type 99 Action Codes (continued)

9343	PA_LMP_RREC_RECOK	Sufficient receiver value - resource receiver.
9344	PA_LMP_REC_CAND	Receiver image candidate.
9345	PA_LPCAP_PATTERN2	Logical partition is to have capping turned on and off to enforce a soft cap.
9346	PA_CPUS_ADJUSTED	Number of required online CPUs was adjusted due to VARYCPUMIN value.
9347	PA_CPU_D204TIME	CPU management processing skipped due to D204ElapsedTime being too small.
9398	PA_LMP_TEST	LPAR Mgmt test race.
9399	PA_LMP_TEST1	LPAR Mgmt test race.
9401	PA_LPD204_ERR	Weight management function failed.
9402	PA_LMP_REC_LOWUTIL	CPU utilization for receiver is below threshold.
9403	PA_PPP_MU_BLKD_PER	A period which is blocked due to CPU delay and moves it up to the next occupied priority if there was only work of equal or less importance at this priority.
9404	PA_GSL_HIGH_DELAY1	That delay time delta computed is not valid. The delay samples for the interval will be ignored.
9405	PA_GSL_HIGH_DELAY2	That delay time delta computed is not valid. The delay samples for the interval will be ignored.
9406	PA_GSL_LPAR_TIMES	Dispatch, delay and wait time non z/OS partition.
9407	PA_CA2_BLKD_PER_NS	PACA2 detects a period which is blocked due to CPU delay and gets no service - uses the projected response time dist only.
9408	PA_CA2_BLKD_PER_CM	PACA2 detects a period which is blocked due to CPU or MPL delay - uses the projected response time dist only.
9501	RA_PAE_MOV_UBA	Resource adjustment, paging availability enhancement, move unbound alias.
9502	RA_PAE_MOV_BDEV	Resource adjustment, paging availability enhancement, move base device alias.

Table 89. SMF Type 99 Action Codes (continued)

9531	SPV_PAE_INV_DEVNUM	Sysevent SETPVCNT, paging availability enhancement, invalid device number.
9532	SPV_PAE_PLIST_INVD	Sysevent SETPVCNT, paging availability enhancement, invalid device number from plist.
9988	PA_MD_WT_CPUU	Proj. Max demand calculation with projected CPUU samples.
9989	PA_MD_NO_CPUU	Proj. Max demand calculation without projected CPUU samples.
9991	PA_PAS_GREC_CPAS	TEMP - goal receiver current speed.
9992	PA_PAS_GREC_PROJ	TEMP - goal receiver current projected using and delay.
9993	PA_PAS_RREC_PROJ	TEMP - resource rec new projected using and delay.
9994	PA_PAS_RREC_SAMPD	TEMP - resource rec using & delay delta.
9995	PA_PAS_GDON_CPAS	TEMP - goal receiver current speed.
9996	PA_PAS_GDON_PROJ	TEMP - goal don current projected using and delay.
9997	PA_PAS_RDON_PROJ	TEMP - resource donor new projected using and delay.
9998	PA_PAS_RDON_SAMPD	TEMP - resource don using & delay delta.





## Appendix B. Application Validation Reason Codes

Table 90. SERVD validation reason codes

Section	Reason	Offset	Description
SERVD	3201	0	Beginning or end of some section as governed by the offset and length fields lies beyond the end of the SERVD
	3202	0	Beginning of some section as governed by the offset field lies within the middle of some other section
	3203	0	End of some section as governed by the offset and length fields lies within the middle of some other section
	3204	0	Some section, as governed by offset and length fields, straddles some other section
	3303	0	The SERVD has either an SVDEF, SVNPA, SVDGR, SVAEA, or an SVSEA offset as zeros.
SERVDHDR	3301	0	Eyecatcher (SERVD_EYECATCHER) is not 'SERVD '
	3302	0	Version (SERVD_VERSION) is 0
SVAEA	3701	SVAEA	Beginning or end of some section as governed by the offset and length fields lies beyond the end of the SVAEA.
	3702	SVAEA	Beginning of some section as governed by the offset field lies within the middle of some other section.
	3703	SVAEA	End of some section as governed by the offset and length fields lies within the middle of some other section.
	3704	SVAEA	Some section, as governed by offset and length fields, straddles some other section.
SVAEAHDR	3801	SVAEA	Eyecatcher (SVAEA_EYECATCHER) is not 'SVAE'.
	3802	SVAEA	Functionality level (SVAEA_FUNCTIONALITY_LEVEL) is zeros.
	3803	SVAEA	Header length (SVAEA_SIZE_OF_HEADER) is incorrect. Does not match the compiled size.
	3804	SVAEA	Application environment entry size (SVAEA_SIZE_AE) is incorrect. Does not match the compiled size.
	3805	SVAEA	Application environment offset (SVAEA_EXT_DATA_OFF) is incorrect. The offset is zero when SVAEA_EXT_DATA_LEN is non-zero.
	3806	SVAEA	The functionality level is less than SVAEA_LEVEL003 and the application environment entry offset or number are non-zero.
SVAEAAE	B901	entry	Duplicate application environment entry.
	3902	entry	Application environment (SVAEA_APPLICATION_ENVIRONMENT_NAME) is not specified.
	3903	entry	Application environment (SVAEA_APPLICATION_ENVIRONMENT_NAME) name is incorrect.
	3904	entry	Sybsystem type (SVAEA_SUBSYSTEM_TYPE) is not specified or is incorrect.

Table 90. SERVD validation reason codes (continued)

Section	Reason	Offset	Description
	3905	entry	Procedure name (SVAEA_PROCEDURE_TYPE) is not specified or is incorrect.
	3906	entry	WLM options (SVAEA_WLM_OPTIONS) has some reserved flags on.
	3907	entry	WLM options (SVAEA_WLM_OPTIONS) has the single server flag (SVAEA_SINGLE_SERVER) on for a subsystem type that does not support the option.
	3908	entry	WLM options (SVAEA_WLM_OPTIONS) has the single sysplex flag (SVAEA_SINGLE_SYSPLEX) on for a subsystem type that does not support the option.
SVAEAEXT	3A01	entry	Extension entry refers to an object (SVAEAROB) that does not exist.
	3A02	entry	End of data (SVAEAEDO + SVAEAEDL) extends beyond the size of the extended data section.
	3A03	entry	Use extension information found and the functionality level is less than LEVEL003. For user extensions, you must be at least at functionality LEVEL003 (LEVEL003 in SVAEA_FUNCTIONALITY_LEVEL).
SVDEF	0001	SVDEF	Beginning or end of some section as governed by the offset and length fields lies beyond the end of the SVDEF
	0002	SVDEF	Beginning of some section as governed by the offset field lies within the middle of some other section
	0003	SVDEF	End of some section as governed by the offset and length fields lies within the middle of some other section
	0004	SVDEF	Some section, as governed by offset and length fields, straddles some other section
SVDEFHDR	0101	SVDEF	Eyecatcher (SVDEFNAM) is not 'SVDE'
	0102	SVDEF	Functionality level is 0 (SVDEFLVL)
	0103	SVDEF	Only checked if releases match - Policy entry size (SVDEFPS) does not match compiled size
	0104	SVDEF	Only checked if releases match - Workload entry size (SVDEFWS) does not match compiled size
	0105	SVDEF	Only checked if releases match - Service class entry size (SVDEFCS) does not match compiled size
	0106	SVDEF	Only checked if releases match - Resource group entry size (SVDEFGS) does not match compiled size
	0107	SVDEF	Only checked if releases match - Report class entry size (SVDEFRS) does not match compiled size
	0108	SVDEF	Only checked if releases match - Service class attribute entry size (SVDEFCAS) does not match compiled size
	0109	SVDEF	Only checked if releases match - Resource group attribute section size (SVDEFGAS) does not match compiled size
	010A	SVDEF	Only checked if releases match - Constant entry size (SVDEFCONS) does not match compiled size
	010B	SVDEF	Only checked if releases match - Period entry size (SVDEFPCS) does not match compiled size

Table 90. SERVD validation reason codes (continued)

Section	Reason	Offset	Description
	010C	SVDEF	Size of the extended data (SVDEF_EXT_DATA_LEN) is nonzero, but the offset to the extended data (SVDEF_EXT_DATA_OFF) is zero
SVDEFPOL	8201	section	Number of policies (SVDEFPN) is 0
	8202	section	Number of policies (SVDEFPN) exceeds 99
	8203	offset	Duplicate policy names were found
	0201	entry	Name field (SVDEFPNM) has leading or imbedded blanks or contains a reserved character: * ? / , . ' ( ) & + - = ; -, =, ;)
	0202	entry	Name field (SVDEFPNM) starts with the letters SYS
SVDEFWKL	8301	offset	Duplicate workload names were found
	8302	section	Number of workloads (SVDEFWN) exceeds 999
	0301	entry	Name field (SVDEFWNM) has leading or imbedded blanks or contains a reserved character: * ? / , . ' ( ) & + - = ; -, =, ;)
	0302	entry	Name field (SVDEFWNM) starts with the letters SYS
SVDEFSCCL	8401	offset	Duplicate service class names were found
	8402	section	Number of service classes (SVDEFSCN) exceeds 100
	0401	entry	Name field (SVDEFSCNM) has leading or imbedded blanks or contains a reserved character: * ? / , . ' ( ) & + - = ; -, =, ;)
	0402	entry	Workload name (SVDEFSCWN) not found in the SVDEF
	0403	entry	Name field (SVDEFSCNM) starts with the letters SYS
	0404	entry	Base attribute for this service class not found in the SVDEF
SVDEFGRP	8601	offset	Duplicate resource group names were found
	8602	section	Number of resource groups (SVDEFGRN) exceeds 32
	0601	entry	Name field (SVDEFGRNM) has leading or imbedded blanks or contains a reserved character: * ? / , . ' ( ) & + - = ; -, =, ;)
	0602	entry	Base attribute for this service class not found in the SVDEF
SVDEFRCL	8701	offset	Duplicate report class names were found
	8702	section	Number of report classes (SVDEFGRN) exceeds 100
	0701	entry	Name field (SVDEFGRNM) has leading or imbedded blanks or contains a reserved character: * ? / , . ' ( ) & + - = ; -, =, ;)
SVDEFCLA	0801	entry	Named service class (SVDEFSCN) not found in service class list
	0802	entry	Named policy (SVDEFSPN) not found in policy list
	0803	entry	Named resource group (SVDEFGRN) not found in resource group list
	0804	entry	Number of periods (SVDEFPCN) out of bounds (must be from 1 to 8)

Table 90. SERVD validation reason codes (continued)

Section	Reason	Offset	Description
	0805	entry	CPU critical option is used (YES) and the service class contains more than 1 period.
	0806	entry	CPU critical option is used (YES) and the current functionality level in svdef is less than LEVEL011.
	0807	entry	Service class is used in CICS or IMS and other subsystem type in a service definition that is LEVEL011 or above.
SVDEFRGA	0901	entry	Named resource group (SVDEFRGN) not found in resource group list
	0902	entry	Named policy (SDVEFRPN) not found in policy list
	0903	entry	Specified minimum value (SVDEFGMN) exceeds 999,999
	0904	entry	Specified maximum value (SVDEFGMX) exceeds 999,999
	0905	entry	Specified minimum value (SVDEFGMN) exceeds maximum value (SVDEFGMX)
SVDEFPPDA	8501	entry	A period other than the last period has a duration (SVDEFDUR) of 0
	8502	entry	A period other than the last period has a discretionary goal (SVDEFDSC)
	8503	entry	Last period has a nonzero duration (SVDEFDUR)
	0501 0502 0503 0504	entry	More than one goal type specified (SVDEFTYP)
	0505	entry	Only checked if releases match - no known goal type (SVDEFTYP) specified
	0506	entry	Percentile or average response time goal specified (SVDEFPRC, SVDEF AVG), but response time units (SVDEFRTU) not between 1 and 4.
	0507	entry	Percentile, average response time, or velocity goal specified (SVDEFPRC, SVDEF AVG, SVDEFVEL), but importance (SVDEFIMP) is not between 1 and 5.
	0508	entry	Percentile or average response time goal (SVDEFPRC, SVDEF AVG) but response time value (SVDEFVAL) is less than 15 milliseconds.
	0509	entry	Percentile or average response time goal (SVDEFPRC, SVDEF AVG) and response time is greater than 24 hours
	050A	entry	Percentile goal (SVDEFPRC) and percentile value (SVDEFPER) exceeds 99
	050B	entry	Velocity goal (SVDEFVEL), and value (SVDEFVAL) exceeds maximum of 99
	050C	entry	Duration (SVDEFDUR) exceeds limit of 999,999,999
	050D	entry	For service definition with functionality LEVEL011 or above, a service class cannot contain any periods that have higher importance levels than previous periods
SVDEFPCNS	0A01	section	CPU coefficient (SDVEFCPU) exceeds maximum of 999,000

Table 90. SERVD validation reason codes (continued)

Section	Reason	Offset	Description
	0A02	section	I/O coefficient (SVDEFIOC) exceeds maximum of 999,000
	0A03	section	MSO coefficient (SVDEFMSO) exceeds maximum of 999,999
	0A04	section	SRB coefficient (SVDEFSRB) exceeds maximum of 999,000
	0A05	section	The dynamic alias management option is set to YES and the functionality level in the SVDEFLVL is not set to LEVEL008 or higher
	0A06	entry	The I/O priority management option is set to 'YES' and the functionality level in the SVDEFLVL is not set to LEVEL003 or higher.
SVDEFEXT	0B01	entry	Extension entry refers to an object (SVDEFROB) that does not exist
	0B02	entry	End of data (SVDEFEDO + SVDEFEDL) extends beyond the size of the extended data section
	0B03	entry	User extension information found and the functionality level is less than LEVEL002. For user extensions, you must be at least at functionality LEVEL002 (LEVEL002 in SVDEFLVL).
SVDCR	1901	SVDCR	Beginning or end of some section as governed by the offset and length fields lies beyond the end of the SVDCR
	1902	SVDCR	Beginning of some section as governed by the offset field lies within the middle of some other section
	1903	SVDCR	End of some section as governed by the offset and length fields lies within the middle of some other section
	1904	SVDCR	Some section, as governed by offset and length fields, straddles some other section
SVDCRHDR	1A01	SVDCR	Eyecatcher (SVDCRNAM) is not 'SVDC'
	1A02	SVDCR	Functionality level is 0 (SVDCRLVL)
	1A03	SVDCR	Only checked if releases match - Nesting level (SVDCRLN) does not match compiled nesting level limit (must be <= 4)
	1A04	SVDCR	Only checked if releases match - Subsystem entry size (SVDCRSS) does not match compiled size
	1A05	SVDCR	Only checked if releases match - Rule entry size (SVDCRRS) does not match compiled size
	1A06	SVDCR	Only checked if releases match - Group entry size (SVDCRGS) does not match compiled size
	1A07	SVDCR	Only checked if releases match - Group value entry size (SVDCRVVS) does not match compiled size
	1A08	SVDCR	Size of the extended data (SVDCR_EXT_DATA_LEN) is nonzero, but the offset to the extended data (SVDCR_EXT_DATA_OFF) is zero
	1A09	SVDCR	WLM version number (SVDCRWVN) is wrong. The functionality level is LEVEL002 or greater, and the WLM version number is 0.

Table 90. SERVD validation reason codes (continued)

Section	Reason	Offset	Description
SVDCRSST	9B01	offset	Duplicate subsystem type names were found
	1B01	entry	Service class for the subsystem type (SVDCRSCN) not found in the SVDEF
	1B02	entry	Report class for the subsystem type (SVDCRSPN) not found in the SVDEF
	1B03	entry	Number of classification rules (SVDCRSRN) is nonzero, but the offset (SVDCRSRO) is 0
SVDCRRUL	1C01	entry	More than one qualifier type (SVDCRRQT) bit is on
	1C02	entry	Group value specified (SVDCRRGI = '1'B) for an accounting information (SVDCRRAC) or subsystem parameter (SVDCRRSP) type rule - this is unsupported.
	1C03	entry	Qualifier value (SVDCRRQV) has leading or imbedded blanks
	1C04	entry	Substring (SVDCRRSU) wildcard (SVDCRRWI) or mask characters (SVDCRRSU) used on a rule that refers to a group (SVDCRRGI).
	1C05	entry	Nesting level of rule (SVDCRRLV) exceeds maximum nesting level indicated in the header (SVDCRLN)
	1C06	entry	Substring (SVDCRRSU) specified for a qualifier type that does not support substringing (hint: only accounting information (SVDCRRAC), subsystem parameter (SVDCRRSP), collection name (SVDCRRQT_COLL_NAME), correlation information (SVDCRRQT_CORR_INFO), procedure name (SVDCRRQT_PROC_NAME), or process name (SVDCRRQT_PROCESS_NAME) support substringing for this WLM MVS version).
	1C07	entry	Substring specified for accounting information extends beyond the end of the maximum size accounting information (143 characters)
	1C08	entry	Substring specified for subsystem parameter extends beyond the end of the maximum size subsystem parameter (255 characters) list
	1C09	entry	Substring specified for collection name extends beyond the end of the maximum size collection name (18 characters) list
	1C0A	entry	Substring specified for correlation information extends beyond the end of the maximum size correlation information (12 characters) list
	1C0B	entry	Service class (SVDCRRCN) was not found in the SVDEF service class list
	1C0C	entry	Service class (SVDCRRPN) was not found in the SVDEF report class list
	1C0D	entry	Substring specified (SVDCRRSU) but value (SVDCRRSV) is 0
	1C0E	entry	Group specified (SVDCRRGI) but named group (SVDCRRQV) not found in group list

Table 90. SERVD validation reason codes (continued)

Section	Reason	Offset	Description
	1C0F	entry	Classification rule was found that has indicators on in the SVDCR reserved for future qualifier bytes. The reserved for future qualifier types are in SVDCRRQT_BYTE3 and SVDCRRQT_BYTE4.
	1C10	entry	No classification rule found. SVDCRRQT is zeros.
	1C11	entry	LEVEL002 classification rules found and the functionality level in the SVDCRLVL is not LEVEL002.
	1C12	entry	LEVEL003 classification rules found and the functionality level in the SVDCRLVL is not LEVEL003.
	1C13	entry	For the procedure name the substring value is greater than 18 or the substring value plus the number of characters is greater than 18.
	1C14	entry	LEVEL004 classification rules found and the functionality level in the SVDCRLVL is not LEVEL004.
	1C15	entry	SVDCRRQV contains characters that are not allowed for a qualifier that takes numeric data. SVDCRRQV must contain a number (in EBCDIC) optionally preceded by one of the supported relational operators.
	1C16	entry	Classification rule comment is found that has description information in the SVDCR, but SVDCR is not at LEVEL006.
	1C17	entry	The PC (process name) classification type was found and the functionality level in the SVDCRLVL is not set to LEVEL007.
	1C18	entry	For the process name the substring value is greater than 32 or the substring value plus the number of characters is greater than 32.
	1C19	entry	Classification rules that uses LEVEL011 qualifier types were found and the functionality level in SVDCRLVL is set to less than LEVEL011.
	1C1A	entry	The SCHEDULING_ENVIRONMENT (SE) qualifier type specified is greater than 16, or the substring value plus the number of character is greater than 16.
	1C1B	entry	Subsystem type that doesn't support storage protection has a classification rule whose storage protection option is set to YES.
	1C1C	entry	The storage protection option is chosen within a classification rule, and the rule is using a service class with a short response time goal.
	1C1D	entry	The storage protection option is chosen within a classification rule, and the rule is using a service class with more than one period.
	1C1E	entry	The storage protection option is chosen within a classification rule, and the rule is using a service class with a discretionary goal.
	1C1F	entry	LEVEL011 classification rule that uses storage critical option was found, and the functionality level in the SVDCRLVL is set to less than LEVEL011.

Table 90. SERVD validation reason codes (continued)

Section	Reason	Offset	Description
	1C20	entry	LEVEL011 classification rule that uses transaction and region management options were found, and the functionality level in the SVDCRLVL is set to less than LEVEL011.
	1C21	entry	Subsystem type that doesn't support transaction region option has a classification rule set to region management.
SVDCRGRP	9D01	entry	Duplicates exist in the list of groups
	1D01	entry	Name field (SVDCRGNM) has leading or imbedded blanks or contains a reserved character: * ? / , . ' ( ) & + - = ; -, =, ;)
	1D02	entry	More than one qualifier type (SVDCRGTY) bit is on
	1D03	entry	Number of group values (SVDCRGVN) is 0
	1D04	entry	Offset to group values (SVDCRGVO) is 0
	1D05	entry	Group exists that has indicators on in the SVDCR that are reserved for future qualifier bytes (SVDCRGTY_BYTE3 and SVDCRGTY_BYTE4).
	1D06	entry	No groups found. SVDCRGTY is zeroes.
	1D07	entry	LEVEL002 classification groups found and the service definition functionality level (SVDCRLVL) is not LEVEL002.
	ID08	entry	LEVEL003 classification rules (SVDCRGTY_PERFORM) found and the functionality level in the SVDCRLVL is not LEVEL003.
	1D09	entry	LEVEL011 classification rules found and the functionality level in the SVDCRLVL is less than LEVEL011.
SVDCRGVS	9E01	entry	Duplicates exist in the list of group values for a given group
	1E01	entry	Group value (SVDCRGVV) has leading or imbedded blanks
	1E02	entry	LEVEL006 classification group comment found, but the functionality level in the SVDCRLVL is not set to LEVEL006.
SVDCREXT	1F01	entry	Extension entry refers to an object (SVDCRROB) that does not exist
SVDCREXT	1F02	entry	End of data (SVDCREDO + SVDCREDL) extends beyond the size of the extended data section
SVDCREXT	1F03	entry	User extension information found and the functionality level is less than LEVEL002. To use user extensions, you must be at least at LEVEL002 (LEVEL002 in SVDCRLVL).
SVNPA	2801	SVNPA	Beginning or end of some section as governed by the offset and length fields lies beyond the end of the SVDCR
	2802	SVNPA	Beginning of some section as governed by the offset field lies within the middle of some other section
	2803	SVNPA	End of some section as governed by the offset and length fields lies within the middle of some other section



Table 90. SERVD validation reason codes (continued)

Section	Reason	Offset	Description
	2804	SVNPA	Some section, as governed by offset and length fields, straddles some other section
SVNPAHDR	2901	SVNPA	Eyecatcher (SVNPANAM) is not 'SVNP'
	2902	SVNPA	Functionality level is 0 (SVNPALVL)
	2903	SVNPA	Only checked if releases match - Number of notepad entries (SVNPANPN) exceeds maximum allowed (500)
	2904	SVNPA	Only checked if releases match - Notepad data entry size (SVDEFNDS) does not match compiled size
SVSEA	3B01	SVSEA	Beginning or end of some section as governed by the offset and length fields lies beyond the end of the SVSEA.
	3B02	SVSEA	Beginning of some section as governed by the offset field lies within the middle of some other section.
	3B03	SVSEA	End of some section as governed by the offset and length fields lies within the middle of some other section.
	3B04	SVSEA	Some section, as governed by offset and length fields, straddles some other section.
SVSEAHDR	3C01	SVSEA	Eyecatcher (SVSEA_EYECATCHER) is not 'SVSE'.
	3C02	SVSEA	Functionality level (SVSEA_FUNCTIONALITY_LEVEL) is zeros.
	3C03	SVSEA	Header length (SVSEA_SIZE_OF_HEADER) is incorrect. Does not match the compiled size.
	3C04	SVSEA	Scheduling environment entry size (SVSEA_SIZE_SE) is incorrect. Does not match the compiled size.
	3C05	SVSEA	Scheduling environment to resource connection size (SVSEA_SIZE_SR) is incorrect. Does not match the compiled size.
	3C06	SVSEA	Resource size (SVSEA_SIZE_RE) is incorrect. Does not match the compiled size.
	3C05	SVSEA	Scheduling environment extension size (SVSEA_SIZE_EXT) is incorrect. Does not match the compiled size.
	3C08	SVSEA	The functionality level is less than SVSEA_LEVEL004 and the scheduling environment entry offset or number are non-zero.
SVSEASE	BD10	entry	Too many scheduling environment entries.
	BD11	entry	Duplicate scheduling environment entry.
	3D12	entry	Scheduling environment name (SVSEA_SE_SCHENV_NAME) is not specified.
	3D13	entry	Scheduling environment name (SVSEA_SE_SCHENV_NAME) is incorrect.
	3D14	entry	Scheduling environment name (SVSEA_SE_SCHENV_NAME) is reserved (cannot start with SYS).
SVSEASR	3D20	entry	In the scheduling environment to resource connection, the scheduling environment name (SVSEA_SR_SCHENV_NAME) is not specified.

Table 90. SERVD validation reason codes (continued)

Section	Reason	Offset	Description
	3D21	entry	In the scheduling environment to resource connection, the scheduling environment name (SVSEA_SR_SCHENV_NAME) is incorrect.
	3D22	entry	In the scheduling environment to resource connection, the resource name (SVSEA_SR_RESOURCE_NAME) is not specified.
	3D23	entry	In the scheduling environment to resource connection, the resource state (SVSEA_SR_RESOURCE_STATE) is not specified.
	3D24	entry	In the scheduling environment to resource connection, the resource state (SVSEA_SR_RESOURCE_STATE) is not valid.
	3D25	entry	In the scheduling environment to resource connection, the scheduling environment name (SVSEA_SR_SCHENV_NAME) is not in the list of defined scheduling environments.
	3D26	entry	In the scheduling environment to resource connection, the the resource name (SVSEA_SR_RESOURCE_NAME) is not in the list of defined resources.
SVSEARE	BD30	entry	Too many resource entries.
	BD31	entry	Duplicate resource entry.
	3D32	entry	Resource name (SVSEA_RE_RESOURCE_NAME) is not specified.
	3D33	entry	Resource name (SVSEA_RE_RESOURCE_NAME) is incorrect.
	3D34	entry	Resource name (SVSEA_RE_RESOURCE_NAME) is reserved (cannot start with SYS).
SVSEAEXT	3E01	entry	Extension entry refers to an object (SVSEAROB) that does not exist is not found.
	3E02	entry	End of data (SVSEAEDO + SVSEAEDL) extends beyond the size of the extended data section
	3E03	entry	User extension information found and the functionality level is less than LEVEL004. For user extensions, you must be at least at functionality LEVEL004 (LEVEL004 in SVDEFLVL).

## Appendix C. C Language Interfaces for Workload Management Services

The following Table 91 shows C language interfaces with their associated Workload Management service. See the *z/OS XL C/C++ Run-Time Library Reference* for more information on these and other C language interfaces.

See also “Interfaces for Sysplex Routing Services” on page 763 for the four C interfaces for accessing the WLM sysplex routing services, and “Interface for Querying a Virtual Server” on page 763 for the C interface to querying a virtual server.

Table 91. C Language Interfaces

C Language Interface	Associated WLM Service	Reference Chapter
CheckSchEnv	IWMSEDES	Chapter 48, “IWMSEDES – Scheduling Environments Determine Execution Service,” on page 413
ConnectExportImport	IWMCONN WORK_MANAGER=NO ROUTER=NO QUEUE_MANAGER=NO SERVER_MANAGER=NO EXPTIMPT=YES	“IWMCONN – Connecting to Workload Management” on page 773
ConnectServer	IWMCONN WORK_MANAGER=NO ROUTER=NO QUEUE_MANAGER=YES SERVER_MANAGER=YES	“IWMCONN – Connecting to Workload Management” on page 773
ConnectWorkMgr	IWMCONN WORK_MANAGER=YES ROUTER=NO QUEUE_MANAGER=YES SERVER_MANAGER=NO EXPTIMPT=YES	“IWMCONN – Connecting to Workload Management” on page 773
ContinueWorkUnit	IWMCREA TYPE=DEPENDENT	“IWMECREA – Create an Enclave” on page 798
CreateWorkUnit	IWMCREA TYPE=INDEPENDENT	“IWMECREA – Create an Enclave” on page 798
DeleteWorkUnit	IWMEDELE	“IWMEDELE – Delete an Enclave” on page 809
DisconnectServer	IWMDISC	“IWMDISC – Disconnecting from Workload Management” on page 792
ExportWorkUnit	IWMEXPT	Chapter 31, “IWMEXPT – WLM Export Service,” on page 285
ExtractWorkUnit	IWMESQRY	Chapter 29, “IWMESQRY – Query Enclave State,” on page 271
ImportWorkUnit	IWMIMPT	Chapter 32, “IWMIMPT – WLM Import Service,” on page 293

Table 91. C Language Interfaces (continued)

C Language Interface	Associated WLM Service	Reference Chapter
JoinWorkUnit	IWMJOIN	Chapter 22, "IWMJOIN – Join an Enclave," on page 223
LeaveWorkUnit	IWMELEAV	Chapter 23, "IWMELEAV – Leave an Enclave," on page 229
QueryMetrics	IWMWSYSQ	Chapter 63, "IWMWSYSQ – Querying System Information," on page 515
QuerySchEnv	IWMSEQRY	Chapter 49, "IWMSEQRY – Scheduling Environments Query Service," on page 419
QueryWorkUnitClassification	IWMECQRY	Chapter 19, "IWMECQRY – Query Enclave Classification Attributes," on page 203
UnDoExportWorkUnit	IWMUEXPT	Chapter 58, "IWMUEXPT – WLM Undo Export Service," on page 483
UnDoImportWorkUnit	IWMUIMPT	Chapter 59, "IWMUIMPT – WLM Undo Import Service," on page 489
_server_classify	IWMCLSFY	
_server_classify_create	IWMCLSFY	
_server_classify_init	IWMCONN, IWMDISC	
_server_classify_pwu	IWMQINS, IWMSSEL, IWMSTEND, IWMESEQRY, IWMSREF, IWMSTBGN, IWMEDELE, IWMECREA, IWMCLSFY, IWMDISC	
_server_thread_query	IWMSINF	

---

## Interfaces for Sysplex Routing Services

The following table shows the C language interfaces that can be used to access WLM sysplex routing services.

**Note:** You need to include the header file IWMWDNSH before invoking these functions.

*Table 92. C Language Interfaces for WLM Sysplex Routing Services*

<b>C Language Interface</b>	<b>Associated WLM Sysplex Routing Service</b>	<b>Reference Chapter</b>
IWMDNREG	IWMSRARG	Chapter 56, "IWMSRARG – Register a Server for Sysplex Routing," on page 465
IWMDNDRG	IWMSRDRS	Chapter 54, "IWMSRDRS – Deregister a Server for Sysplex Routing," on page 451
IWMDNGRP	IWMSRDNS	Chapter 53, "IWMSRDNS – Get Sysplex Routing Location List," on page 445
IWMDNSRV	IWMSRSRS	Chapter 57, "IWMSRSRS – Determine Where to Route Work," on page 473

---

## Interface for Querying a Virtual Server

Products can use the query virtual server interface to obtain a virtual server's ID and capacity. The C interface for this query is IWMQVS, and the assembler interface is SYSEVENT QVS. Both forms of this query return a QVS structure which maps the returned identification and capacity information. See *z/OS MVS Programming: Authorized Assembler Services Reference SET-WTO* for more information.



---

## Appendix D. WLM Services supporting 31-bit address space only

---

### IWMAEDEF — Defining Dynamic Application Environments to Workload Management

The IWMAEDEF service defines dynamic application environments to WLM. The service can be used by queue manager address spaces to add new application environments after they connected to WLM and to delete the dynamic application environments before they disconnect from WLM.

Furthermore, the service can be used to define the method how server spaces should be resumed for static and dynamic application environments.

Before using this service, the caller must connect to WLM using the IWM4CON service, specifying `Work_Manager=Yes`, and `Queue_Manager=Yes`.

A queueing manager must not insert requests for a dynamic and static application environment with the same application environment name concurrently.

#### Note

It is recommended to use the new equivalent service IWM4AEDF introduced with z/OS V1R7. IWM4AEDF also supports 64-bit address space. For further information refer to Chapter 64, “IWM4AEDF — Defining Dynamic Application Environments to Workload Management,” on page 523.

## Environment

The requirements for the caller are:

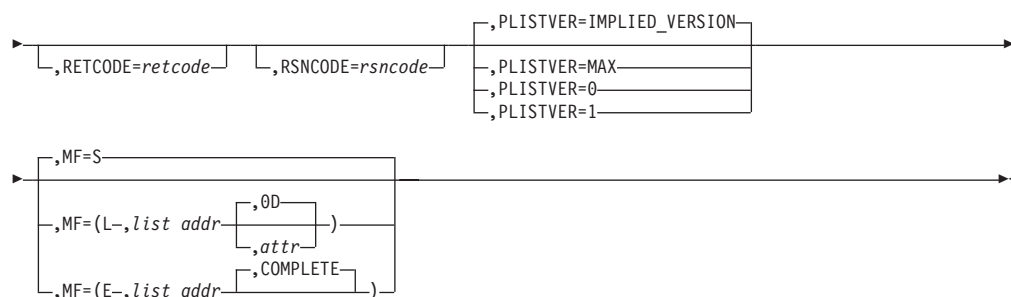
<b>Minimum authorization:</b>	Supervisor state or program key mask (PKM) allowing keys 0-7.
<b>Dispatchable unit mode:</b>	Task or SRB
<b>Cross memory mode:</b>	Any HASN, any SASN. PASN must be the address space which connected to WLM (i.e. the address space that was home when IWM4CON was issued for <code>Queue_Manager=Yes</code> , or <code>Server_Manager=Yes</code> with <code>Server_Type=Queue</code> ).
<b>AMODE:</b>	31-bit
<b>ASC mode:</b>	Primary
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks may be held.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

## Programming Requirements

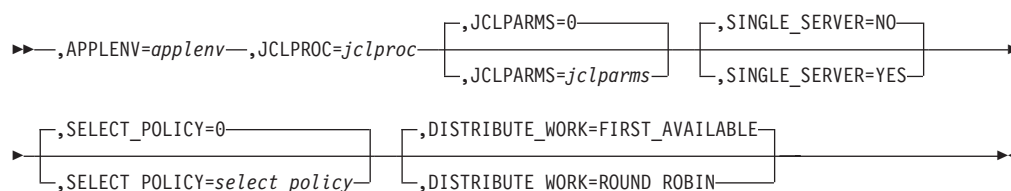
1. The macro CVT must be included to use this macro.
2. The macro IWMYCON must be included to use this macro.
3. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
4. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded







### parameters-1



## Parameters

The parameters are explained as follows:

### *name*

An optional symbol, starting in column 1, that is the name on the IWMAEDEF macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

### **,APPLENV=applenv**

When FUNC=ADD is specified, a required input parameter, which contains the name of the static or dynamic application environment.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32-character field.

### **CONNTKN=conntkn**

A required input parameter, which contains the connect token returned by the IWM4CON macro.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32 bit field.

### **,DISTRIBUTE\_WORK=FIRST\_AVAILABLE**

### **,DISTRIBUTE\_WORK=ROUND\_ROBIN**

When FUNC=ADD is specified, an optional parameter that controls how Workload Management resumes bound server spaces that are waiting for work. The default is DISTRIBUTE\_WORK=FIRST\_AVAILABLE.

### **,DISTRIBUTE\_WORK=FIRST\_AVAILABLE**

Workload Management wakes up the server space that has been suspended first (default).

### **,DISTRIBUTE\_WORK=ROUND\_ROBIN**

Workload Management wakes up the server space that has the smallest number of affinities. If there are several server spaces with the same number of affinities, workload management will start the server space with the smallest number of active server tasks.

### **,FUNC=ADD**

**,FUNC=DELETE**

**,FUNC=MODIFY**

A required parameter that indicates how the caller uses the service

**,FUNC=ADD**

indicates that the caller wants to add a dynamic application environment to WLM.

**,FUNC=DELETE**

indicates that the caller wants to delete its interest in the dynamic application environment.

**,FUNC=MODIFY**

indicates that the caller wants to redefine the method how server spaces should be resumed for static and dynamic application environments.

**,JCLPARMS=*jclparms***

**,JCLPARMS=0**

When **FUNC=ADD** is specified, an optional input parameter, which contains the parameters which are passed to the start procedure of the server manager address spaces by WLM. The default is 0.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 115-character field.

**,JCLPROC=*jclproc***

When **FUNC=ADD** is specified, a required input parameter, which contains the name of the start procedure which is used by WLM to start server manager address spaces for the application environment.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

**,MF=S**

**,MF=(L,*list addr*)**

**,MF=(L,*list addr,attr*)**

**,MF=(L,*list addr,OD*)**

**,MF=(E,*list addr*)**

**,MF=(E,*list addr,COMPLETE*)**

An optional input parameter that specifies the macro form.

Use **MF=S** to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. **MF=S** is the default.

Use **MF=L** to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the **PLISTVER** parameter may be coded with the list form of the macro.

Use **MF=E** to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

**,*list addr***

The name of a storage area to contain the parameters. For **MF=S** and **MF=E**, this can be an RS-type address or an address in register (1)-(12).

*,attr*

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

**,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

**,PLISTVER=IMPLIED\_VERSION**

**,PLISTVER=MAX**

**,PLISTVER=0**

**,PLISTVER=1**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form, when both are assembled with the same level of the system. In this way, MAX ensures that the parameter list does not overwrite nearby storage.

- **0**, which supports all parameters except those specifically referenced in higher versions.

**1**, which supports both the following parameters and those from version 0:

DISTRIBUTE\_WORK            STATIC

**To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX
- A decimal value of 0, or 1

**,RETCODE=*retcode***

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,RSNCODE=*rsncode***

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,SELECT\_POLICY=*select\_policy***

**,SELECT\_POLICY=0**

When FUNC=ADD is specified, an optional input parameter, which tells WLM

## IWMAEDEF Macro

how to select work if work requests are directly routed to the server address space. Only 0,1 and 2 are valid select policies. 0 is the default which is also selected if an invalid policy is specified.

The select policy options 0,1 and 2 have the following meaning:

- 0 Default, the oldest request on either the service class or server address space queue is selected first.
- 1 The request on the server address space queue (if present) is selected first independently of the times the requests have been inserted.
- 2 The request on the service class queue is always selected first.

The default is 0.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an one-byte field.

**,SINGLE\_SERVER=NO**

**,SINGLE\_SERVER=YES**

When FUNC=ADD is specified, an optional parameter indicating whether one or multiple server spaces should be started for the application environment. The default is SINGLE\_SERVER=NO.

**,SINGLE\_SERVER=NO**

Multiple server spaces should be started for the application environment (default).

**,SINGLE\_SERVER=YES**

Only one server space should be started for the application environment.

**,STATIC=NO**

**,STATIC=YES**

When FUNC=MODIFY is specified, an optional parameter that controls whether a static or dynamic application environment should be updated. The default is STATIC=NO.

**,STATIC=NO**

indicates that the caller wants to modify a dynamic application environment (default).

**,STATIC=YES**

indicates that the caller wants to modify a static application environment.

## ABEND Codes

None.

## Return and Reason Codes

When the IWMAEDEF macro returns control to your program:

- GPR 15 (and *retcode*, when you code RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, when you code RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 93. Return and Reason Codes for the IWMAEDEF Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> lwmRetCodeOk <b>Meaning:</b> Successful completion. <b>Action:</b> None required.
8	—	<b>Equate Symbol:</b> lwmRetCodeInvocError <b>Meaning:</b> Invalid invocation environment or parameters.
8	xxxx0803	<b>Equate Symbol:</b> lwmRsnCodeDisabled <b>Meaning:</b> Caller is disabled. <b>Action:</b> Avoid requesting this function while disabled.
8	xxxx0804	<b>Equate Symbol:</b> lwmRsnCodeLocked <b>Meaning:</b> Caller is locked. <b>Action:</b> Avoid requesting this function while locked.
8	xxxx080B	<b>Equate Symbol:</b> lwmRsnCodeBadPI <b>Meaning:</b> Error accessing parameter list. <b>Action:</b> Check for possible storage overlay.
8	xxxx0821	<b>Equate Symbol:</b> lwmRsnCodeBadConn <b>Meaning:</b> Input connect token does not pass validity checking. <b>Action:</b> Make sure to use the connect token returned by the IWM4CON service requesting Queue_Manager=Yes, or Server_Manager=Yes with Server_Type=Queue.
8	xxxx0823	<b>Equate Symbol:</b> lwmRsnCodeDatoff <b>Meaning:</b> Caller invoked service while DATOFF <b>Action:</b> Avoid requesting this function in this environment.
8	xxxx0824	<b>Equate Symbol:</b> lwmRsnCodeAmode24 <b>Meaning:</b> Caller invoked service but was in 24 bit addressing mode. <b>Action:</b> Request this function only when you are in 31 bit addressing mode.
8	xxxx0825	<b>Equate Symbol:</b> lwmRsnCodeAscModeNotPrimary <b>Meaning:</b> Caller invoked service but was not in primary ASC mode. <b>Action:</b> Avoid requesting this function in this environment.
8	xxxx0827	<b>Equate Symbol:</b> lwmRsnCodeRsvdNot0 <b>Meaning:</b> Reserved field in parameter list was non-zero. <b>Action:</b> Check for use of keywords that are not supported by the MVS release on which the program is running.
8	xxxx0828	<b>Equate Symbol:</b> lwmRsnCodeBadVersion <b>Meaning:</b> Version number in parameter list is not valid. <b>Action:</b> Check for possible storage overlay of the parameter list.

## IWMAEDEF Macro

Table 93. Return and Reason Codes for the IWMAEDEF Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx083F	<p><b>Equate Symbol:</b> IwmRsnCodePrimaryNotOwnConn</p> <p><b>Meaning:</b> Primary address space does not own the passed connect token.</p> <p><b>Action:</b> Ensure that the primary address space has previously connected to WLM using the IWM4CON macro. Ensure that the connect token returned by the IWM4CON macro is passed to the IWMAEDEF macro.</p>
8	xxxx0840	<p><b>Equate Symbol:</b> IwmRsnCodeServiceNotEnabled</p> <p><b>Meaning:</b> Caller's space connection is not enabled for this service</p> <p><b>Action:</b> Make sure that Queue_Manager=Yes, or Server_Manager=Yes with Server_Type=Queue is specified on the IWM4CON request to enable this service.</p>
8	xxxx0842	<p><b>Equate Symbol:</b> IwmRsnCodeNoWLMConnect</p> <p><b>Meaning:</b> Caller's space disconnected from WLM during processing of the request.</p> <p><b>Action:</b> None.</p>
8	xxxx0890	<p><b>Equate Symbol:</b> IwmRsnCodeApplEnvExists</p> <p><b>Meaning:</b> The caller tried to add an application environment that has already been defined. subsystem type.</p> <p><b>Action:</b> Check whether the correct application environment name is being used. Make sure that a unique application environment name is used when adding application environments.</p>
8	xxxx0891	<p><b>Equate Symbol:</b> IwmRsnCodeApplEnvNotFound</p> <p><b>Meaning:</b> The caller tried to delete or modify an application environment that does not exist.</p> <p><b>Action:</b> Check whether the correct application environment name is being used.</p>
C	—	<p><b>Equate Symbol:</b> IwmRetCodeEnvError</p> <p><b>Meaning:</b> Environmental error.</p>
C	xxxx0C01	<p><b>Equate Symbol:</b> IwmRsnCodeNoStg</p> <p><b>Meaning:</b> Storage is not available for the request.</p> <p><b>Action:</b> Contact your system programmer. There is a common storage shortage.</p>
10	—	<p><b>Equate Symbol:</b> IwmRetCodeCompError</p> <p><b>Meaning:</b> Component error.</p> <p><b>Action:</b> Contact your system programmer.</p>

## Example

To add a dynamic application environment:

```

IWMAEDEF CONNTKN=CONNTOKEN,           X
        FUNC=ADD,                       X
        APPLENV=APPLENV                 X
        JCLPROC=JCLPROC                 X
        RETCODE=RC,RSNCODE=RSN

```

```

*
* Storage areas
*
CONNTOKEN DS   FL4           Contains the connect token
*                               associated with the use of WLM
*                               Queuing services as returned by
*                               IWM4CON
*                               IWM4CON
APPLENV   DS   CL32          Contains the application
*                               environment name
JCLPROC   DS   CL8           Contains the name of the
*                               start procedure
RC         DS   F            Return code
RSN        DS   F            Reason code

```

---

## IWMCONN – Connecting to Workload Management

The purpose of this service is to connect a calling address space to WLM. This service returns a token which is needed to invoke other services. This service can be used to:

- Request that WLM Work Management services be available to the connecting address space and optionally to pass topology information to WLM.
- Request that WLM Work Queuing services be available to the connecting address space.
- Request that WLM Work Execution services be available to the connecting address space.
- Request that WLM Work Balancing services be available to the connecting address space.
- Request that WLM export and import services be available to the connecting address space.

Note that:

- The space which is connected is the current home address space.
- Only a single connection is allowed to be active for a given address space at any given time.
- For each connected task/space, WLM will establish a dynamic resource manager (RESMGR) to be associated with the current task/space. When it receives control, it will free any accumulated resources and delete any enclaves associated with the connect token. This implies that the resource manager will logically perform the disconnect function and the connect token can no longer be passed to WLM services.

### Note

It is recommended to use the new equivalent service IWM4CON introduced with z/OS V1R6. IWM4CON also supports 64-bit address space. For further information refer to Chapter 65, “IWM4CON – Connecting to Workload Management,” on page 533.

## Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	For WORK_MANAGER=YES or ROUTER=YES, QUEUE_MANAGER=YES or EXPTIMPT=YES, supervisor state or program key mask (PKM) allowing keys 0-7.  For SERVER_MANAGER=YES, problem state with any PSW key.
<b>Dispatchable unit mode:</b>	Task
<b>Cross memory mode:</b>	Non-XMEM when input key is a user key or SERVER_MANAGER = YES, otherwise XMEM, any P,S,H.
<b>AMODE:</b>	31-bit
<b>ASC mode:</b>	Primary
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks may be held.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

## Programming Requirements

1. Make sure no EUT FRRs are established.
2. The macro CVT must be included to use this macro.
3. The macro IWMYCON must be included to use this macro.
4. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
5. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, IWMRSCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.
6. All character data, unless otherwise specified, is assumed to be left justified and padded with blanks on the right, as needed, to occupy the specified number of bytes.

## Restrictions

1. This macro may only be used on MVS/ESA (version 3 or higher), NOT versions 1 or 2 of MVS.
2. This macro may not be used during task/address space termination.
3. Only a single connection is allowed to be active for a given address space at any given time.
4. Specification of both Queue\_Manager=Yes, and Server\_Manager=Yes requires that Server\_Type=Queue. Specification of Server\_Type=Routing is rejected.
5. Specification of both Router=Yes, and Server\_Manager=Yes requires that Server\_Type=Routing. Specification of Server\_Type=Queue is rejected.
6. If the callers recovery routine should get control as a result of requesting this service, the function cannot be guaranteed to be complete. It is possible that a token has been saved in the parameter list where the connect token would reside upon successful completion. This token may be passed to IWMDISC to prevent the address space from being disabled from future IWMCONN requests,



but the token should not be used for other services. IWMDISC in these circumstances may give a warning return code indicating that no connection was established, however.

7. If the key specified on IWMCONN is a user key (8-F) or SERVER\_MANAGER=YES was specified, then the caller must be in non-cross-memory mode (P=S=H)
8. While not a restriction for IWMCONN, it should be noted that when the key specified is a user key (8-F), the Connect token may only be passed to IWMCLSFY, IWMRPT, or IWMMNTFY services, when the current primary matches primary at the time IWMCONN is invoked.

## Input Register Information

Before issuing the IWMCONN macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

## Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Reason code if GR15 return code is non-zero
1	Used as work registers by the system
2-13	Unchanged
14	Used as work registers by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

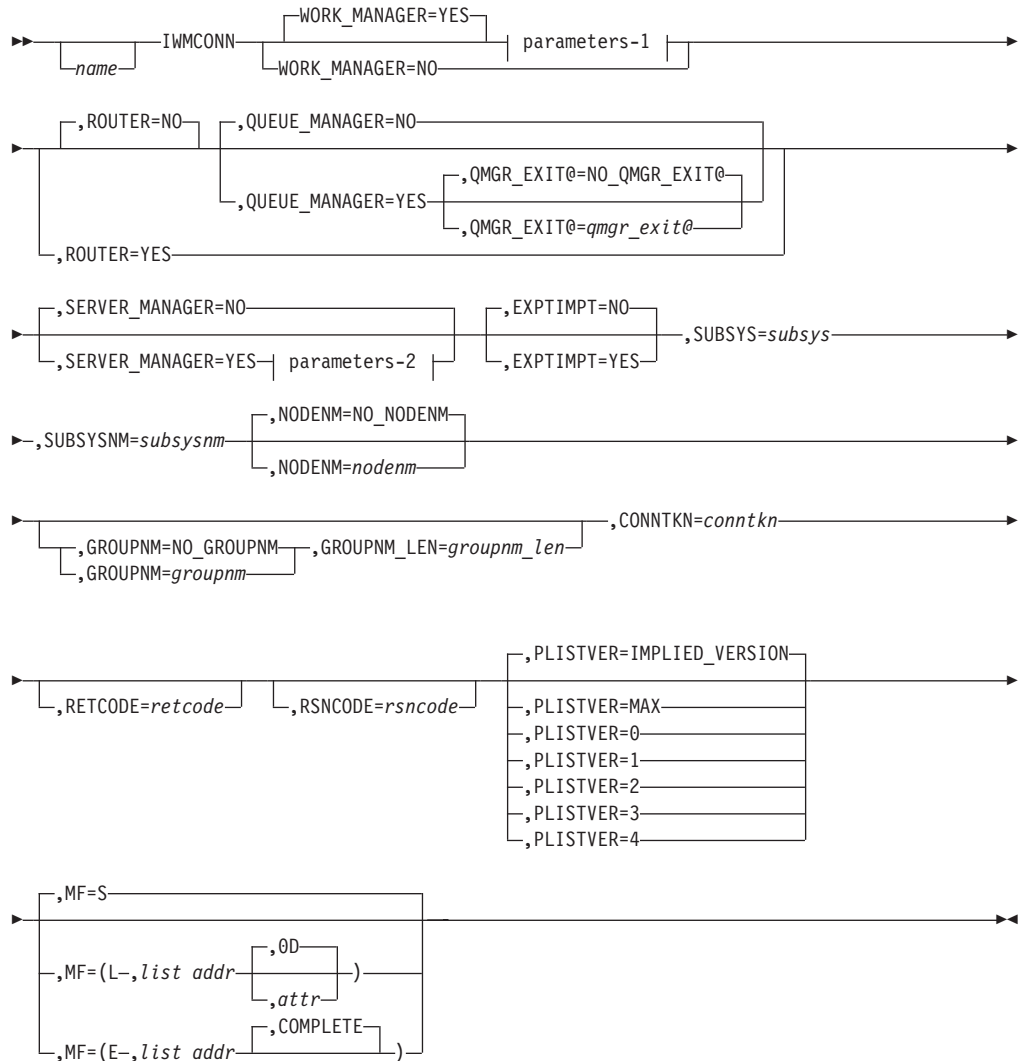
Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

## Performance Implications

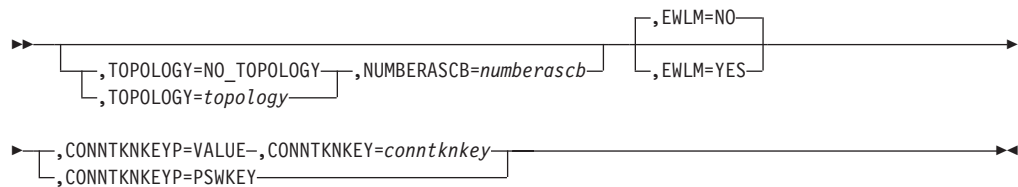
None.

# Syntax

## main diagram

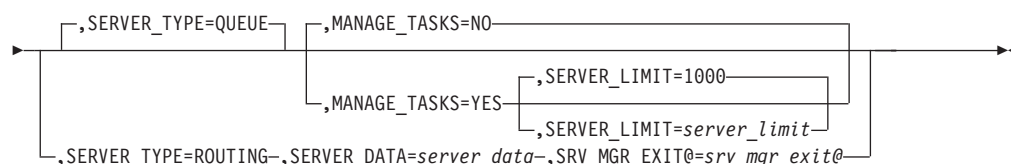


## parameters-1



## parameters-2





## Parameters

The parameters are explained as follows:

### *name*

An optional symbol, starting in column 1, that is the name on the IWMCONN macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

### **,APPLENV=applenv**

When SERVER\_MANAGER=YES is specified, a required input parameter, which contains the application environment under which work requests are served.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32-character field.

### **,CONNTKN=conntkn**

A required output parameter, which will receive the connect token.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32 bit field.

### **,CONNTKNKEY=conntknkey**

When CONNTKNKEYP=VALUE and WORK\_MANAGER=YES are specified, a required input parameter, which contains the key for which the various branch entry services using the CONNTKN returned by lwmmconn must have PSW update authority. These other services include Classify (lwmmclsfy), Report (lwmmrpt), Notify (lwmmntfy). Create (lwmmcrea) is a PC interface and hence is excluded. The low order 4 bits (bits 4-7) contain the key value. The high order 4 bits (bits 0-3) must be zeros.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8 bit field.

### **,CONNTKNKEYP=VALUE**

### **,CONNTKNKEYP=PSWKEY**

When WORK\_MANAGER=YES is specified, a required parameter, which describes how the input key should be obtained.

### **,CONNTKNKEYP=VALUE**

indicates that the key is being passed explicitly via CONNTKNKEY.

### **,CONNTKNKEYP=PSWKEY**

indicates that the current PSW key should be used.

### **,DYNAMIC=NO**

### **,DYNAMIC=YES**

When SERVER\_MANAGER=YES is specified, an optional parameter indicating whether the server manager connects to a dynamic or static application environment. The default is DYNAMIC=NO.

### **,DYNAMIC=NO**

The server manager connects to a static application environment. This is the default.

**,DYNAMIC=YES**

The server manager connects to a dynamic application environment.

**,EWLM=NO**

**,EWLM=YES**

When WORK\_MANAGER=YES is specified, an optional parameter, which indicates if this work manager intends to participate in cross-platform enterprise workload management (EWLM). The default is EWLM=NO.

**,EWLM=NO**

The work manager interacts only with WLM and no interaction with EWLM takes place. This is the default.

**,EWLM=YES**

The work manager participates in cross-platform enterprise workload management and interacts with EWLM.

**,EXPTIMPT=NO**

**,EXPTIMPT=YES**

An optional parameter indicating whether the space needs access to the export and import services (IWMEXPT, IWMUEXPT, IWMIMPT, IWMUIMPT). The default is EXPTIMPT=NO.

**,EXPTIMPT=NO**

The connecting address space will not use the export and import services.

**,EXPTIMPT=YES**

The connecting address space will use the export and import services.

**,GROUPNM=*groupnm***

**,GROUPNM=NO\_GROUPNM**

An optional input parameter, which contains the name of an application group, for example, a group of similar or cooperating subsystem instances. A group name can be up to 255 characters long. Provision of a data area initialized to all blanks is equivalent to specification of NO\_GROUPNM. The default is NO\_GROUPNM. This indicates that no group name is passed.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a character field.

**,GROUPNM\_LEN=*groupnm\_len***

When GROUPNM=*groupnm* is specified, a required input parameter, which contains the length of the group name. A group name can be up to 255 characters long.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

**,MANAGE\_TASKS=NO**

**,MANAGE\_TASKS=YES**

When SERVER\_TYPE=QUEUE and SERVER\_MANAGER=YES are specified, an optional parameter indicating that WLM will manage the server instances (tasks), selecting work from a work queue.

If YES is specified the caller must use service IWMSINF to obtain the number of server instances to start from WLM.

The meaning of PARALLEL\_EU changes in this case. PARALLEL\_EU is only used to determine the number of tasks to start if the application environment cannot be managed by WLM. Other wise PARALLEL\_EU can be used to limit the number of server tasks to start initially.

The server can define the SERVER\_LIMIT parameter to specify a limit for the number of server tasks supported by the application.

**,MANAGE\_TASKS=NO**

The connecting address space starts the number of server instances as provided with parallel\_eu.

**,MANAGE\_TASKS=YES**

The connecting address space uses IWMSINF to obtain the number of server instances to start from WLM.

**,MF=S**

**,MF=(L,list addr)**

**,MF=(L,list addr,attr)**

**,MF=(L,list addr,0D)**

**,MF=(E,list addr)**

**,MF=(E,list addr,COMPLETE)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

*,list addr*

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

*,attr*

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

**,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

**,NODENM=nodenm**

**,NODENM=NO\_NODENM**

An optional input parameter, which contains the node name to be used for classifying work requests when Work\_Manager=Yes is specified or taken as default. The node name identifies a specific subcomponent of the generic subsystem type.

When Server\_Manager=Yes and Server\_Type=Queue is specified, the node name should match the node name specified on the corresponding Connect for the Queue\_Manager, for example, all servers associated with the Queue\_Manager have identical node names.

## IWMCONN Macro

If a product chooses to use both `Work_Manager=Yes` and `Server_Manager=Yes` on a single invocation of IWMCONN for a space, then the rules for `Server_Manager` apply, for example, the node name refers to the node name of the space playing the role of `Queue_Manager`.

If the caller connects to the WLM work queueing services, the combination of the subsystem type, node name and the subsystem name must be unique to that MVS system. Node name can be omitted. The default is `NO_NODENM`.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

### **,NUMBERASCB=*numberascb***

When `TOPOLOGY=topology` and `WORK_MANAGER=YES` are specified, a required input parameter, which contains the number of ASCBs in the list passed via `xTOPOLOGY`. While there is no restriction on the number of entries in the list, the current support will only look at the first 10 entries. The number specified must be positive (hence also non-zero).

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

### **,PARALLEL\_EU=*parallel\_eu***

When `SERVER_MANAGER=YES` is specified, a required input parameter, which contains the maximum number of tasks (TCBs) within the address space which will be used to concurrently process distinct work requests if `MANAGE_TASKS=YES` is not in effect. When `Select (Iwmssel)` is used to obtain a work request, which might then be passed to another task (TCB) for processing under a `Begin (Iwmstbgn)` environment, this count represents the number of tasks (TCBs) which can be running concurrently against these work requests, i.e. the number of concurrent `Begin` environments. It is important that this count represent the actual number of tasks (TCBs) which can be utilized, and not merely some approximate upper bound, as this value will influence system algorithms.

If `MANAGE_TASKS=YES` is in effect, the application environment managed by WLM `PARALLEL_EU` is not used. In this case the parameter is only used as described above if no procedure name was defined for the application environment.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

### **,PLISTVER=IMPLIED\_VERSION**

#### **,PLISTVER=MAX**

#### **,PLISTVER=0**

#### **,PLISTVER=1,PLISTVER=2**

#### **,PLISTVER=3**

#### **,PLISTVER=4**

An optional input parameter that specifies the version of the macro. `PLISTVER` determines which parameter list the system generates. `PLISTVER` is an optional input parameter on all forms of the macro, including the list form. When using `PLISTVER`, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the `PLISTVER` parameter, `IMPLIED_VERSION` is the default.

- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify `PLISTVER=MAX` on the list form of the macro. Specifying `MAX` ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form, when both are assembled with the same level of the system. In this way, `MAX` ensures that the parameter list does not overwrite nearby storage.

- **0**, if you use the currently available parameters.
- **1**, which supports the following parameters along with the parameters from version 0:

APPLENV	QUEUE_MANAGER	SERVER_TYPE
EXPTIMPT	ROUTER	SRV_MGR_EXIT@
PARALLEL_EU	SERVER_DATA	WORK_MANAGER
QMGR_EXIT@	SERVER_MANAGER	

- **2**, which supports the following parameters and those from version 0 and 1:

MANAGE_TASKS	SERVER_LIMIT
--------------	--------------

- **3**, which supports both the following parameters and those from version 0, 1, and 2:

DYNAMIC	NODENM
---------	--------

- **4**, which supports both the following parameters and those from version 0, 1, 2, and 3:

EWLM	GROUPNM	GROUPNM_LEN
------	---------	-------------

**To code:** Specify one of the following:

- `IMPLIED_VERSION`
- `MAX`
- A decimal value of 0, 1, 2, 3, or 4

**,QMGR\_EXIT@=qmgr\_exit@**

**,QMGR\_EXIT@=NO\_QMGR\_EXIT@**

When `QUEUE_MANAGER=YES` and `ROUTER=NO` are specified, an optional input parameter that contains the address of the Queue Manager Connect Exit to be invoked when the system wishes to inform the queue manager of actions it should perform. The exit will be called in enabled, unlocked TCB mode with no FRRs set, but may be called in a cross-memory environment. The mapping of the parameter list for the exit and its invocation environment is given by the list form of the `IWMQCXIT` macro.

The system may choose to discontinue calling the exit upon repetitive abnormal completions, i.e. where the system recovery routine is percolated to from an error within the exit. The exit must be callable from any address space and remain available after the queue manager disconnects or terminates. The default is `NO_QMGR_EXIT@`, which indicates that no queue manager exit is provided.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a pointer field.

**,QUEUE\_MANAGER=NO**

### **,QUEUE\_MANAGER=YES**

When ROUTER=NO is specified, an optional parameter indicating that WLM Work Queuing services be available to the connecting address space. For example:

- Insert (Iwmqins)
- Delete (Iwmqdel)

If YES is specified, the combination of the subsystem type and the subsystem name must be unique to that MVS system. The default is QUEUE\_MANAGER=NO.

### **,QUEUE\_MANAGER=NO**

The connecting address space will not use the WLM Work Queuing services.

### **,QUEUE\_MANAGER=YES**

The connecting address space will be using the WLM Work Queuing services.

### **,RETCODE=*retcode***

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

### **,ROUTER=NO**

### **,ROUTER=YES**

An optional parameter, which describes whether recommendations for sysplex routing to servers associated with the same subsystem type and name are requested. The default is ROUTER=NO.

### **,ROUTER=NO**

indicates that recommendations for sysplex routing via IWMSRFSV are not required.

### **,ROUTER=YES**

indicates that recommendations for sysplex routing via IWMSRFSV is required. Note that only server spaces which have the same Subsystem type and name AND which specified Server\_Type=Routing are considered when IWMSRFSV is invoked.

If YES is specified, the combination of the subsystem type and the subsystem name must be unique to that MVS system.

### **,RSNCODE=*rsncode***

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

### **,SERVER\_DATA=*server\_data***

When SERVER\_TYPE=ROUTING and SERVER\_MANAGER=YES are specified, a required input parameter, which contains whatever data is needed to uniquely identify the server when recommended by MVS through use of the IWMSRFSV interface. The structure of this data is undefined to MVS, and will be returned to the program invoking IWMSRFSV when the server is returned.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32-character field.

### **,SERVER\_LIMIT=*server\_limit***

### **,SERVER\_LIMIT=1000**

When MANAGE\_TASKS=YES, SERVER\_TYPE=QUEUE and



SERVER\_MANAGER=YES are specified, an optional input parameter indicating the architectural limit of the application for the number of server instances which can be supported.

This parameter can be used to tell WLM the upper limit up to which WLM will recommend to start server instances. If the parameter is omitted or is set higher than 1000, WLM will use 1000 as upper limit instead. The default is 1000.

**To code:** Specify the RS-type address of a fullword field.

**,SERVER\_MANAGER=NO**

**,SERVER\_MANAGER=YES**

An optional parameter indicating whether the space needs access to a family of services specified by SERVER\_TYPE. The default is SERVER\_MANAGER=NO.

**,SERVER\_MANAGER=NO**

The connecting address space will not use any of the various server related WLM services documented under SERVER\_TYPE.

**,SERVER\_MANAGER=YES**

The connecting address space will be acting in the role of a server and needs access to the family of services specified by SERVER\_TYPE.

Specification of both Queue\_Manager=Yes, and Server\_Manager=Yes requires that Server\_Type=Queue. Specification of Server\_Type=Routing is rejected.

Specification of both Router=Yes, and Server\_Manager=Yes requires that Server\_Type=Routing. Specification of Server\_Type=Queue is rejected.

**,SERVER\_TYPE=QUEUE**

**,SERVER\_TYPE=ROUTING**

When SERVER\_MANAGER=YES is specified, an optional parameter, which describes what type of services are used by the server. The default is SERVER\_TYPE=QUEUE.

**,SERVER\_TYPE=QUEUE**

indicates that the server selects work from a queue, and thus requests that WLM Work Execution services be available to the connecting address space. For example:

- Select (Iwmssel)
- Begin (Iwmstbgn)
- End (Iwmstend)

The server also has the WLM Work Queuing services available to the connecting address space when the corresponding Queue Manager with the same subsystem type and name is active on the same MVS image (see following macros for macro specific restrictions). For example:

- Insert (Iwmqins)
- Delete (Iwmqdel)

**,SERVER\_TYPE=ROUTING**

indicates that the server receives work by way of routing, and may be selected by the IWMSRFSV (Find Server) macro interface. Note that the space which invokes the IWMSRFSV service must Connect with Router=Yes.

Termination of the router with the same subsystem type and name on the same MVS image will not cause notification to the server to terminate. This coordination, if required, must be handled through a different protocol than use of Connect.

### **,SRV\_MGR\_EXIT@=***srv\_mgr\_exit@*

When `SERVER_TYPE=ROUTING` and `SERVER_MANAGER=YES` are specified, a required input parameter that is to contain the address of the Server Manager Connect Exit to be invoked when the system wishes to inform the server of actions it should perform. This exit will be called in SRB mode, with a non cross-memory environment, where `HASN=SASN=PASN=HASN` at the time IWMCONN was invoked. The mapping of the parameter list for the exit and its invocation environment is given by the list form of the IWMSCXIT macro.

Note that it may be possible for the exit to be called before the caller has received control back from IWMCONN. The exit or any program it drives (synchronously or asynchronously) must synchronize with the program issuing IWMCONN to ensure that IWMCONN has returned a connect token prior to issuing IWMDISC (disconnect) or any other services that need the connect token.

The system may cause the space to become ineligible to be recommended by IWMSRFSV upon repetitive errors in calling the exit specified. The exit must be callable from the server address space and remain available after the server manager disconnects or the connecting server TCB terminates. The exit need not persist upon memory termination of the server.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a pointer field.

### **,SUBSYS=***subsys*

A required input parameter, which contains the generic subsystem type (e.g. IMS, CICS, etc.). When `WORK_MANAGER=YES` is specified, this is the primary category under which classification rules are grouped.

If the caller connects to the WLM work queueing services by specifying `QUEUE_MANAGER=YES`, or requests sysplex routing by specifying `ROUTER=YES`, the combination of the subsystem type and the subsystem name must be unique to that MVS system.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 4-character field.

### **,SUBSYSNM=***subsysnm*

A required input parameter, which contains the subsystem name to be used for classifying work requests when `Work_Manager=Yes` is specified or taken as default. The subsystem name identifies a specific instance of the generic subsystem type.

When `Server_Manager=Yes` and `Server_Type=Queue` is specified, the subsystem name should match the subsystem name specified on the corresponding Connect for the Queue\_Manager, i.e. all servers associated with the Queue\_Manager have identical subsystem names.

When `Server_Manager=Yes` and `Server_Type=Routing` is specified, the subsystem name should match the subsystem name specified on the corresponding Connect for Router=Yes, i.e. all servers associated with the Router have identical subsystem names.

If a product chooses to use both `Work_Manager=Yes` and `Server_Manager=Yes` on a single invocation of IWMCONN for a space, then the rules for `Server_Manager` apply, i.e. the subsystem name refers to the subsystem name of the space playing the role of Queue\_Manager or Router.

If the caller connects to the WLM work queuing services, or to sysplex routing services, the combination of the subsystem type and the subsystem name must be unique to that MVS system.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

**,TOPOLOGY=*topology***

**,TOPOLOGY=NO\_TOPOLOGY**

When **WORK\_MANAGER=YES** is specified, an optional input parameter, which represents a list of ASCB addresses for the address spaces which comprise the subsystem. This list should **ONLY** include address spaces which do **NOT** surface as the current home address space when `lwmminit` or `lwmrela` are used to establish the delay monitoring environments, but that may participate as dispatchable units (TCBs or SRBs) in serving work requests. If the current primary or home space is a space not surfacing in a monitoring environment and its execution can affect the response time of work flowing through the subsystem, then it should appear in the list. Neither current primary nor current home are defaults. While there are no limits on the number of address spaces, this information is less precise than that provided by monitoring environments. The default is **NO\_TOPOLOGY**, which indicates that no topology information was passed.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a character field.

**WORK\_MANAGER=YES**

**WORK\_MANAGER=NO**

An optional parameter indicating that WLM Work Management services be available to the connecting address space. For example:

- Classify (`lwmclsfy`)
- Report (`lwmrpt`)
- Notify (`lwmmtfy`)
- Enclave Create (`lwmecrea`)
- Modify Connect (`lwmwmcon`)

If **NO** is specified, the above services cannot be used, except for the form of Notify that does not pass an input connect token. The default is **WORK\_MANAGER=YES**.

**WORK\_MANAGER=YES**

The connecting address space will be using the WLM Work Management services.

**WORK\_MANAGER=NO**

The connecting address space will not use the WLM Work Management services. Specifying this keyword may reduce the use of system resources.

## ABEND Codes

None.

## Return and Reason Codes

When the IWMCONN macro returns control to your program:

- GPR 15 (and *retcode*, if you coded `RETCODE`) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, if you coded `RSNCODE`) contains reason code.

## IWMCONN Macro

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 94. Return and Reason Codes for the IWMCONN Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> IwmRetCodeOk <b>Meaning:</b> Successful completion. <b>Action:</b> None required.
4	—	<b>Equate Symbol:</b> IwmRetCodeWarning <b>Meaning:</b> Successful completion, unusual conditions noted.
4	xxxx0401	<b>Equate Symbol:</b> IwmRsnCodeNoWLM <b>Meaning:</b> System does not support work manager services. This return code is only set when the MVS release is prior to MVS/ESA SP5.1.0 or prior to HBB6603 for IWMCONN QUEUE_MANAGER/SERVER_MANAGER services. <b>Action:</b> None required.
8	—	<b>Equate Symbol:</b> IwmRetCodeInvocError <b>Meaning:</b> Invalid invocation environment or parameters.
8	xxxx0801	<b>Equate Symbol:</b> IwmRsnCodeSrbMode <b>Meaning:</b> Caller is in SRB mode. <b>Action:</b> Avoid requesting this function while in SRB mode.
8	xxxx0802	<b>Equate Symbol:</b> IwmRsnCodeXmemUserKeyTkn <b>Meaning:</b> Caller is in cross-memory mode while the token was requested in user key. <b>Action:</b> Avoid requesting this function while in cross-memory mode.
8	xxxx0803	<b>Equate Symbol:</b> IwmRsnCodeDisabled <b>Meaning:</b> Caller is disabled. <b>Action:</b> Avoid requesting this function while disabled.
8	xxxx0804	<b>Equate Symbol:</b> IwmRsnCodeLocked <b>Meaning:</b> Caller is locked. <b>Action:</b> Avoid requesting this function while locked.
8	xxxx080B	<b>Equate Symbol:</b> IwmRsnCodeBadPI <b>Meaning:</b> Error accessing parameter list. <b>Action:</b> Check for possible storage overlay. Also check if you call this macro in 64-bit address mode. Refer to the description of reason code xxxx089E for further information.
8	xxxx0810	<b>Equate Symbol:</b> IwmRsnCodeEutFrr <b>Meaning:</b> Caller has EUT FRR established. <b>Action:</b> Avoid requesting this function with an EUT FRR set.
8	xxxx0812	<b>Equate Symbol:</b> IwmRsnCodeBadAscb <b>Meaning:</b> Bad ASCB address passed. <b>Action:</b> Check for possible storage overlay.

Table 94. Return and Reason Codes for the IWMCONN Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0823	<b>Equate Symbol:</b> lwmRsnCodeDatoff <b>Meaning:</b> Caller invoked service while DATOFF <b>Action:</b> Avoid requesting this function in this environment.
8	xxxx0824	<b>Equate Symbol:</b> lwmRsnCodeAmode24 <b>Meaning:</b> Caller invoked service but was in 24 bit addressing mode. <b>Action:</b> Request this function only when you are in 31 bit addressing mode.
8	xxxx0825	<b>Equate Symbol:</b> lwmRsnCodeAscModeNotPrimary <b>Meaning:</b> Caller invoked service but was not DAT on Primary ASC mode. <b>Action:</b> Avoid requesting this function in this environment.
8	xxxx0826	<b>Equate Symbol:</b> lwmRsnCodeTaskTerm <b>Meaning:</b> Caller invoked service while task termination is in progress for the TCB associated with the owner. <b>Action:</b> Avoid requesting this function in this environment.
8	xxxx0827	<b>Equate Symbol:</b> lwmRsnCodeRsvdNot0 <b>Meaning:</b> Reserved field in parameter list was non-zero. <b>Action:</b> Check for possible storage overlay of the parameter list.
8	xxxx0828	<b>Equate Symbol:</b> lwmRsnCodeBadVersion <b>Meaning:</b> Version number in parameter list is not valid. <b>Action:</b> Check for possible storage overlay of the parameter list.
8	xxxx0829	<b>Equate Symbol:</b> lwmRsnCodeBadOptions <b>Meaning:</b> Parameter list omits required parameters or supplies mutually exclusive parameters or provides data associated with options not selected. <b>Action:</b> Check for possible storage overlay of the parameter list.
8	xxxx082C	<b>Equate Symbol:</b> lwmRsnCodeBadNumberAscb <b>Meaning:</b> NUMBERASCB variable is not a positive value. <b>Action:</b> Check for possible storage overlay of the parameter list or variable.
8	xxxx082E	<b>Equate Symbol:</b> lwmRsnCodeConnectExists <b>Meaning:</b> Connect has already been established for the current home address space. <b>Action:</b> Avoid requesting this function when a connection already exists.

Table 94. Return and Reason Codes for the IWMCONN Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0840	<p><b>Equate Symbol:</b> IwmRsnCodeServiceNotEnabled</p> <p><b>Meaning:</b> Requested connection type cannot be established in the current execution environment. This occurs when SERVER_MANAGER=YES is specified and the program is run as a batch job in a WLM-managed job class.</p> <p><b>Action:</b> Run the program as a started task.</p>
8	xxxx0841	<p><b>Equate Symbol:</b> IwmRsnCodeXMemMode</p> <p><b>Meaning:</b> Caller is in cross memory mode.</p> <p><b>Action:</b> Invoke the function in non-cross memory mode.</p>
8	xxxx0847	<p><b>Equate Symbol:</b> IwmRsnCodeOtherSpaceConnected</p> <p><b>Meaning:</b> Another address space with the same subsystem type and name is connected to WLM on the MVS image and has the role of queue manager or router.</p> <p><b>Action:</b> Avoid requesting this function with duplicate values.</p>
8	xxxx0849	<p><b>Equate Symbol:</b> IwmRsnCodeWLMServBadAPPL</p> <p><b>Meaning:</b> The application environment name (APPLENV=) specified is not the same as the one used by WLM to start the server.</p> <p><b>Action:</b> Verify that the start parameters for the application environment are coded correctly in the WLM ISPF application, and that those parameters are used by the started JCL procedure.</p>
8	xxxx084A	<p><b>Equate Symbol:</b> IwmRsnCodeWLMServBadSSN</p> <p><b>Meaning:</b> The subsystem name (SUBSYSNM=) specified is not the same as the one used by WLM to start the server.</p> <p><b>Action:</b> Verify that the start parameters for the application environment are coded correctly in the WLM ISPF application, and that those parameters are used by the started JCL procedure.</p>
8	xxxx084B	<p><b>Equate Symbol:</b> IwmRsnCodeWLMServBadSST</p> <p><b>Meaning:</b> The subsystem type (SUBSYS=) specified is not the same as the one used by WLM to start the server.</p> <p><b>Action:</b> Verify that the start parameters for the application environment are coded correctly in the WLM ISPF application, and that those parameters are used by the started JCL procedure.</p>
8	xxxx084D	<p><b>Equate Symbol:</b> IwmRsnCodeNotAuthConnect</p> <p><b>Meaning:</b> The caller must be supervisor state or have PSW key mask 0-7 authority to connect to the requested WLM services.</p> <p><b>Action:</b> Avoid requesting this function in this environment.</p>
8	xxxx084E	<p><b>Equate Symbol:</b> IwmRsnCodeWImServBadType</p> <p><b>Meaning:</b> For WLM started servers, the SERVER_TYPE= is not the one used to start the server.</p> <p><b>Action:</b> Specify the correct SERVER_TYPE.</p>

Table 94. Return and Reason Codes for the IWMCONN Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0853	<p><b>Equate Symbol:</b> IwmRsnCodeWlmQmBadType</p> <p><b>Meaning:</b> There is a queue manager or router environment of the specified subsystem name, but of a different type than that specified by the caller.</p> <p><b>Action:</b> Verify that the option for queue manager/router is specified correctly on IWMCONN. If the option is correct, then server address spaces for a different Server_Type exist and must terminate before the current space may connect as a queue manager or router.</p>
8	xxxx0855	<p><b>Equate Symbol:</b> IwmRsnCodeBadNumEUMax</p> <p><b>Meaning:</b> PARALLEL_EU variable is greater than the maximum of 1000.</p> <p><b>Action:</b> Specify a value between 1 and 1000.</p>
8	xxxx0856	<p><b>Equate Symbol:</b> IwmRsnCodeBadNumEUMin</p> <p><b>Meaning:</b> PARALLEL_EU variable is less than the minimum of 1.</p> <p><b>Action:</b> Specify a value between 1 and 1000.</p>
8	xxxx085C	<p><b>Equate Symbol:</b> IwmRsnCodeWrongNumEU</p> <p><b>Meaning:</b> Caller invoked service with a PARALLEL_EU value which is different from the PARALLEL_EU of existing servers in the application environment</p> <p><b>Action:</b> Ensure that all servers in the application environment specify the same PARALLEL_EU value.</p>
8	xxxx0873	<p><b>Equate Symbol:</b> IwmRsnCodeWrongSrvLmt</p> <p><b>Meaning:</b> Caller invoked service with a SERVER_LIMIT parameter setting which is different from the SERVER_LIMIT of existing servers in the application environment</p> <p><b>Action:</b> Ensure that all servers in the application environment specify the same SERVER_LIMIT value.</p>
8	xxxx0874	<p><b>Equate Symbol:</b> IwmRsnCodeWrongMngTsk</p> <p><b>Meaning:</b> Caller invoked service with a MANAGE_TASKS parameter setting which is different from the MANAGE_TASKS of existing servers in the application environment</p> <p><b>Action:</b> Ensure that all servers in the application environment specify the same MANAGE_TASKS value.</p>
8	xxxx0878	<p><b>Equate Symbol:</b> IwmRsnCodeBadNumLimitMax</p> <p><b>Meaning:</b> Caller invoked service with a SERVER_LIMIT parameter setting which exceeds the maximum number of tasks which can be started in a server address space.</p> <p><b>Action:</b> Correct number or do not specify SERVER_LIMIT parameter in order to use the default.</p>

Table 94. Return and Reason Codes for the IWMCONN Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0879	<p><b>Equate Symbol:</b> IwmRsnCodeBadNumLimitMin</p> <p><b>Meaning:</b> Caller invoked service with a SERVER_LIMIT parameter setting which is less than what has been defined on the PARALLEL_EU parameter.</p> <p><b>Action:</b> Ensure that SERVER_LIMIT is always greater or equal to PARALLEL_EU.</p>
8	xxxx087A	<p><b>Equate Symbol:</b> IwmRsnCodeNoQServer</p> <p><b>Meaning:</b> The MANAGE_TASKS parameter is not allowed when QUEUE_SERVER=YES has been specified.</p> <p><b>Action:</b> Ensure to use the parameters correctly.</p>
8	xxxx088E	<p><b>Equate Symbol:</b> IwmRsnCodeWlmServBadSSND</p> <p><b>Meaning:</b> For WLM started servers, the NODENM= is not the one used to start the server.</p> <p><b>Action:</b> Specify the correct NODENM.</p>
8	xxxx088F	<p><b>Equate Symbol:</b> IwmRsnCodeApplNotSSN</p> <p><b>Meaning:</b> The application environment name is defined for use by a different subsystem node.</p> <p><b>Action:</b> Check whether the correct application environment name is being used.</p>
8	xxxx089E	<p><b>Equate Symbol:</b> IwmRsnCodeServiceAModeMismatch</p> <p><b>Meaning:</b> The caller is in 64-bit address mode and tried to invoke a service macro that is only enabled for a 31-bit environment.</p> <p><b>Action:</b> Use the 64-bit enabled service macro (IWM4CON) or change the address mode of the caller to 31-bit.</p>
C	—	<p><b>Equate Symbol:</b> IwmRetCodeEnvError</p> <p><b>Meaning:</b> Environmental error.</p>
C	xxxx0C01	<p><b>Equate Symbol:</b> IwmRsnCodeNoStg</p> <p><b>Meaning:</b> Storage is not available for the request.</p> <p><b>Action:</b> There is a storage shortage. The function may work successfully at a later time.</p>
C	xxxx0C09	<p><b>Equate Symbol:</b> IwmRsnCodeNoResmgr</p> <p><b>Meaning:</b> Resource manager could not be established.</p> <p><b>Action:</b> No action required. This condition may be due to a storage shortage condition.</p>
C	xxxx0C14	<p><b>Equate Symbol:</b> IwmRsnCodeNoWorkShutDown</p> <p><b>Meaning:</b> No work selected. Caller is to shutdown.</p> <p><b>Action:</b> The server should shut down (terminate).</p>
C	xxxx0C19	<p><b>Equate Symbol:</b> IwmRsnCodeNotSecAuthConnect</p> <p><b>Meaning:</b> The caller is not authorized by SAF to connect to WLM with SERVER_MANAGER=YES.</p> <p><b>Action:</b> The security administrator must grant access to the appropriate resource.</p>



Table 94. Return and Reason Codes for the IWMCONN Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
C	xxxx0C1A	<p><b>Equate Symbol:</b> IwmRsnCodeApplNotDefined</p> <p><b>Meaning:</b> The application environment name is not defined in the active WLM policy.</p> <p><b>Action:</b> Check whether the correct application environment name is being used. If so, a service administrator must define the application environment in the WLM service definition.</p>
C	xxxx0C1B	<p><b>Equate Symbol:</b> IwmRsnCodeApplNotSST</p> <p><b>Meaning:</b> The application environment name is defined for use by a different subsystem type in the active WLM policy.</p> <p><b>Action:</b> Check whether the correct application environment name is being used. If so, a service administrator must change the application environment in the WLM service definition to specify the correct subsystem type.</p>
C	xxxx0C1F	<p><b>Equate Symbol:</b> IwmRsnCodeServerExists</p> <p><b>Meaning:</b> A server exists for the specified application environment which only allows 1 such server in the sysplex.</p> <p><b>Action:</b> Check whether the correct application environment name is being used. If so and the current server is shutting down, a retry may be successful after a delay.</p>
C	xxxx0C22	<p><b>Equate Symbol:</b> IwmRsnCodeApplEnvQuiesced</p> <p><b>Meaning:</b> The specified application environment has been quiesced, server cannot be started for the request.</p> <p><b>Action:</b> Restart the application environment and then retry the request.</p>
10	—	<p><b>Equate Symbol:</b> IwmRetCodeCompError</p> <p><b>Meaning:</b> Component error.</p> <p><b>Action:</b> No action required. The function may be successful if invoked again.</p>

## Examples

To connect to workload management specifying: a key value of 8, and a list of 7 address spaces involved in processing work, specify:

```
IWMCONN  SUBSYS=GENSUB,SUBSYSNM=SUBNAME,
         TOPOLOGY=LISTASCBS,NUMBERASCB=NUMSPACE
         CONNTKN=CTKN,CONNTKNKEYP=VALUE,CONNTKNKEY=KEY,
         RETCODE=RC,RSNCODE=RSN,
```

Where the following are declared:

GENSUB	DS	CL4	Generic subsystem type
SUBNAME	DS	CL8	Subsystem name
LISTASCBS	DS	CL28	List of 7 address spaces
NUMSPACE	DC	F'7'	Number of ASCBs
CTKN	DS	FL4	Connect token
KEY	DS	XL1	Key value

---

## IWMDISC – Disconnecting from Workload Management

IWMDISC allows the caller to disconnect from the workload management services. This means that the input connect token can no longer be passed to workload management macros such as IWMCLSFY and IWMRPT. When a program disconnects, any enclaves associated with the input connect token are deleted from the system. Any SRBs running in the enclave are run as preemptible SRBs at the priority of the home address space. Any enclave TCBs are converted to ordinary TCBs.

You should issue this macro once during shutdown of the connecting address space.

### Note

It is recommended to use the new equivalent service IWM4DIS introduced with z/OS V1R6. IWM4DIS also supports 64-bit address space. For further information refer to Chapter 66, “IWM4DIS – Disconnecting from Workload Management,” on page 553.

## Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	When the corresponding Connect (IWMCONN) invocation specified WORK_MANAGER=YES or QUEUE_MANAGER=YES, ROUTER=YES, or EXPTIMPT=YES, supervisor state or program key mask (PKM) allowing keys 0-7.
	When the corresponding Connect (IWMCONN) invocation specified WORK_MANAGER=NO, QUEUE_MANAGER=NO, ROUTER=NO, EXPTIMPT=NO, and SERVER_MANAGER=YES, problem state with any PSW key.
<b>Dispatchable unit mode:</b>	Task or SRB
	When the corresponding Connect (IWMCONN) invocation specified SERVER_MANAGER=YES, task mode.
<b>Cross memory mode:</b>	The current Home address space must be the same as Home when the corresponding Connect was invoked. Any PASN, any SASN.
	When the corresponding Connect (IWMCONN) invocation specified SERVER_MANAGER=YES, non-cross memory mode, P=S=H.
<b>AMODE:</b>	31-bit
<b>ASC mode:</b>	Primary
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks may be held.
	When the corresponding Connect (IWMCONN) invocation specified SERVER_MANAGER=YES, SERVER_TYPE=ROUTING, NO FRRs may be set.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

## Programming Requirements

1. The macro CVT must be included to use this macro.
2. The macro IWMYCON must be included to use this macro.
3. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
4. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, IWMSRNCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.

## Restrictions

1. No FRRs may be set when calling to disconnect a space which is connected as a server manager with server type = routing.
2. If the key specified on IWMCONN was a user key (8-F), then the following must ALL be true:
  - caller must be in non-cross-memory mode (P=S=H). This implies that the current primary must match the primary at the time that IWMCONN was invoked. Running in a subspace is not supported.
  - must be in TCB mode (not SRB)
  - current TCB must match the TCB at the time that IWMCONN was invoked.
3. This service should not be invoked while in a RTM termination routine (resource manager) for the TCB owning the connect token since MVS will have its own resource cleanup routine and unpredictable results would occur. It is legitimate to use this service while in a recovery routine, however, or in mainline processing.

## Input Register Information

Before issuing the IWMDISC macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

## Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Reason code if GR15 return code is non-zero
1	Used as work registers by the system
2-13	Unchanged
14	Used as work registers by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

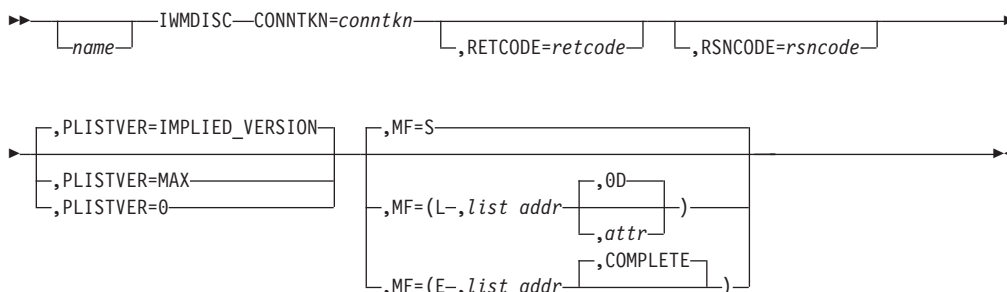
Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

## Performance Implications

None.

## Syntax

### main diagram



## Parameters

The parameters are explained as follows:

### *name*

An optional symbol, starting in column 1, that is the name on the IWMDISC macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

### **CONNTKN=conntkn**

A required input parameter, which contains the connect token for the environment to be disconnected.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32 bit field.

### **,MF=S**

**,MF=(L,list addr)**

**,MF=(L,list addr,attr)**

**,MF=(L,list addr,OD)**

**,MF=(E,list addr)**

**,MF=(E,list addr,COMPLETE)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

**,list addr**

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

**,attr**

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

**,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

**,PLISTVER=IMPLIED\_VERSION****,PLISTVER=MAX****,PLISTVER=0**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form, when both are assembled with the same level of the system. In this way, MAX ensures that the parameter list does not overwrite nearby storage.

- **0**, if you use the currently available parameters.

**To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX
- A decimal value of 0

**,RETCODE=retcode**

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,RSNCODE=rsncode**

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

## ABEND Codes

None.

## Return and Reason Codes

When the IWMDISC macro returns control to your program:

- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, if you coded RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 95. Return and Reason Codes for the IWMDISC Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> <code>IwmRetCodeOk</code> <b>Meaning:</b> Successful completion. <b>Action:</b> None required.
4	—	<b>Equate Symbol:</b> <code>IwmRetCodeWarning</code> <b>Meaning:</b> Successful completion, unusual conditions noted.
4	xxxx0401	<b>Equate Symbol:</b> <code>IwmRsnCodeNoWLM</code> <b>Meaning:</b> System does not support work manager services. This return code is only set when the MVS release is prior to MVS/ESA SP5.1.0. <b>Action:</b> None required.
4	xxxx0409	<b>Equate Symbol:</b> <code>IwmRsnCodeNoConn</code> <b>Meaning:</b> Input connection token does not reflect an active connection to WLM. <b>Action:</b> None required.
8	—	<b>Equate Symbol:</b> <code>IwmRetCodeInvocError</code> <b>Meaning:</b> Invalid invocation environment or parameters.
8	xxxx0801	<b>Equate Symbol:</b> <code>IwmRsnCodeSrbMode</code> <b>Meaning:</b> Caller is in SRB mode. <b>Action:</b> Avoid requesting this function while in SRB mode.
8	xxxx0802	<b>Equate Symbol:</b> <code>IwmRsnCodeXmemUserKeyTkn</code> <b>Meaning:</b> Caller is in cross-memory mode while the token was obtained in a user key. <b>Action:</b> Avoid requesting this function while in cross-memory mode.
8	xxxx0803	<b>Equate Symbol:</b> <code>IwmRsnCodeDisabled</code> <b>Meaning:</b> Caller is disabled. <b>Action:</b> Avoid requesting this function while disabled.
8	xxxx0804	<b>Equate Symbol:</b> <code>IwmRsnCodeLocked</code> <b>Meaning:</b> Caller is locked. <b>Action:</b> Avoid requesting this function while locked.

Table 95. Return and Reason Codes for the IWMDISC Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0809	<p><b>Equate Symbol:</b> <code>IwmRsnCodeSrbUserKeyTkn</code></p> <p><b>Meaning:</b> Caller is in SRB mode, while the token was obtained in a user key (8-F).</p> <p><b>Action:</b> Avoid requesting this function in SRB mode for tokens associated with user key.</p>
8	xxxx080A	<p><b>Equate Symbol:</b> <code>IwmRsnCodeTcbNotOwnerUserKeyTkn</code></p> <p><b>Meaning:</b> Current TCB is not the owner, while the token was obtained in a user key (8-F).</p> <p><b>Action:</b> Avoid requesting this function under a TCB other than the owner for a token associated with user key.</p>
8	xxxx080B	<p><b>Equate Symbol:</b> <code>IwmRsnCodeBadPI</code></p> <p><b>Meaning:</b> Error accessing parameter list.</p> <p><b>Action:</b> Check for possible storage overlay. Also check if you call this macro in 64-bit address mode. Refer to the description of <code>xxxx089E</code> for further information.</p>
8	xxxx0810	<p><b>Equate Symbol:</b> <code>IwmRsnCodeEutFrr</code></p> <p><b>Meaning:</b> Caller has EUT FRR established.</p> <p><b>Action:</b> Avoid requesting this function with an EUT FRR set.</p>
8	xxxx0823	<p><b>Equate Symbol:</b> <code>IwmRsnCodeDatoff</code></p> <p><b>Meaning:</b> Caller invoked service while DATOFF</p> <p><b>Action:</b> Avoid requesting this function in this environment.</p>
8	xxxx0824	<p><b>Equate Symbol:</b> <code>IwmRsnCodeAmode24</code></p> <p><b>Meaning:</b> Caller invoked service but was in 24 bit addressing mode.</p> <p><b>Action:</b> Request this function only when you are in 31 bit addressing mode.</p>
8	xxxx0825	<p><b>Equate Symbol:</b> <code>IwmRsnCodeAscModeNotPrimary</code></p> <p><b>Meaning:</b> Caller invoked service but was not DAT on Primary ASC mode.</p> <p><b>Action:</b> Avoid requesting this function in this environment.</p>
8	xxxx0826	<p><b>Equate Symbol:</b> <code>IwmRsnCodeTaskTerm</code></p> <p><b>Meaning:</b> Caller invoked service while task termination is in progress for the TCB associated with the owner.</p> <p><b>Action:</b> Avoid requesting this function in this environment.</p>
8	xxxx0827	<p><b>Equate Symbol:</b> <code>IwmRsnCodeRsvdNot0</code></p> <p><b>Meaning:</b> Reserved field in parameter list was non-zero.</p> <p><b>Action:</b> Check for possible storage overlay of the parameter list.</p>
8	xxxx0828	<p><b>Equate Symbol:</b> <code>IwmRsnCodeBadVersion</code></p> <p><b>Meaning:</b> Version number in parameter list is not valid.</p> <p><b>Action:</b> Check for possible storage overlay of the parameter list.</p>

Table 95. Return and Reason Codes for the IWMDISC Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx082F	<p><b>Equate Symbol:</b> IwmRsnCodeWrongHome</p> <p><b>Meaning:</b> Caller invoked the service from the wrong home address space.</p> <p><b>Action:</b> Invoke the function with the correct home address space.</p>
8	xxxx0841	<p><b>Equate Symbol:</b> IwmRsnCodeXMemMode</p> <p><b>Meaning:</b> Caller is in cross memory mode.</p> <p><b>Action:</b> Invoke the function in non-cross memory mode.</p>
8	xxxx084D	<p><b>Equate Symbol:</b> IwmRsnCodeNotAuthConnect</p> <p><b>Meaning:</b> The caller must be supervisor state or have PSW key mask 0-7 authority to disconnect from the requested WLM services.</p> <p><b>Action:</b> Avoid requesting this function in this environment.</p>
8	xxxx089E	<p><b>Equate Symbol:</b> IwmRsnCodeServiceAModeMismatch</p> <p><b>Meaning:</b> The caller is in 64-bit address mode and tried to invoke a service macro that is only enabled for a 31-bit environment.</p> <p><b>Action:</b> Use the 64-bit enabled service macro (IWM4DIS) or change the address mode of the caller to 31-bit.</p>
10	—	<p><b>Equate Symbol:</b> IwmRetCodeCompError</p> <p><b>Meaning:</b> Component error.</p> <p><b>Action:</b> No action required. The function may be successful if invoked again.</p>

## IWMCREA – Create an Enclave

The purpose of this service is to create an Enclave where possibly multiple SRBs and/or TCBs may be simultaneously executing or scheduled. For the duration of each Enclave, all SRBs and TCBs associated with the Enclave are treated as part of a single work request. All SRBs and/or TCBs associated with the Enclave accumulate service as a single entity and are managed as a single entity. The address spaces where Enclave SRBs are dispatched, as defined by the ENV= parameter of IEAMSCHD, should be non-swappable.

For more information about managing address spaces with enclaves, see “Performance Management of Address Spaces with Enclaves” on page 45.

**Note:** An address space must be non-swappable if it has enclave SRBs dispatched and SYSEVENT ENCASSOC has not been issued.

For TYPE=INDEPENDENT Enclaves, a new work business unit of work is created and classified according to the input Connect token’s subsystem type and subsystem name, along with whatever other attributes are passed via the Classify parameter list. The current home address space is considered the owner.

For TYPE=DEPENDENT enclaves, SRM considers the enclave to be part of the current home address space’s transaction, which then becomes the owning space. This space need not be connected to WLM via IWMCONN.



For TYPE=MONENV enclaves, SRM considers the enclave to be part of the address space's transaction which is delayed according to the input monitoring environment, as set when IWMMINIT or IWMMRELA was used. This space becomes the owning space. This space need not be connected to WLM via IWMCONN.

For both TYPE=MONENV and TYPE=DEPENDENT enclaves, SRM will change the enclave to TYPE=INDEPENDENT if the owning address space's transaction ends.

For both TYPE=MONENV and TYPE=DEPENDENT enclaves, SRM will check the enclave for period switch when the owning address space is swapped in. If the owning address space is swapped out SRM will continue to accumulate service for any enclaves owned by the space, but will not check the address space and any owned enclave for period switch until the address space is swapped in again. The presence of enclaves does not make the address space appear to be ready from an SRM point of view.

Enclaves are deleted if the owning address space terminates. TYPE=INDEPENDENT enclaves are deleted if the owning address space disconnects or the TCB which connected terminates.

Enclaves should only be created when this environment is ready for execution, and should not be used when prolonged queueing effects are possible prior to the scheduling of the first SRB (IEAMSCHD) or the first task join (IWMEJOIN). "Prolonged" would certainly include times measured in seconds. The service allows the caller to pass the queueing time prior to creation of the Enclave so that this may be separately reported.

#### Note

It is recommended to use the new equivalent service IWM4ECRE introduced with z/OS V1R6. IWM4ECRE also supports 64-bit address space. For further information refer to Chapter 67, "IWM4ECRE – Create an Enclave," on page 561.

## Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Supervisor state or program key mask (PKM) allowing keys 0-7.
<b>Dispatchable unit mode:</b>	Task or SRB
<b>Cross memory mode:</b>	Any PASN, any HASN, any SASN
<b>AMODE:</b>	31-bit
<b>ASC mode:</b>	Primary or access register (AR)
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks may be held.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

## Programming Requirements

1. The macro IWMYCON must be included to use this macro.
2. The macro CVT must be included to use this macro.

## IWMECREA Macro

3. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
4. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, IWMSRNCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.
5. All character data, unless otherwise specified, is assumed to be left justified and padded with blanks on the right, as needed, to occupy the specified number of bytes.

## Restrictions

1. The Connect token from the input classify parameter list must be owned by the current home address space and must be associated with a system key (0-7), as specified on IWMCONN. The Classify parameter list and hence the Connect token is only relevant for TYPE=INDEPENDENT Enclaves.
2. Since this service may only be used by system-like code, some validity checking on the parameter list is not performed. These checks would only be needed if the macro were not used to invoke the service routine.
3. The variable length fields associated with the classify parameter list (the classify parameter list is only relevant for certain options) given by the CLSFY keyword have the following limitations in addition to those documented in IWMCLSFY:
  - SUBSYSPM is limited to 255 bytes
  - COLLECTION is limited to 18 bytes
  - CORRELATION is limited to 12 bytes
4. When TYPE(MONENV) is specified the following apply:
  - If the key specified on IWMMCREA was a user key (8-F), then primary or home addressability must exist to the performance block IWMMCREA obtained. This condition is satisfied by ensuring that the current primary or home address space matches primary (=home) at the time that IWMMCREA was invoked.
  - The caller must serialize to prevent any delay monitoring services from being invoked concurrently for the environment represented by the monitoring token.
  - Only limited checking is done against the input monitoring token.
  - TYPE=MONENV enclaves cannot be created for report-only monitoring environments.
5. This macro may only be used on z/OS R2 or higher levels for EXSTARTDEFER keyword.

## Input Register Information

Before issuing the IWMECREA macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

## Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Reason code if GR15 return code is non-zero
1	Used as work registers by the system
2-13	Unchanged
14	Used as work registers by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

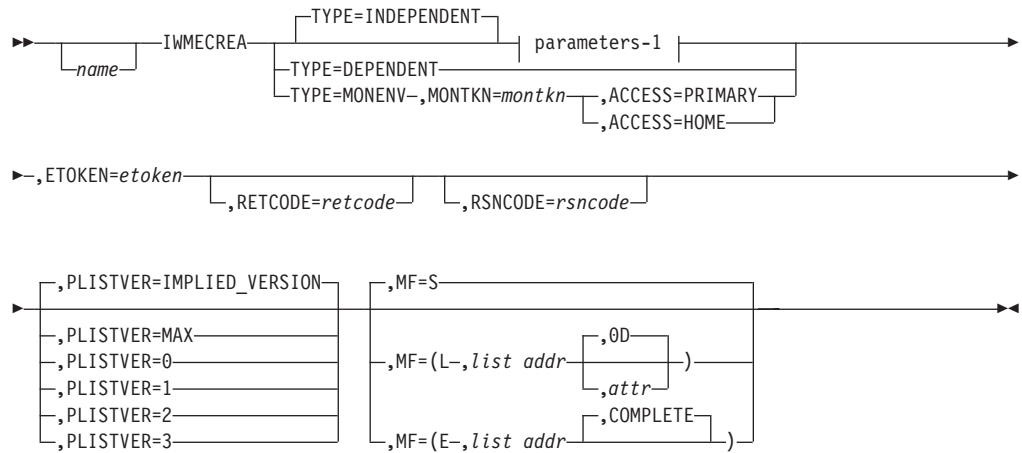
Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

## Performance Implications

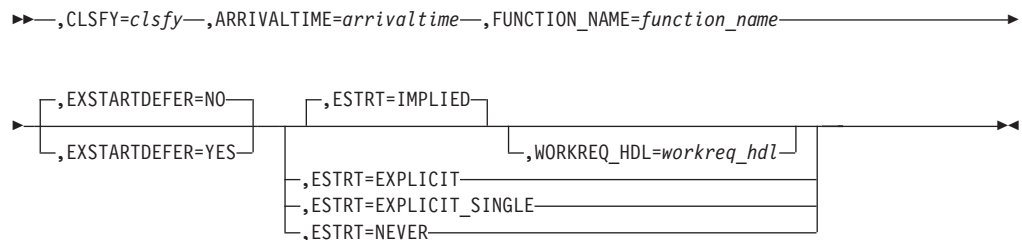
None.

## Syntax

### main diagram



### parameters-1



## Parameters

The parameters are explained as follows:

### name

An optional symbol, starting in column 1, that is the name on the IWMECREA macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

### ,ACCESS=PRIMARY

### **,ACCESS=HOME**

When TYPE=MONENV is specified, a required parameter, which describes how to access the monitoring environment.

### **,ACCESS=PRIMARY**

indicates that the monitoring environment can be accessed in the caller's primary address space. This would be appropriate if the monitoring environment was established (by IWMMCREA) to be used by routines in a specific system key or if it was established to be used in a specific user key in the current primary.

### **,ACCESS=HOME**

indicates that the monitoring environment must be accessed in the home address space, which is not the caller's primary address space. This would be appropriate if the monitoring environment was established (by IWMMCREA) for use by a specific user key.

### **,ARRIVALTIME=*arrivaltime***

When TYPE=INDEPENDENT is specified, a required input parameter, which contains the work arrival time in STCK format. This is the time at which the business work request is considered to have arrived and from which point the system evaluates elapsed time for the work request.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 64 bit field.

### **,CLSFY=*clsfy***

When TYPE=INDEPENDENT is specified, a required input parameter, which contains the classification information in the format of the parameter list for IWMCLSFY. NOTE that this name is the data area name, not its pointer. IWMCLSFY MF(M) should be used to initialize the area prior to invocation of IWMECREA. The Classify parameter list must be from MVS/ESA SP5.2.0 or later.

Note that the variable length fields associated with the classify parameter list given by the CLSFY keyword have the following limitations in addition to those documented in IWMCLSFY:

- SUBSYSPM is limited to 255 bytes
- COLLECTION is limited to 18 bytes
- CORRELATION is limited to 12 bytes

**To code:** Specify the RS-type address, or address in register (2)-(12), of a character field.

### **,ESTRT=IMPLIED**

### **,ESTRT=EXPLICIT**

### **,ESTRT=EXPLICIT\_SINGLE**

### **,ESTRT=NEVER**

When TYPE=INDEPENDENT is specified, an optional parameter, which denotes how the work manager indicates the start and end point of an EWLM work request when participating in cross-platform enterprise workload management (EWLM). The default is ESTRT=IMPLIED.

### **,ESTRT=IMPLIED**

If the work manager previously connected to WLM with IWMMCONN EWLM=YES, a work request is started implicitly when the enclave is created. If IWMESTOP was not invoked before, the work request will be stopped implicitly when the enclave is deleted.

**,ESTRT=EXPLICIT**

The work manager indicates the start and end point of an EWLM work request by invoking the services IWMESTRT and IWMESTOP. Note that this option is only meaningful, if the work manager previously connected to WLM with IWMCONN EWLM=YES.

**,ESTRT=EXPLICIT\_SINGLE**

Indicates the same as option ESTRT=EXPLICIT and, in addition, the application ensures that only one work request is active. No nested calls to IWMESTRT are allowed. If this option is specified the CPU consumption on all EWLM enclave services (IWMEGCOR, IWMESTRT, IWMESTOP, IWMEBLK, IWMEUBLK) will be reduced. If ESTRT=EXPLICIT\_SINGLE is specified on IWMECREA, the application must also add the EWLMODE=EXPLICIT\_SINGLE parameter on all calls to IWMEGCOR, IWMESTRT, IWMESTOP, IWMEBLK and IWMEUBLK. If this parameter is used, the application has some restrictions on all calls to IWMEGCOR, IWMESTRT, IWMESTOP, IWMEBLK and IWMEUBLK. Refer to the corresponding macro descriptions for details.

**,ESTRT=NEVER**

Indicates that this enclave will never use any EWLM-related enclave services (IWMEGCOR, IWMESTRT, IWMESTOP, IWMEBLK, and IWMEUBLK) after the enclave has been created, even if the work manager has registered (IWM4CON or IWMCONN) with EWMLM=YES. Moreover IWMECREA will not start an EWLM work request on the enclave and will not do any EWLM-related processing.

**,ETOKEN=*etoken***

A required output parameter, which will receive the enclave token.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

**,EXSTARTDEFER=NO****,EXSTARTDEFER= $\bar{Y}$ ES**

When TYPE=INDEPENDENT is specified, an optional parameter, which indicates whether the Enclave execution start time should begin when the first IWMSTBGN or IWMEJOIN is executed. The time between enclave create and the first IWMSTBGN or IWMEJOIN is assumed to be the queue time. The default is EXSTARTDEFER=NO

**,EXSTARTDEFER=NO**

indicates that the Enclave execution start time should not begin when the first IWMSTBGN or IWMEJOIN is executed.

**,EXSTARTDEFER=YES**

indicates that the Enclave execution start time should begin when the first IWMSTBGN or IWMEJOIN is executed.

**,FUNCTION\_NAME=*function\_name***

When TYPE=INDEPENDENT is specified, a required input parameter, which contains the descriptive name for the function for which the Enclave was created.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

**,MF=S**

**,MF=(L,*list addr*)**

**,MF=(L,*list addr,attr*)**

**,MF=(L,*list addr,OD*)**

**,MF=(E,*list addr*)**

**,MF=(E,*list addr*,**COMPLETE**)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

**,*list addr***

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

**,*attr***

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

**,**COMPLETE****

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

**,MONTKN=*montkn***

When TYPE=MONENV is specified, a required input parameter, which contains the delay monitoring token which describes the current business unit of work. If the monitoring environment is related to an address space, then it must be the current home address space.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32 bit field.

**,PLISTVER=IMPLIED\_VERSION**

**,PLISTVER=MAX**

**,PLISTVER=0**

**,PLISTVER=1**

**,PLISTVER=2**

**,PLISTVER=3**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify `PLISTVER=MAX` on the list form of the macro. Specifying `MAX` ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form, when both are assembled with the same level of the system. In this way, `MAX` ensures that the parameter list does not overwrite nearby storage.

- **0**, if you use the currently available parameters.
- **1**, which supports both the following parameters and those from version 0 :

`ACCESS`                      `MONTKN`                      `TYPE`

- **2**, which supports both the following parameters and those from version 0 and 1:

`EXSTARTDEFER`

- **3**, which supports both the following parameters and those from version 0, 1, and 2:

`ESTRT`                                      `WORKREQ_HDL`

**To code:** Specify one of the following:

- `IMPLIED_VERSION`
- `MAX`
- A decimal value of 0, 1, 2, or 3

**,RETCODE=retcode**

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,RSNCODE=rsncode**

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**TYPE=INDEPENDENT**

**TYPE=DEPENDENT**

**TYPE=MONENV**

An optional parameter, which indicates the type of Enclave being created. The default is `TYPE=INDEPENDENT`.

**TYPE=INDEPENDENT**

indicates that the Enclave represents a new business unit of work with its own business objectives.

**TYPE=DEPENDENT**

indicates that the Enclave represents a continuation of the business unit of work represented by the current home address space.

**TYPE=MONENV**

indicates that the Enclave represents a continuation of the business unit of work represented by the input monitoring environment.

**,WORKREQ\_HDL=workreq\_hdl**

When `ESTRT=IMPLIED` and `TYPE=INDEPENDENT` are specified, an optional output parameter that will receive the handle which represents the work

## IWMECREA Macro

request. The application must pass this handle to the other work request services IWMESTOP, IWMEBLK, IWMEUBLK, and IWMEGCOR.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

## ABEND Codes

None.

## Return and Reason Codes

When the IWMECREA macro returns control to your program:

- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, if you coded RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 96. Return and Reason Codes for the IWMECREA Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> <code>IwmRetCodeOk</code> <b>Meaning:</b> Successful completion. <b>Action:</b> None required.
8	—	<b>Equate Symbol:</b> <code>IwmRetCodeInvocError</code> <b>Meaning:</b> Invalid invocation environment or parameters.
8	xxxx0803	<b>Equate Symbol:</b> <code>IwmRsnCodeDisabled</code> <b>Meaning:</b> Caller is disabled. <b>Action:</b> Avoid requesting this function while disabled.
8	xxxx0804	<b>Equate Symbol:</b> <code>IwmRsnCodeLocked</code> <b>Meaning:</b> Caller is locked. <b>Action:</b> Avoid requesting this function while locked.
8	xxxx080B	<b>Equate Symbol:</b> <code>IwmRsnCodeBadPI</code> <b>Meaning:</b> Error accessing parameter list. <b>Action:</b> Check for possible storage overlay. Also check if you call this macro in 64-bit address mode. Refer to the description of xxxx089E for further information.
8	xxxx0820	<b>Equate Symbol:</b> <code>IwmRsnCodeBadMonEnv</code> <b>Meaning:</b> Monitoring environment does not pass short form verification. <b>Action:</b> Check for possible storage overlay.
8	xxxx0821	<b>Equate Symbol:</b> <code>IwmRsnCodeBadConn</code> <b>Meaning:</b> Connect token from the input classify parameter list does not pass validity checking. <b>Action:</b> Check for possible storage overlay.



Table 96. Return and Reason Codes for the IWMECREA Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0824	<p><b>Equate Symbol:</b> lwmRsnCodeAmode24</p> <p><b>Meaning:</b> Caller invoked service but was in 24 bit addressing mode.</p> <p><b>Action:</b> Request this function only when you are in 31 bit addressing mode.</p>
8	xxxx0828	<p><b>Equate Symbol:</b> lwmRsnCodeBadVersion</p> <p><b>Meaning:</b> Version number in parameter list is not valid.</p> <p><b>Action:</b> Check for possible storage overlay of the parameter list.</p>
8	xxxx0836	<p><b>Equate Symbol:</b> lwmRsnCodeMaxEnclave</p> <p><b>Meaning:</b> Enclave could not be created because the Enclave limit has been reached.</p> <p><b>Action:</b> Check for possible problems wherein Enclaves are not being deleted as expected or excessive numbers of Enclaves are being created in a loop.</p>
8	xxxx0837	<p><b>Equate Symbol:</b> lwmRsnCodeUserKeyConntkn</p> <p><b>Meaning:</b> Connect token from the input classify parameter list is associated with a user key.</p> <p><b>Action:</b> Invoke the function with a token associated with a system key.</p>
8	xxxx0838	<p><b>Equate Symbol:</b> lwmRsnCodeClisfyAreaTooBig</p> <p><b>Meaning:</b> Input area associated with classification information is larger than supported.</p> <p><b>Action:</b> Invoke the function with an area of the proper size. Check for possible storage overlay.</p>
8	xxxx0839	<p><b>Equate Symbol:</b> lwmRsnCodeClisfyPITooSmall</p> <p><b>Meaning:</b> Input Classify parameter list is too small.</p> <p><b>Action:</b> Invoke the function with an area of the proper size. Check for possible storage overlay.</p>
8	xxxx083B	<p><b>Equate Symbol:</b> lwmRsnCodeHomeNotOwnConn</p> <p><b>Meaning:</b> Home address space does not own the connect token from the input classify parameter list.</p> <p><b>Action:</b> Invoke the function with the correct home address space.</p>
8	xxxx0840	<p><b>Equate Symbol:</b> lwmRsnCodeServiceNotEnabled</p> <p><b>Meaning:</b> Caller's space connection is not enabled for this service.</p> <p><b>Action:</b> Avoid requesting this function under the input connection. IWMCNN options must be specified previously to enable this service.</p>
8	xxxx085D	<p><b>Equate Symbol:</b> lwmRsnCodeMonenvNotHome</p> <p><b>Meaning:</b> The input monitoring environment is related to an address space other than home.</p> <p><b>Action:</b> None required.</p>

Table 96. Return and Reason Codes for the IWMECREA Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0892	<p><b>Equate Symbol:</b> IwmRsnCodeEWLMCorrNotAllowed</p> <p><b>Meaning:</b> Passed classification information must not contain an EWLM correlator (EWLM_CORR).</p> <p><b>Action:</b> Do not pass an EWLM correlator within the classification information when invoking the service with ESTRT=EXPLICIT.</p>
8	xxxx0894	<p><b>Equate Symbol:</b> IwmRsnCodeInvalidEWLMCorr</p> <p><b>Meaning:</b> Passed classification information contains an ARM correlator (EWLM_CORR) that does not pass validity checking. The architected ARM correlator length field in the first two bytes of the EWLM_CORR is either less than 4 (X'0004') or greater than 512 (X'0200').</p> <p><b>Action:</b> Check the specification of the EWLM correlator in the classification information.</p>
8	xxxx0895	<p><b>Equate Symbol:</b> IwmRsnCodeEWLMServNotEnabled</p> <p><b>Meaning:</b> Service is not enabled because caller invoked the IWMCONN service with EWLM=NO.</p> <p><b>Action:</b> Specify the parameter WORKREQ_HDL only when connected with EWLM=YES.</p>
8	xxxx089E	<p><b>Equate Symbol:</b> IwmRsnCodeServiceAModeMismatch</p> <p><b>Meaning:</b> The caller is in 64-bit address mode and tried to invoke a service macro that is only enabled for a 31-bit environment.</p> <p><b>Action:</b> Use the 64-bit enabled service macro (IWM4ECRE) or change the address mode of the caller to 31-bit.</p>
C	—	<p><b>Equate Symbol:</b> IwmRetCodeEnvError</p> <p><b>Meaning:</b> Environmental error.</p>
C	xxxx0C01	<p><b>Equate Symbol:</b> IwmRsnCodeNoStg</p> <p><b>Meaning:</b> Storage is not available for the request.</p> <p><b>Action:</b> There is a storage shortage. The function may work successfully at a later time.</p>
C	xxxx0C0C	<p><b>Equate Symbol:</b> IwmRsnCodeClassifyFail</p> <p><b>Meaning:</b> Received a non-zero return code from the classification service, IWMCLSFY.</p> <p><b>Action:</b> No action required. Reinvoking the function later may succeed.</p>
C	xxxx0C0D	<p><b>Equate Symbol:</b> IwmRsnCodeBadClsfy</p> <p><b>Meaning:</b> Classification apparently can not access the current policy, possibly due to a policy switch in progress.</p> <p><b>Action:</b> Invoke the function when the conditions are alleviated.</p>
C	xxxx0C20	<p><b>Equate Symbol:</b> IwmRsnCodeDepClassifyFail</p> <p><b>Meaning:</b> Unable to obtain classification attributes for a dependent enclave.</p> <p><b>Action:</b> None required.</p>

Table 96. Return and Reason Codes for the IWMECREA Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
C	xxxx0C21	<p><b>Equate Symbol:</b> IwmRsnCodeNoMonEnvErr</p> <p><b>Meaning:</b> Input monitoring token indicates no monitoring environment was established.</p> <p><b>Action:</b> None required.</p>
10	—	<p><b>Equate Symbol:</b> IwmRetCodeCompError</p> <p><b>Meaning:</b> Component error.</p> <p><b>Action:</b> No action required. The function may be successful if invoked again.</p>

## IWMEDELE – Delete an Enclave

The purpose of this service is to delete an enclave, so that no SRBs or TCBs exist within the enclave and no new SRBs may be scheduled into the enclave, nor may any TCBs join the enclave. Some residual enclave related CPU time will not be accounted back to the work request whenever active enclave SRBs/TCBs were present at the time IWMEDELE is invoked. SRBs scheduled to the enclave which have not completed will be converted to ordinary preemptable SRBs. TCBs joined to the enclave which have not completed will be converted to ordinary TCBs.

If IWMEDELE is invoked for an enclave which is registered, the enclave is considered only logically deleted while all its functionality stays in place. Physical deletion is deferred until all interested parties have deregistered the enclave. The caller does not receive any notice when the physical deletion of the enclave is done.

When an enclave is deleted, the work request is considered to have finished and all related resource accounting will be finalized.

IWMEDELE cannot be used to delete a foreign enclave. The IWMUIMPT macro must be used instead.

### Note

It is recommended to use the new equivalent service IWM4EDEL introduced with z/OS V1R6. IWM4EDEL also supports 64-bit address space. For further information refer to Chapter 68, “IWM4EDEL – Delete an Enclave,” on page 573.

## Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Supervisor state or program key mask (PKM) allowing keys 0-7.
<b>Dispatchable unit mode:</b>	Task or SRB
<b>Cross memory mode:</b>	Any PASN, any HASN, any SASN
<b>AMODE:</b>	31-bit
<b>ASC mode:</b>	Primary or access register (AR)

## IWMEDELE Macro

<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks may be held. FRR environments may be established.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

## Programming Requirements

1. The macro CVT must be included to use this macro.
2. The macro IWMYCON must be included to use this macro.
3. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
4. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, IWMSRNCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.
5. All character data, unless otherwise specified, is assumed to be left justified and padded with blanks on the right, as needed, to occupy the specified number of bytes.
6. Since this service may only be used by system-like code, some validity checking on the parameter list is not performed. These checks would only be needed if the macro were not used to invoke the service routine.

## Restrictions

This macro supports multiple versions. Some keywords are only supported by certain versions. Refer to the PLISTVER parameter description for further information.

## Input Register Information

Before issuing the IWMEDELE macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

## Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Reason code if GR15 return code is non-zero
1	Used as work register by the system
2-13	Unchanged
14	Used as work registers by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

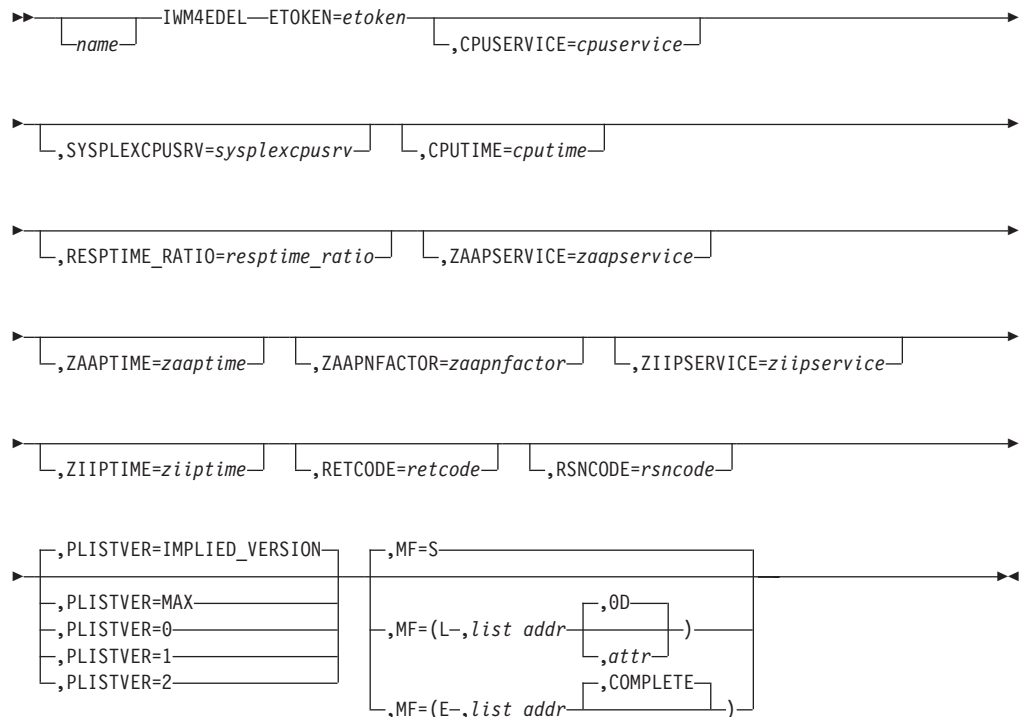
Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

## Performance Implications

None.

## Syntax

### main diagram



## Parameters

The parameters are explained as follows:

### *name*

An optional symbol, starting in column 1, that is the name on the IWMEDELE macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

### **,CPUSERVICE=cpuservice**

An optional output parameter, which will contain the CPU service accumulated by the enclave on the local system.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 64-bit field.

### **,CPUTIME=cputime**

An optional output parameter, which will contain the total CPU time accumulated by the enclave on the local system.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 64-bit field.

## IWMEDELE Macro

### **ETOKEN=etoken**

A required input parameter, which contains the enclave token to be returned.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

### **,MF=S**

**,MF=(L,list addr)**

**,MF=(L,list addr,attr)**

**,MF=(L,list addr,0D)**

**,MF=(E,list addr)**

**,MF=(E,list addr,COMPLETE)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

**,list addr**

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

**,attr**

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

**,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

**,PLISTVER=IMPLIED\_VERSION**

**,PLISTVER=MAX**

**,PLISTVER=0**

**,PLISTVER=1**

**,PLISTVER=2**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form, when both are assembled with the same level of the system. In this way, MAX ensures that the parameter list does not overwrite nearby storage.

- **0**, which supports all parameters except those specifically referenced in higher versions.
- **1**, which supports both the following parameters and those from version 0:

RESPTIME\_RATIO

- **2**, which supports both the following parameters and those from version 0 and 1:

ZAAPNFACTOR	ZAAPTIME	ZIIPTIME
ZAAPSERVICE	ZIIPSERVICE	

**To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX
- A decimal value of 0, 1, or 2

**,RESPTIME\_RATIO=***resptime\_ratio*

An optional output parameter, which contains the response time ratio times 100:  $\text{act.resp.time} / \text{goal} * 100$  if the enclave has a response time goal (limited to:  $1 \leq \text{RESPTIME\_RATIO} \leq 1000$ ) 0 otherwise

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32-bit field.

**,RETCODE=***retcode*

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,RSNCODE=***rsncode*

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,SYSPLEXCPUSRV=***sysplexcpusrv*

An optional output parameter, which will contain the CPU service accumulated by the enclave on the local system and on other systems through the use of the IWMEPT and IWMIMPT services. If the IWMEPT and IWMIMPT services were not used, SYSPLEXCPUSRV returns the same value as CPUSERVICE.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 64-bit field.

**,ZAAPNFACTOR=***zaapnfactor*

An optional output parameter, which contains the normalization factor for application assist processors (zAAPs). If zAAPs are running at a different speed, multiply zAAP service and times with this factor and divide the result by 256 to normalize the values to the speed of regular CPs. Note however, that if there has been a speed change of zAAP processors during the life time of the enclave, this calculation will return incorrect data.

## IWMEDELE Macro

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

### **,ZAAPSERVICE=***zaapservice*

An optional output parameter, which contains the application assist processor (zAAP) service accumulated by the enclave on the local system. The value is not normalized to the speed of regular CPs, but is expressed in zAAP speed which might be different. You may use ZAAPNFACTOR to normalize the value to the speed of regular CPs. Note however, that if the zAAP speed changed during the life time of the enclave, this value cannot be normalized correctly.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 64-bit field.

### **,ZAAPTIME=***zaaptime*

An optional output parameter, which contains the total application assist processor (zAAP) time accumulated by the enclave on the local system. The value is not normalized to the speed of regular CPs, but is expressed in zAAP speed which might be different. You may use ZAAPNFACTOR to normalize the value to the speed of regular CPs. Note however, that if the zAAP speed changed during the life time of the enclave, this value cannot be normalized correctly.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 64-bit field.

### **,ZIIPSERVICE=***ziipservice*

An optional output parameter, which contains the integrated information processor (zIIP) service accumulated by the enclave on the local system. The service is normalized to standard processor speed.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 64-bit field.

### **,ZIIPTIME=***ziiptime*

An optional output parameter, which contains the total integrated information processor (zIIP) time accumulated by the enclave on the local system. The time is normalized to standard processor speed.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 64-bit field.

## ABEND Codes

None.

## Return and Reason Codes

When the IWMEDELE macro returns control to your program:

- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, if you coded RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.



Table 97. Return and Reason Codes for the IWM4EDEL Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> IwmRetCodeOk <b>Meaning:</b> Successful completion. <b>Action:</b> None required.
4	—	<b>Equate Symbol:</b> IwmRetCodeWarning <b>Meaning:</b> Successful completion, unusual conditions noted.
4	xxxx0411	<b>Equate Symbol:</b> IwmRsnCodeEnclActive <b>Meaning:</b> Input enclave had 1 or more SRBs scheduled or running, or 1 or more TCBs joined to the enclave. <b>Action:</b> None required.
8	—	<b>Equate Symbol:</b> IwmRetCodeInvocError <b>Meaning:</b> Invalid invocation environment or parameters.
8	xxxx0803	<b>Equate Symbol:</b> IwmRsnCodeDisabled <b>Meaning:</b> Caller is disabled. <b>Action:</b> Avoid requesting this function while disabled.
8	xxxx0804	<b>Equate Symbol:</b> IwmRsnCodeLocked <b>Meaning:</b> Caller is locked. <b>Action:</b> Avoid requesting this function while locked.
8	xxxx080B	<b>Equate Symbol:</b> IwmRsnCodeBadPI <b>Meaning:</b> Error accessing parameter list. <b>Action:</b> Check for possible storage overlay.
8	xxxx0824	<b>Equate Symbol:</b> IwmRsnCodeAmode24 <b>Meaning:</b> Caller invoked service but was in 24 bit addressing mode. <b>Action:</b> Request this function only when you are in 31 bit addressing mode.
8	xxxx0828	<b>Equate Symbol:</b> IwmRsnCodeBadVersion <b>Meaning:</b> Version number in parameter list is not valid. <b>Action:</b> Check for possible storage overlay of the parameter list.
8	xxxx083A	<b>Equate Symbol:</b> IwmRsnCodeBadEnclave <b>Meaning:</b> Enclave token does not pass verification. <b>Action:</b> Check for possible storage overlay of the enclave token, or asynchronous events which may have deleted the enclave.
8	xxxx0872	<b>Equate Symbol:</b> IwmRsnCodeForeignEnclave <b>Meaning:</b> The enclave is foreign. <b>Action:</b> Use the IWMUIMPT macro to delete a foreign enclave.

## IWMEDELE Macro

Table 97. Return and Reason Codes for the IWM4EDEL Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
10	—	<b>Equate Symbol:</b> IwmRetCodeCompError <b>Meaning:</b> Component error. <b>Action:</b> No action required. The function may be successful if invoked again.

## Example

To allow the current task to join an Enclave:

```
        IWMEDELE  ETOKEN=ENCTOKEN,RETCODE=RC,RSNCODE=RSN
*
* Storage areas
*
ENCTOKEN DS    CL8           Contains the Enclave token
*
RC        DS    F            Return code
RSN       DS    F            Reason code
```

---

## IWMMCHST – Monitoring Change State Of Work Unit

The purpose of this service is to reflect in a monitoring environment what the current state of a work request is with respect to delays.

### Note

It is recommended to use the new equivalent service IWM4MCHS introduced with z/OS V1R6. IWM4MCHS also supports 64-bit address space. For further information refer to Chapter 70, “IWM4MCHS – Monitoring Change State Of Work Unit,” on page 587.

## Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Problem state. PSW key must either be 0 or match the value supplied on IWMMCREA.
<b>Dispatchable unit mode:</b>	Task or SRB
<b>Cross memory mode:</b>	Any PASN, any HASN, any SASN
<b>AMODE:</b>	31-bit
<b>ASC mode:</b>	Primary
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	Suspend locks are allowed.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

## Programming Requirements

1. The macro IWMYCON must be included to use this macro.
2. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
3. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded

from comparison with the reason code values described above. The constant, IWMSRNCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.

## Restrictions

1. Caller is responsible for error recovery
2. Only limited checking is done against the input monitoring token.
3. If the key specified on IWMMCREA was a user key (8-F), then the primary addressability must exist to the performance block IWMMCREA obtained. This condition is satisfied by ensuring that current primary matches primary at the time that IWMMCREA was invoked. If this service is invoked in a subspace, the condition may be satisfied by ensuring that the performance block is shared with the base space.
4. The caller must serialize to prevent any delay monitoring services from being invoked concurrently for the environment represented by the monitoring token.
5. This macro may only be used on z/OS R2 or higher levels for the following state/resources:
  - STATE(ACTIVE\_APPL)
  - RESOURCE(SSL\_THREAD)
  - RESOURCE(REG\_THREAD)
  - RESOURCE(REG\_TO\_WRKTB)
  - RESOURCE(TYPE1)
  - RESOURCE(TYPE2)
  - RESOURCE(TYPE3)
  - RESOURCE(TYPE4)
  - RESOURCE(TYPE5)
6. This macro may only be used on z/OS R8 or higher versions for the following resources:
  - RESOURCE(BUFFER\_POOL\_IO)
7. This macro may only be used on z/OS R8 or higher versions for RESTKN keyword.

## Input Register Information

Before issuing the IWMCHST macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

## Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Reason code if GR15 return code is non-zero
1	Used as work registers by the system
2-13	Unchanged
14	Used as work registers by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

## IWMMCHST Macro

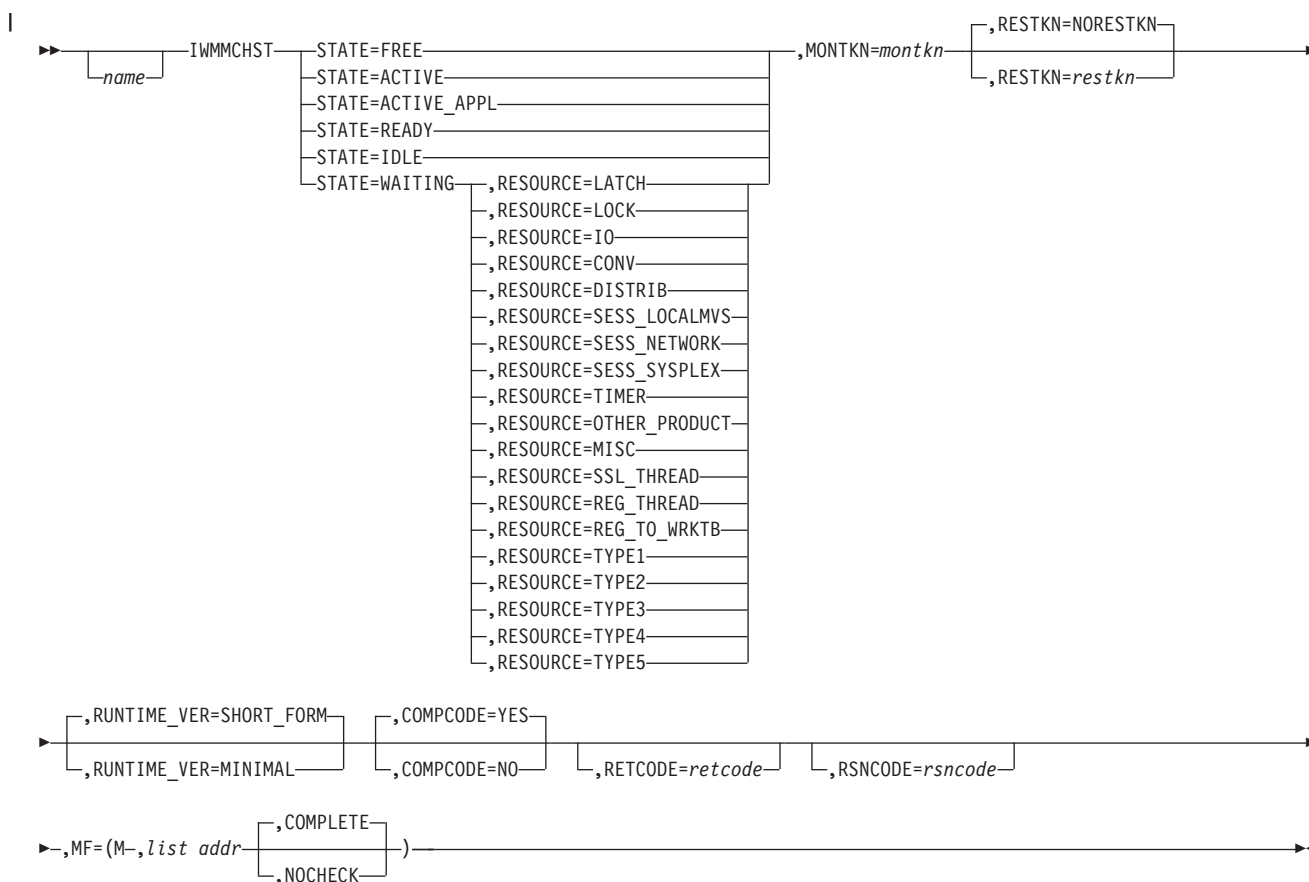
Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

## Performance Implications

None.

## Syntax

### main diagram



## Parameters

The parameters are explained as follows:

### *name*

An optional symbol, starting in column 1, that is the name on the IWMMCHST macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

### **,COMPCODE=YES**

### **,COMPCODE=NO**

An optional parameter, which indicates whether completion status for this service is needed. The default is `COMPCODE=YES`.

**,COMPCODE=YES**

indicates that completion status is needed.

**,COMPCODE=NO**

indicates that completion status is not needed. Registers 0, 15 cannot be used as reason code and return code registers upon completion of the macro expansion. For this reason neither RETCODE NOR RSNCODE may be specified when COMPCODE(NO) is specified.

**,MONTKN=*montkn***

A required input parameter, which contains the delay monitoring token

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32-bit field.

**,RESOURCE=LATCH****,RESOURCE=LOCK****,RESOURCE=IO****,RESOURCE=CONV****,RESOURCE=DISTRIB****,RESOURCE=SESS\_LOCALMVS****,RESOURCE=SESS\_NETWORK****,RESOURCE=SESS\_SYSPLEX****,RESOURCE=TIMER****,RESOURCE=OTHER\_PRODUCT****,RESOURCE=MISC****,RESOURCE=SSL\_THREAD****,RESOURCE=REG\_THREAD****,RESOURCE=REG\_TO\_WRKTB****,RESOURCE=TYPE1****,RESOURCE=TYPE2****,RESOURCE=TYPE3****,RESOURCE=TYPE4****,RESOURCE=TYPE5**

When STATE=WAITING is specified, a required parameter, which indicates the resource that the work manager is waiting for on behalf of the work request described by the monitoring environment.

**,RESOURCE=LATCH**

indicates that the work manager is waiting on a latch.

**,RESOURCE=LOCK**

indicates that the work manager is waiting on a lock.

**,RESOURCE=IO**

indicates that the work manager is waiting on an activity related to an I/O request. This may either be an actual I/O operation or some function associated with an IO request that cannot be more precisely determined by the work manager, for example, locks, buffers, etc.

**,RESOURCE=CONV**

indicates that the work manager is waiting on a conversation. This may be used in conjunction with IWMMSWCH to identify where the target is located.

**,RESOURCE=DISTRIB**

indicates that the work manager is waiting on a distributed request. This says at a high level that some function or data must be routed prior to resumption of the work request. This is to be contrasted with Waiting on

## IWMMCHST Macro

Conversation, which is a low level view of the precise resource that is needed. A distributed request could involve waiting on a conversation as part of its processing.

### **,RESOURCE=SESS\_LOCALMVS**

indicates that the work manager is waiting to establish a session somewhere in the current MVS image.

### **,RESOURCE=SESS\_NETWORK**

indicates that the work manager is waiting to establish a session somewhere in the network.

### **,RESOURCE=SESS\_SYSPLEX**

indicates that the work manager is waiting to establish a session somewhere in the sysplex.

### **,RESOURCE=TIMER**

indicates that the work request is waiting on a timer.

### **,RESOURCE=OTHER\_PRODUCT**

indicates that the work manager is waiting on another product to complete its function.

### **,RESOURCE=MISC**

indicates that the work manager is waiting on some unidentified resource, possibly among the previous categories.

### **,RESOURCE=SSL\_THREAD**

indicates that the work manager is waiting on an SSL thread.

### **,RESOURCE=REG\_THREAD**

indicates that the work manager is waiting on a regular processing thread.

### **,RESOURCE=REG\_TO\_WRKTB**

indicates that the work manager is waiting for the registration to a worktable.

### **,RESOURCE=TYPE1**

indicates that the work manager is waiting for resource type 1.

### **,RESOURCE=TYPE2**

indicates that the work manager is waiting for resource type 2.

### **,RESOURCE=TYPE3**

indicates that the work manager is waiting for resource type 3.

### **,RESOURCE=TYPE4**

indicates that the work manager is waiting for resource type 4.

### **,RESOURCE=TYPE5**

indicates that the work manager is waiting for resource type 5.

### **,RESTKN=*restkn***

### **,RESTKN=NORESTKN**

An optional input parameter, which contains the token of the managed resource previously registered with register resource (IWMMREG). The default is NORESTKN, which indicates that no resource token is provided.

NORESTKN preserves the existing resource token, if any.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

**,RETCODE=retcode**

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,RSNCODE=rsncode**

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,RUNTIME\_VER=SHORT\_FORM****,RUNTIME\_VER=MINIMAL**

An optional parameter, which indicates what level of runtime verification will be performed. The default is RUNTIME\_VER=SHORT\_FORM.

**,RUNTIME\_VER=SHORT\_FORM**

indicates that checking should verify that a monitoring environment is established and passes a short form of verification prior to being used.

**,RUNTIME\_VER=MINIMAL**

indicates that checking will only be done to verify that a monitoring environment may be established, assuming that it would be valid and useable if established.

**STATE=FREE****STATE=ACTIVE****STATE=ACTIVE\_APPL****STATE=READY****STATE=IDLE****STATE=WAITING**

A required parameter, which indicates the current state for the work request.

**STATE=FREE**

indicates that the work manager has no work request associated with the monitoring environment.

**STATE=ACTIVE**

indicates that there is a program executing on behalf of the work request described by the monitoring environment. This is an indication from the perspective of the work manager using this service, who should not try to factor in MVS decisions in preempting work, etc.

**STATE=ACTIVE\_APPL**

indicates that there is an application program executing on behalf of the work request described by the monitoring environment. This is an indication from the perspective of the work manager using this service, who should not try to factor in MVS decisions in preempting work. This state represents the application activity in contrast to the active (subsystem) state.

**STATE=READY**

indicates that there is a program ready to execute on behalf of the work request described by the monitoring environment, but the work manager has given priority to another work request.

**STATE=IDLE**

indicates that the work manager has no work requests that it is allowed to service within the monitoring environment. This represents a delay that is not under the control of the work manager itself and which it cannot eliminate. This may be caused by limits imposed by the installation or by the nature of the work request itself.

## IWMMCHST Macro

### STATE=WAITING

indicates that the work manager is waiting for a resource on behalf of the work request described by the the monitoring environment.

## ABEND Codes

None.

## Return and Reason Codes

When the IWMMCHST macro returns control to your program:

- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, if you coded RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 98. Return and Reason Codes for the IWMMCHST Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> <code>IwmRetCodeOk</code> <b>Meaning:</b> Successful completion. <b>Action:</b> None required.
4	—	<b>Equate Symbol:</b> <code>IwmRetCodeWarning</code> <b>Meaning:</b> Successful completion, unusual conditions noted.
4	xxxx0402	<b>Equate Symbol:</b> <code>IwmRsnCodeNoMonEnv</code> <b>Meaning:</b> Input monitoring token indicates no monitoring environment was established. <b>Action:</b> None required.
8	—	<b>Equate Symbol:</b> <code>IwmRetCodeInvocError</code> <b>Meaning:</b> Invalid invocation environment or parameters.
8	xxxx0820	<b>Equate Symbol:</b> <code>IwmRsnCodeBadMonEnv</code> <b>Meaning:</b> Monitoring environment does not pass short form verification. <b>Action:</b> Check for possible storage overlay.

---

## IWMMCREA – Create Delay Monitoring Environment

The purpose of this service is to create a single delay monitoring environment or a number of delay monitoring environments so that work and resource managers may utilize other delay monitoring services to reflect to MVS the execution states and delays associated with work requests.

There are three types of monitoring environments available, management monitoring environments, report-only monitoring environments and buffer pool management only environments. Management monitoring environments provide both, performance management and performance reporting. Report-only monitoring environments can be used for performance reporting only. Buffer pool management only monitoring environments provide only buffer pool performance management for enclaves.



Optionally, if you specify REPORTONLY=YES the monitoring environment is used for reporting purposes only.

Furthermore, if you specify BPMGMTONLY=YES, the monitoring environment is used for buffer pool management for enclaves only.

#### Note

It is recommended to use the new equivalent service IWM4MCRE introduced with z/OS V1R6. IWM4MCRE also supports 64-bit address space. For further information refer to Chapter 71, "IWM4MCRE – Create Delay Monitoring Environment," on page 595.

## Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Supervisor state or program key mask (PKM) allowing keys 0-7.
<b>Dispatchable unit mode:</b>	Task
<b>Cross memory mode:</b>	Non-XMEM or XMEM. Any P.S.H.
<b>AMODE:</b>	31-bit
<b>ASC mode:</b>	Primary
<b>Interrupt status:</b>	Enabled for I/O and external interrupts No (EUT) FRR established.
<b>Locks:</b>	No locks may be held.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

## Programming Requirements

1. The macro CVT must be included to use this macro.
2. The macro IWMYCON must be included to use this macro.
3. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
4. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, IWMSRNCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.
5. All character data, unless otherwise specified, is assumed to be left justified and padded with blanks on the right, as needed, to occupy the specified number of bytes.

## Restrictions

1. This macro may not be used during task/address space termination.
2. If the key specified on IWMMLCREA is a user key (8-F), then the caller must be in non-cross-memory mode (P=S=H)
3. While not a restriction for IWMMLCREA, it should be noted that when the key specified is a user key (8-F), the delay monitoring token may only be passed (to any service whatsoever), when primary addressability exists to the performance block obtained by IWMMLCREA. This condition may be satisfied by ensuring that the then current primary matches primary at the time that IWMMLCREA was

## IWMMCREA Macro

invoked. If these other services are invoked in a subspace, the condition may be satisfied by ensuring that the performance block is shared with the base space.

4. This service provides a task and address space resource managers to clean up any resources obtained during task and address space terminations. Once the calling task or address space terminates, the monitoring token returned by IWMMCREA must not be used for any services.
5. This macro supports multiple versions. Some keywords are only supported by certain versions. Refer to the PLISTVER parameter description for further information.

## Input Register Information

Before issuing the IWMMCREA macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

## Output Register Information

When control returns to the caller, the GPRs contain:

### Register

#### Contents

- 0 Reason code if GR15 return code is non-zero
- 1 Used as work registers by the system
- 2-13 Unchanged
- 14 Used as work registers by the system
- 15 Return code

When control returns to the caller, the ARs contain:

### Register

#### Contents

- 0-1 Used as work registers by the system
- 2-13 Unchanged
- 14-15 Used as work registers by the system

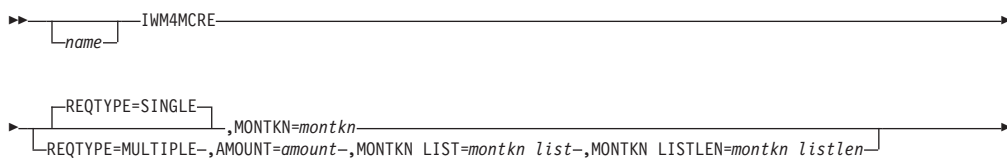
Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

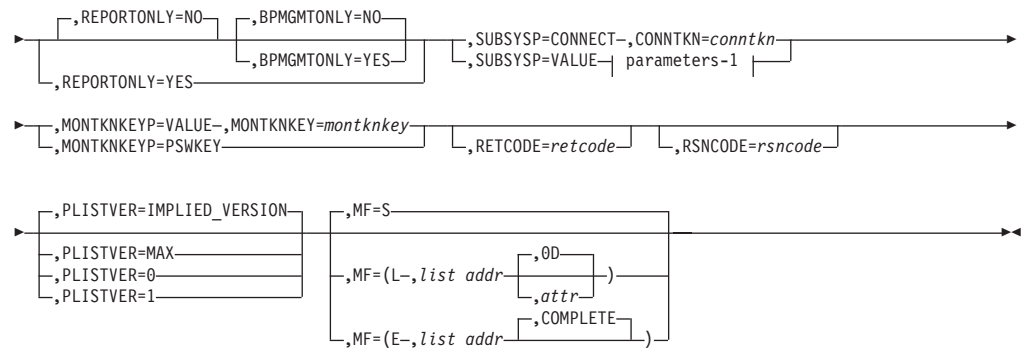
## Performance Implications

None.

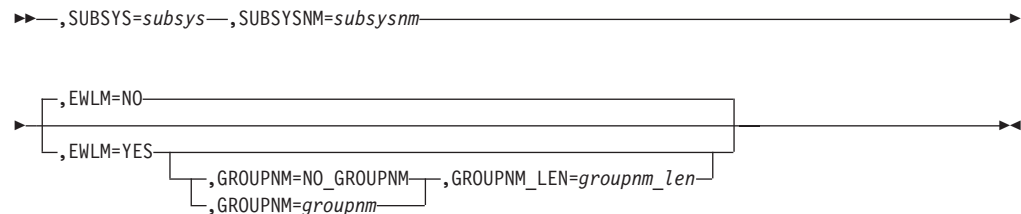
## Syntax

### main diagram





**parameters-1**



**Parameters**

The parameters are explained as follows:

*name*

An optional symbol, starting in column 1, that is the name on the IWMMCREA macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

**,AMOUNT=amount**

When REQTYPE=MULTIPLE is specified, a required input parameter, which specifies the number of delay monitoring environments to be created.

While there is no restriction on the number of delay monitoring environments to be created, caller should only create the minimum number of delay monitoring environments that are needed.

If there are too many unused delay monitoring environments existing in the system, the storage and CPU overheads may be significant.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

**,BPMGMTONLY=NO**  
**,BPMGMTONLY=YES**

When REPORTONLY=NO is specified, an optional parameter, which indicates whether the monitoring environment is for buffer pool management purposes only (YES) or (NO). The default is BPMGMTONLY=NO.

**,BPMGMTONLY=NO**

indicates that the monitoring environment is not for buffer pool management purposes.

**,BPMGMTONLY=YES**

indicates that the monitoring environment is for buffer pool management purposes only.

## IWMCREA Macro

**,CONNTKN=conntkn**

When SUBSYSP=CONNECT is specified, a required input parameter, which contains the connect token associated with the subsystem.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32 bit field.

**,EWLM=NO**

**,EWLM=YES**

IF SUBSYSP=CONNECT is specified, this parameter is not allowed. In this case the EWLM=YES/NO specification is inherited from the IWM4CON (or IWMCONN) EWLM=YES/NO specification. When SUBSYSP=VALUE is specified, an optional parameter, which indicates if the created monitoring environment is intended to participate in Enterprise Workload Management (EWLM). The default is EWLM=NO.

**,EWLM=NO**

The monitoring environment cannot be used to report on ARM work requests.

**,EWLM=YES**

The monitoring environment participates in cross-platform Enterprise Workload Management and interacts with EWLM. An ARM application instance will be registered and started using the passed subsystem type (SUBSYS), subsystem name (SUBSYSNM), and the new parameter group name (GROUPNM, GROUPNM\_LEN) - an already existing ARM registration for the same address space with identical SUBSYS, SUBSYSNM, GROUPNM and GROUPNM\_LEN parameters will be reused. All ARM work requests associated with the created monitoring environment are reported for this ARM application instance.

**,GROUPNM=groupnm**

**,GROUPNM=NO\_GROUPNM**

When EWLM=YES and SUBSYSP=VALUE are specified, an optional input parameter, which contains the name of an application group, for example, a group of similar or cooperating subsystem instances. A group name can be up to 255 characters long. Provision of a data area initialized to all blanks is equivalent to specification of NO\_GROUPNM. NO\_GROUPNM indicates that no group name is passed. This is the default.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a character field.

**,GROUPNM\_LEN=groupnm\_len**

When GROUPNM=groupnm, EWLM=YES and SUBSYSP=VALUE are specified, a required input parameter, which contains the length of the group name. A group name can be up to 255 characters long.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

**,MF=S**

**,MF=(L,list addr)**

**,MF=(L,list addr,attr)**

**,MF=(L,list addr,OD)**

**,MF=(E,list addr)**

**,MF=(E,list addr,COMPLETE)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

*,list addr*

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

*,attr*

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

**,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

**,MONTKN=*montkn***

When REQTYPE=SINGLE is specified, a required output parameter, which will receive the delay monitoring token.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32 bit field.

**,MONTKN\_LIST=*montkn\_list***

When REQTYPE=MULTIPLE is specified, a required input parameter, which specifies an area into which a list of delay monitoring tokens will be placed.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a character field.

**,MONTKN\_LISTLEN=*montkn\_listlen***

When REQTYPE=MULTIPLE is specified, a required input parameter, which specifies the length (in bytes) of the area identified by the **MONTKN\_LIST** keyword.

Size of this area must be at least the size of **MONTKN** (see **MONTKN** keyword) times **AMOUNT**. If the user specified area is not large enough to return the delay monitoring tokens, a specific return/reason code will be returned and the request will not be processed.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

**,MONTKNKEY=*montknkey***

When MONTKNKEYP=VALUE is specified, a required input parameter, which contains the key in which the delay monitoring services will be invoked subsequently when using the output MONTKN. The low order 4 bits (bits 4-7) contain the key value. The high order 4 bits (bits 0-3) must be 0.

## IWMPCREA Macro

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8 bit field.

**,MONTKNKEYP=VALUE**

**,MONTKNKEYP=PSWKEY**

A required parameter, which describes how the input key should be obtained.

**,MONTKNKEYP=VALUE**

indicates that the key is being passed explicitly via MONTKNKEY.

**,MONTKNKEYP=PSWKEY**

indicates that the current PSW key should be used.

**,PLISTVER=IMPLIED\_VERSION**

**,PLISTVER=MAX**

**,PLISTVER=0**

**,PLISTVER=1**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form, when both are assembled with the same level of the system. In this way, MAX ensures that the parameter list does not overwrite nearby storage.

- **0**, which supports all parameters except those specifically referenced in higher versions.
- **1**, which supports both the following parameters and those from version 0:

BPMGMTONLY	GROUPNM
EWLM	GROUPNM_LEN

**To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX
- A decimal value of 0, or 1

**,REPORTONLY=NO**

**,REPORTONLY=YES**

An optional parameter, which indicates whether the monitoring environment is for reporting purposes only (YES) or (NO). The default is REPORTONLY=NO.

**,REPORTONLY=NO**

indicates that the monitoring environment is for management and reporting purposes.

**,REPORTONLY=YES**

indicates that the monitoring environment is for reporting purposes only.

**REQTYPE=SINGLE**

**REQTYPE=MULTIPLE**

An optional parameter that indicates whether the request is to create a single delay monitoring environment or to create multiple delay monitoring environments. The default is REQTYPE=SINGLE.

**REQTYPE=SINGLE**

The request is to create a single delay monitoring environment.

**REQTYPE=MULTIPLE**

The request is to create a number of delay monitoring environments.

**,RETCODE=*retcode***

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,RSNCODE=*rsncode***

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,SUBSYS=*subsys***

When SUBSYSP=VALUE is specified, a required input parameter, which contains the generic subsystem type (e.g. IMS, CICS, etc.).

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 4-character field.

**,SUBSYSNM=*subsysnm***

When SUBSYSP=VALUE is specified, a required input parameter, which contains the subsystem name.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

**,SUBSYSP=CONNECT****,SUBSYSP=VALUE**

A required parameter, which describes how the calling subsystem is providing identification.

**,SUBSYSP=CONNECT**

indicates that the connect token is being passed.

**,SUBSYSP=VALUE**

indicates that the subsystem name is being passed directly.

**ABEND Codes**

None.

**Return and Reason Codes**

When the IWMMLCREA macro returns control to your program:

- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, if you coded RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 99. Return and Reason Codes for the IWM4MCRE Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> IwmRetCodeOk <b>Meaning:</b> Successful completion. <b>Action:</b> None required.
4	—	<b>Equate Symbol:</b> IwmRetCodeWarning <b>Meaning:</b> Successful completion, unusual conditions noted.
4	xxxx0401	<b>Equate Symbol:</b> IwmRsnCodeNoWLM <b>Meaning:</b> System does not support delay monitoring services. The delay monitoring token returned is useable in other services. However use of this token will NOT result in the action requested of those services. This return code is only set when the MVS release is prior to MVS/ESA SP5.1.0. <b>Action:</b> None required.
4	xxxx0409	<b>Equate Symbol:</b> IwmRsnCodeNoConn <b>Meaning:</b> Connect token does not reflect a successful Connect. The delay monitoring token returned is useable in other services. However use of this token will NOT result in the action requested of those services. <b>Action:</b> None required.
8	—	<b>Equate Symbol:</b> IwmRetCodeInvocError <b>Meaning:</b> Invalid invocation environment or parameters.
8	xxxx0801	<b>Equate Symbol:</b> IwmRsnCodeSrbMode <b>Meaning:</b> Caller is in SRB mode. <b>Action:</b> Avoid requesting this function while in SRB mode.
8	xxxx0802	<b>Equate Symbol:</b> IwmRsnCodeXmemUserKeyTkn <b>Meaning:</b> Caller is in cross-memory mode while the token was requested in user key. <b>Action:</b> Avoid requesting this function while in cross-memory mode.
8	xxxx0803	<b>Equate Symbol:</b> IwmRsnCodeDisabled <b>Meaning:</b> Caller is disabled. <b>Action:</b> Avoid requesting this function while disabled.
8	xxxx0804	<b>Equate Symbol:</b> IwmRsnCodeLocked <b>Meaning:</b> Caller is locked. <b>Action:</b> Avoid requesting this function while locked.
8	xxxx080B	<b>Equate Symbol:</b> IwmRsnCodeBadPI <b>Meaning:</b> Error accessing parameter list. <b>Action:</b> Check for possible storage overlay.
8	xxxx0810	<b>Equate Symbol:</b> IwmRsnCodeEutFrr <b>Meaning:</b> Caller has EUT FRR established. <b>Action:</b> Avoid requesting this function with an EUT FRR set.



Table 99. Return and Reason Codes for the IWM4MCRE Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0821	<p><b>Equate Symbol:</b> IwmRsnCodeBadConn</p> <p><b>Meaning:</b> Input connect token does not pass validity checking.</p> <p><b>Action:</b> Check for possible storage overlay.</p>
8	xxxx0823	<p><b>Equate Symbol:</b> IwmRsnCodeDatoff</p> <p><b>Meaning:</b> Caller invoked service while DATOFF</p> <p><b>Action:</b> Avoid requesting this function in this environment.</p>
8	xxxx0824	<p><b>Equate Symbol:</b> IwmRsnCodeAmode24</p> <p><b>Meaning:</b> Caller invoked service but was in 24 bit addressing mode.</p> <p><b>Action:</b> Request this function only when you are in 31 bit addressing mode.</p>
8	xxxx0825	<p><b>Equate Symbol:</b> IwmRsnCodeAscModeNotPrimary</p> <p><b>Meaning:</b> Caller invoked service but was not DAT on Primary ASC mode.</p> <p><b>Action:</b> Avoid requesting this function in this environment.</p>
8	xxxx0826	<p><b>Equate Symbol:</b> IwmRsnCodeTaskTerm</p> <p><b>Meaning:</b> Caller invoked service while task termination is in progress for the TCB associated with the owner.</p> <p><b>Action:</b> Avoid requesting this function in this environment.</p>
8	xxxx0827	<p><b>Equate Symbol:</b> IwmRsnCodeRsvdNot0</p> <p><b>Meaning:</b> Reserved field in parameter list was non-zero.</p> <p><b>Action:</b> Check for possible storage overlay of the parameter list.</p>
8	xxxx0828	<p><b>Equate Symbol:</b> IwmRsnCodeBadVersion</p> <p><b>Meaning:</b> Version number in parameter list is not valid.</p> <p><b>Action:</b> Check for possible storage overlay of the parameter list.</p>
8	xxxx0829	<p><b>Equate Symbol:</b> IwmRsnCodeBadOptions</p> <p><b>Meaning:</b> Parameter list omits required parameters or supplies mutually exclusive parameters or provides data associated with options not selected.</p> <p><b>Action:</b> Check for possible storage overlay of the parameter list.</p>
8	xxxx0844	<p><b>Equate Symbol:</b> IwmRsnCodeBadMonTknListLen</p> <p><b>Meaning:</b> The storage area length specified on the MONTKN_LISTLEN parameter is not large enough to contain the data being returned. No data is returned.</p> <p><b>Action:</b> Invoke the function with an output area sufficient to receive the data.</p>
C	—	<p><b>Equate Symbol:</b> IwmRetCodeEnvError</p> <p><b>Meaning:</b> Environmental error.</p>

Table 99. Return and Reason Codes for the IWM4MCRE Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
C	xxxx0C01	<p><b>Equate Symbol:</b> IwmRsnCodeNoStg</p> <p><b>Meaning:</b> Storage is not available for the request.</p> <p><b>Action:</b> There is a storage shortage. The function may work successfully at a later time.</p>
C	xxxx0C09	<p><b>Equate Symbol:</b> IwmRsnCodeNoResmgr</p> <p><b>Meaning:</b> Resource manager could not be established.</p> <p><b>Action:</b> No action required. This condition may be due to a storage shortage condition.</p>
10	—	<p><b>Equate Symbol:</b> IwmRetCodeCompError</p> <p><b>Meaning:</b> Component error.</p> <p><b>Action:</b> No action required. The function may be successful if invoked again.</p>

---

## IWMMINIT – Initialize Monitoring Environment

IWMMINIT allows the caller to supply MVS with some or all of the work request attributes needed for the monitoring environment. The attributes include userid, transaction name, transaction class, source LU, and LU 6.2 token.

There are two types of monitoring environments available, management monitoring environments and report-only monitoring environments. Management monitoring environments provide both performance management and performance reporting. Report-only monitoring environments can be used for performance reporting only.

Use the REPORTONLY=YES parameter to specify the monitoring environment will be used for reporting purposes only.

If you are invoking IWMMINIT with the REPORTONLY=YES parameter, ASSOCIATE=ENCLAVE or ASSOCIATE=ADDRESS\_SPACE must be specified to associate the monitoring environment with an enclave or an address space.

For management monitoring environments, if possible, invoke IWMMINIT immediately following IWMCLSFY, and pass the service class for the work request. Without the associated service class in the monitoring environment, delay information cannot be accumulated and reported accurately.

IWMMINIT can be issued multiple times for the same work request. The first time you invoke IWMMINIT for a work request, you must specify MODE=RESET, otherwise the previous work request's attributes are associated with this work request. Any subsequent time you invoke IWMMINIT from the same address space for the same monitoring token for the same work request, specify MODE=RETAIN. If the caller subsystem work manager consists of multiple address spaces (with multiple monitoring tokens), the first time IWMMINIT is invoked in each address space for a given work request must specify MODE=RESET. Any subsequent invocations for the same work request should specify MODE=RETAIN.

If you are invoking IWM4MINI multiple times for the same work request, only one of the invocations should specify EXSTARTTIME=exstarttime. It is up to you to decide at which point in the subsystem work manager's processing you consider the real execution start time.

Optionally, with this macro, you can use the OWNER\_TOKEN and OWNER\_DATA parameters to specify a token for the user/owner of the monitoring environment for your own use.

#### Note

It is recommended to use the new equivalent service IWM4MINI introduced with z/OS V1R6. IWM4MINI also supports 64-bit address space. For further information refer to Chapter 73, "IWM4MINI – Initialize Monitoring Environment," on page 611.

## Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Either problem state or supervisor state. PSW key must either be 0 or match the value supplied on IWMMCREA.
<b>Dispatchable unit mode:</b>	Task or SRB
<b>Cross memory mode:</b>	Any PASN, any HASN, any SASN If the key specified on IWMMCREA was a user key (8-F), then primary addressability must be the same as when IWMMCREA was invoked.
<b>AMODE:</b>	31-bit
<b>ASC mode:</b>	Primary Any P,S,H.
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	Locked or unlocked
<b>Control parameters:</b>	Control parameters must be in the primary address space.

## Programming Requirements

1. The macro IWMYCON must be included to use this macro.
2. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
3. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, IWMSNCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.

## Restrictions

1. All parameter areas must reside in current primary, except that the TCB (if specified) must reside in current home.
2. Caller is responsible for error recovery.
3. Only limited checking is done against the input monitoring token.
4. If the key specified on IWMMCREA was a user key (8-F), then the primary addressability must exist to the performance block IWMMCREA obtained. This condition is satisfied by ensuring that current primary matches primary at the

## IWMMINIT Macro

time that IWMMCREA was invoked. If this service is invoked in a subspace, the condition may be satisfied by ensuring that the performance block is shared with the base space.

5. The caller must serialize to prevent any delay monitoring services from being invoked concurrently for the environment represented by the monitoring token.
6. This macro may only be used on z/OS R2 or higher levels for REPORTONLY and ASSOCIATE keywords.
7. The BPMGMTONLY keyword may only be used on systems running z/OS R8 or higher.
8. This macro supports multiple versions. Some keywords are only supported by certain versions. Refer to the PLISTVER parameter description for further information.

## Input Register Information

Before issuing the IWMMINIT macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

## Output Register Information

When control returns to the caller, the GPRs contain:

### Register

	Contents
0	Reason code if GR15 return code is non-zero
1	Used as work registers by the system
2-13	Unchanged
14	Used as work registers by the system
15	Return code

When control returns to the caller, the ARs contain:

### Register

	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

## Performance Implications

None.

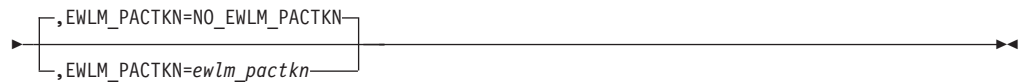
## Syntax

### main diagram

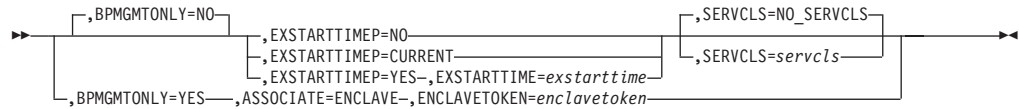




## IWMMINIT Macro



### parameters-2



## Parameters

The parameters are explained as follows:

### *name*

An optional symbol, starting in column 1, that is the name on the IWM4MINI macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

### **,ARRIVALTIME=arrivaltime**

When ARRIVALTIMEP=YES and MODE=RESET are specified, a required input parameter, which contains the work arrival time in STCK format.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 64 bit field.

### **,ARRIVALTIMEP=CURRENT**

### **,ARRIVALTIMEP=YES**

When MODE=RESET is specified, a required parameter, which indicates whether the work arrival time is passed.

### **,ARRIVALTIMEP=CURRENT**

indicates that the current time should be supplied by the service.

### **,ARRIVALTIMEP=YES**

indicates that the work arrival time is passed.

### **,ASID=asid**

When ASSOCIATE=ADDRESS\_SPACE and REPORTONLY=YES are specified, a required input parameter which contains the address space ID.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 16-bit field.

### **,ASSOCIATE=ENCLAVE**

### **,ASSOCIATE=ADDRESS\_SPACE**

When REPORTONLY=YES is specified, a required parameter, which indicates whether the monitoring environment should be associated to an enclave or an address space.

### **,ASSOCIATE=ENCLAVE**

indicates that the monitoring environment should be associated to an enclave.

### **,ASSOCIATE=ADDRESS\_SPACE**

indicates that the monitoring environment should be associated to an address space.

**,ASSOCIATE=ENCLAVE**

When BPMGMTONLY=YES and REPORTONLY=NO are specified, a required parameter, which indicates whether the monitoring environment should be associated only to an enclave

**,ASSOCIATE=ENCLAVE**

indicates that the monitoring environment should be associated to an enclave.

**,BPMGMTONLY=NO****,BPMGMTONLY=NO**

When REPORTONLY=NO is specified, an optional parameter, which indicates whether the monitoring environment is for bufferpool management purposes only (YES) or not (NO). The default is BPMGMTONLY=NO.

**,BPMGMTONLY=NO**

indicates that the monitoring environment is not for bufferpool management purposes only.

**,BPMGMTONLY=YES**

indicates that the monitoring environment is for bufferpool management purposes only.

**,CONTINUEP=YES****,CONTINUEP=NO**

A required parameter, which indicates whether it is known (YES) or not (NO) that there exists another monitoring environment for this same work request.

**,CONTINUEP=YES**

indicates that the existence of a prior monitoring environment for the work request is known.

**,CONTINUEP=NO**

indicates that it is not known whether there exists a prior monitoring environment for the work request. If MODE(RESET) is specified, no status is saved. If MODE(RETAIN) is specified, the existing status is preserved.

**,DISPTYPE=TCB****,DISPTYPE=SRB**

When MODE=RESET is specified, a required parameter, which describes the nature of the MVS dispatchable units which participate in processing work requests associated with the delay monitoring environment established by this service.

**,DISPTYPE=TCB**

indicates that work requests run in TCB mode under a TCB within the current home address space. Note that in cross-memory mode, this may be different from the current primary address space.

**,DISPTYPE=SRB**

indicates that work requests run in SRB mode within the current home address space.

**,DISPTYPE=SAVEDTYPE****,DISPTYPE=TCB****,DISPTYPE=SRB**

When MODE=RETAIN is specified, a required parameter, which describes the nature of the MVS dispatchable units which participate in processing work requests associated with the delay monitoring environment established by this service.

### **,DISPTYPE=SAVEDTYPE**

indicates that the information saved when MODE(RESET) was used is still applicable.

### **,DISPTYPE=TCB**

indicates that work requests run in TCB mode under a TCB within the current home address space. Note that in cross-memory mode, this may be different from the current primary address space.

### **,DISPTYPE=SRB**

indicates that work requests run in SRB mode within the current home address space.

### **,DURATION=EXECUTION**

### **,DURATION=BEGIN\_TO\_END**

When MODE=RESET is specified, an optional parameter, which indicates the duration of the work request over which the delays are to be represented. The default is DURATION=EXECUTION.

### **,DURATION=EXECUTION**

indicates that the monitoring environment will reflect delays from the point where an application or transaction program is given control, i.e. the execution phase. Typically a monitoring environment with this scope would be passed to IWMMNTFY to pass the execution time for the work request.

### **,DURATION=BEGIN\_TO\_END**

indicates that the monitoring environment will reflect delays from the arrival of the work request into the MVS sysplex until its completion. Ordinarily use of this option would be in close proximity to the time when the work request is classified. Typically a monitoring environment with this duration would be passed to IWMRPT to report the total elapsed time for the work request.

### **,DURATION=PREV\_VALUE**

### **,DURATION=EXECUTION**

### **,DURATION=BEGIN\_TO\_END**

When MODE=RETAIN is specified, an optional parameter, which indicates the duration of the work request over which the delays are to be represented. The default is DURATION=PREV\_VALUE.

### **,DURATION=PREV\_VALUE**

indicates that the duration for delays has been specified on a previous invocation.

### **,DURATION=EXECUTION**

indicates that the monitoring environment will reflect delays from the point where an application or transaction program is given control, for example, the execution phase. Typically a monitoring environment with this duration would be passed to IWMMNTFY to pass the execution time for the work request.

### **,DURATION=BEGIN\_TO\_END**

indicates that the monitoring environment will reflect delays from the arrival of the work request into the MVS sysplex until its completion. Ordinarily use of this option would be in close proximity to the time when the work request is classified. Typically, a monitoring environment with this duration is passed to IWMRPT to report the total elapsed time for the work request.

### **,ENCLAVETOKEN=*enclavetoken***

When ASSOCIATE=ENCLAVE and REPORTONLY=YES are specified, a required input parameter, which contains the enclave token.



**To code:** Specify the RS-type address, or address in register (2)-(12), of a 64-bit field.

| **,EWLM\_PACORR=*ewlm\_pacorr***

| **,EWLM\_PACORR=NO\_EWLM\_PACORR**

| When MODE=RESET is specified, an optional input parameter, which contains  
| the cross-platform enterprise workload management (EWLM) parent correlator  
| associated with the work request.

| **Note:**

- | • If EWLM\_PACORR is specified and the correlator does not contain a  
| valid ARM correlator, reason code `IwmRsnCodeInvalidEWLMCorr` is  
| returned to the caller. Refer to Table 78 on page 624 for further  
| information. If the correlator is a valid ARM correlator, but cannot be  
| understood by EWLM (no EWLM format), it is ignored.
- | • The parameters EWLM\_PACORR and EWLM\_PACTKN are mutually  
| exclusive.

| The default is NO\_EWLM\_PACORR. It indicates that no EWLM parent  
| correlator is passed.

| **To code:** Specify the RS-type address, or address in register (2)-(12), of a  
| character field.

| **,EWLM\_PACTKN=*ewlm\_pactkn***

| **,EWLM\_PACTKN=NO\_EWLM\_PACTKN**

| When MODE=RESET is specified, an optional input parameter, which contains  
| the cross-platform Enterprise Workload Management (EWLM) parent correlator  
| token associated with the work request.

| If EWLM\_PACTKN is specified and the correlator token does not contain a valid  
| correlator token, reason code `IwmRsnCodeInvalidEWLMCorr` is returned to the  
| caller. Refer to Table 78 on page 624 for further information. The default is  
| NO\_EWLM\_PACTKN. It indicates that no EWLM correlator token is passed.

| **To code:** Specify the RS-type address, or address in register (2)-(12), of a  
| character field.

| **,EXSTARTTIME=*exstarttime***

| When EXSTARTTIMEP=YES, BPMGMTONLY=NO and REPORTONLY=NO are  
| specified, a required input parameter, which contains the start execution time in  
| STCK format.

| **To code:** Specify the RS-type address, or address in register (2)-(12), of a  
| 64-bit field.

| **,EXSTARTTIMEP=NO**

| **,EXSTARTTIMEP=CURRENT**

| **,EXSTARTTIMEP=YES**

| When BPMGMTONLY=NO and REPORTONLY=NO are specified, a required  
| parameter, which indicates whether the execution start time value is passed.

| **,EXSTARTTIMEP=NO**

| indicates that the execution start time value is not passed.

| If MODE(RETAIN) is specified, EXSTARTTIMEP(NO) preserves the existing  
| execution start time, if any.

| **,EXSTARTTIMEP=CURRENT**

| indicates that the current time should be supplied by the service.

## IWMMINIT Macro

**,EXSTARTTIMEP=YES**  
indicates that the start execution time value is passed.

**,FROM=NONE**  
**,FROM=LOCALMVS**  
**,FROM=SYSPLEX**  
**,FROM=NETWORK**  
When CONTINUEP=YES is specified, a required parameter.

**,FROM=NONE**  
indicates that there is no other environment.

**,FROM=LOCALMVS**  
indicates that such an environment should exist on the current MVS.

**,FROM=SYSPLEX**  
indicates that such an environment should exist in the current syplex, but is not expected to be on the current MVS image.

**,FROM=NETWORK**  
indicates that such an environment may exist, but is not expected to be in the current sysplex.

**,LU62TKN=*lu62tkn***  
**,LU62TKN=NO\_LU62TKN**  
An optional input parameter, which contains LU 6.2 token for the work request. This is not an SNA term, but it is comprised of the following fields which are defined by SNA for the FMH5.

- Logical Unit of Work Identifier length byte, in binary, which may have the values 0 or 10-26 decimal (inclusive)
- Logical Unit of Work Identifier
  - Length byte for the network qualified LU name, in binary, which may have the values 1-17 decimal (inclusive)
  - Network qualified LU network name (1-17 bytes)
  - Logical Unit of Work Instance Number, in binary (6 bytes)
  - Logical Unit of Work Sequence Number, in binary (2 bytes)
- Conversation Correlator Field (0 to 9 bytes)
  - Length byte for the Conversation Correlator, in binary, which may have the values 0-8 decimal (inclusive)
  - Conversation Correlator of the sending transaction (1-8 bytes)

The Conversation Correlator Field (which includes its length byte) may be dropped when its length byte is 0. The default is NO\_LU62TKN. indicates that no LU 6.2 token was passed.

If MODE(RETAIN) is specified, NO\_LU62TKN will preserve the existing LU6.2 token, if any.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a character field.

**,LU62TKN\_FMT=LU\_NO\_CC\_27**  
**,LU62TKN\_FMT=FULL\_LU\_NO\_CC\_27**  
**,LU62TKN\_FMT=FULL\_LU\_0\_CC\_28**  
**,LU62TKN\_FMT=FULL\_LU\_CC\_36**  
**,LU62TKN\_FMT=OTHER**  
When LU62TKN=*lu62tkn* is specified, a required parameter, which indicates the format/length of the LU 6.2 token.

**,LU62TKN\_FMT=LU\_NO\_CC\_27**  
indicates that a fixed length token of 27 bytes is provided, with no

conversation correlator (not even its length byte). The LU name may be 1-17 bytes. Bytes at the end of the token are padded with hexadecimal zeros, if necessary, to form a full 27 bytes.

**,LU62TKN\_FMT=FULL\_LU\_NO\_CC\_27**

indicates that the fully qualified LU name (17 bytes) is used, but no conversation correlator (not even its length byte) is provided. This format is architected to be 27 bytes long.

**,LU62TKN\_FMT=FULL\_LU\_0\_CC\_28**

indicates that the fully qualified LU name (17 bytes) is used, and the conversation correlator length byte is present and has the value 0. This format is architected to be 28 bytes long.

**,LU62TKN\_FMT=FULL\_LU\_CC\_36**

indicates that the fully qualified LU name (17 bytes) is used, and the conversation correlator is provided with a length of 8 (maximum allowed). This format is architected to be 36 bytes long.

**,LU62TKN\_FMT=OTHER**

indicates that the format of the LU 6.2 token is different from those specified by the remaining keywords.

**,LU62TKN\_LEN=*lu62tkn\_len***

When LU62TKN\_FMT=OTHER and LU62TKN=*lu62tkn* are specified, a required input parameter, which contains the length of the LU62 token. Valid values are in the range 1-36 decimal (inclusive).

**To code:** Specify the RS-type address, or address in register (2)-(12), of an one-byte field.

**,MODE=RESET**

**,MODE=RETAIN**

A required parameter, which indicates how previous attributes established for a monitoring environment should be treated. This does not refer to (or include) attributes established in IWM4MCRE.

**,MODE=RESET**

indicates that previous attributes should be discarded.

**,MODE=RETAIN**

indicates that previous attributes should be retained unless explicitly specified.

**MONTKN=*montkn***

A required input parameter, which contains the delay monitoring token

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32-bit field.

**,OWNER\_DATA=*owner\_data***

**,OWNER\_DATA=NO\_OWNER\_DATA**

An optional input parameter, which contains data maintained by the user/owner of the monitoring environment. The format is undefined to MVS. The default is NO\_OWNER\_DATA which indicates that no owner data is provided.

If MODE(RETAIN) is specified, NO\_OWNER\_DATA will preserve the existing owner data, if any.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32 bit field.

**,OWNER\_TOKEN=*owner\_token***

## IWMMINIT Macro

### **,OWNER\_TOKEN=NO\_OWNER\_TOKEN**

An optional input parameter, which contains a token maintained by the user or owner of the monitoring environment. The format is undefined to MVS. The default is NO\_OWNER\_TOKEN which indicates that no owner token is provided.

If MODE(RETAIN) is specified, NO\_OWNER\_TOKEN preserves the existing owner token, if any.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32-bit field.

### **,PLISTVER=IMPLIED\_VERSION**

#### **,PLISTVER=MAX**

#### **,PLISTVER=0**

#### **,PLISTVER=1**

An optional input parameter that specifies the version of the macro. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want to indicate the latest version currently possible.
- **0**, which supports all parameters except those specifically referenced in higher versions.
- **1**, which supports both the following parameters and those from version 0:

BPMGMTONLY

EWLM\_PACORR

EWLM\_PACTKN

**To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX
- A decimal value of 0, or 1

### **REPORTONLY=NO**

### **REPORTONLY=YES**

An optional parameter, which indicates whether the monitoring environment is for reporting purposes only (YES) or not (NO). The default is REPORTONLY=NO.

#### **,REPORTONLY=NO**

indicates that the monitoring environment is not for reporting purposes only.

#### **,REPORTONLY=YES**

indicates that the monitoring environment is for reporting purposes only.

### **,RETCODE=*retcode***

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

### **,RSNCODE=*rsncode***

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

### **,SCOPE=SHARED**

### **,SCOPE=SINGLE**

A required parameter, which indicates the scope of work passed.

**,SCOPE=SHARED**

indicates that multiple work requests, possibly from different service classes, could be described.

**,SCOPE=SINGLE**

indicates that only a single work request is described.

**,SERVCLS=*servcls*****,SERVCLS=NO\_SERVCLS**

When **BPMGMTONLY=NO** and **REPORTONLY=NO** are specified, an optional input parameter, which contains the service class token. The default is **NO\_SERVCLS** which indicates that no service class token was passed.

If **MODE(RETAIN)** is specified, **NO\_SERVCLS** preserves the existing service class token, if any.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32-bit field.

**,SOURCELU=*sourcelu*****,SOURCELU=NO\_SOURCELU**

An optional input parameter, which contains the LU name associated with the requestor. This may be the fully qualified **NETID.LUNAME**, for example, network name (1-8 bytes), followed by a period, followed by the LU name for the requestor (1-8 bytes). It may also be the 1-8 byte local LU name, with no network qualifier. The **SOURCELU** field may be from 1-17 characters. In the assembler form, the macro determines the length of the field as follows:

1. If the field is specified by register notation, it will be assumed to be 17 characters (padded with blanks) and a full 17 characters will be copied.
2. If the field is specified using an RS form name, then the length will be determined using the L' assembler function. When the length is less than 17 characters, the macro will pad with blanks. When the length is greater than or equal to 17 characters, the macro copies the first 17 bytes.

In the PL/AS form, the rules for the PL/AS compiler determines when to pad with blanks, for example, less than 17 characters implies padding, 17 or more implies a 17 character copy.

This is intended to be the same value as used in **IWMCLSFY**, and may be distinct from the LU name contained within the LU 6.2 token. For environments where the LU name may be available on some, but not all flows, provision of a data area initialized to all blanks is equivalent to specification of **NO\_SOURCELU** when **MODE(RESET)** is specified. Providing an area of all blanks when **MODE(RETAIN)** is specified will cause that to be used. The default is **NO\_SOURCELU** which indicates that no source LU name was passed.

If **MODE(RETAIN)** is specified, **NO\_SOURCELU** preserves the existing source LU name, if any.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a character field.

**,TCB=*tcb*****,TCB=NO\_TCB**

When **DISPTYPE=TCB** and **MODE=RESET** are specified, an optional input parameter, which defines the TCB within the current home address space which will serve the work request. Note that this name is not the pointer to the TCB, but the name of the data area containing the TCB. A typical invocation might replace **xTCB** with **TCB**.

## IWMMINIT Macro

Generally, the input TCB specified should be the TCB under which the work request (for, example, a transaction program) runs and under which the delay information is recorded (in spite of the fact that task switches may occur). The default is NO\_TCB which indicates that no TCB is currently associated with the monitoring environment for this work request.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a character field.

**,TCB=*tcb***

**,TCB=NO\_TCB**

When DISPTYPE=TCB and MODE=RETAIN are specified, an optional input parameter, which defines the TCB within the current home address space which will serve the work request. Note that this name is not the pointer to the TCB, but the name of the data area containing the TCB. A typical invocation might replace xTCB with TCB.

Ordinarily the input TCB specified should be the TCB under which the work request (e.g. transaction program) runs and under which the delay information is recorded (in spite of the fact that task switches may occur). The default is NO\_TCB which indicates that no TCB is currently associated with the monitoring environment for this work request.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a character field.

**,TRXCLASS=*trxclass***

**,TRXCLASS=NO\_TRXCLASS**

An optional input parameter, which contains a class name within a subsystem. For environments where the transaction class is available on some, but not all flows, provision of a data area initialized to all blanks is equivalent to specification of NO\_TRXCLASS when MODE(RESET) is specified. Providing an area of all blanks when MODE(RETAIN) is specified will cause that to be used. The default is NO\_TRXCLASS. indicates that no class name was passed.

If MODE(RETAIN) is specified, NO\_TRXCLASS will preserve the existing transaction class, if any.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

**,TRXNAME=*trxname***

**,TRXNAME=NO\_TRXNAME**

An optional input parameter, which contains the transaction name. For environments where the transaction name is available on some, but not all flows, provision of a data area initialized to all blanks is equivalent to specification of NO\_TRXNAME when MODE(RESET) is specified. Providing an area of all blanks when MODE(RETAIN) is specified will cause that to be used. The default is NO\_TRXNAME. indicates that no transaction name was passed.

If MODE(RETAIN) is specified, NO\_TRXNAME will preserve the existing transaction name, if any.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

**,TTRACETOKEN=*ttracetoken***

**,TTRACETOKEN=NO\_TTRACETOKEN**

An optional input parameter, which contains the transaction trace token. The default is NO\_TTRACETOKEN which indicates that no transaction trace token was passed.

If MODE(RETAIN) is specified, NO\_TTRACETOKEN preserves the existing transaction trace token, if any.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32-character field.

**,USERID=userid**

**,USERID=NO\_USERID**

An optional input parameter, which contains the local userid associated with the work request. For environments where the user id is available on some, but not all flows, provision of a data area initialized to all blanks is equivalent to specification of NO\_USERID when MODE(RESET) is specified. Providing an area of all blanks when MODE(RETAIN) is specified will cause that to be used. The default is NO\_USERID. indicates that no userid was passed.

If MODE(RETAIN) is specified, NO\_USERID preserves the existing user id, if any.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

## ABEND Codes

None.

## Return and Reason Codes

When the IWMMINIT macro returns control to your program:

- GPR 15 (and *retcode*, when you code RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, when you code RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 100. Return and Reason Codes for the IWM4MINI Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	IwmRetCodeOk: Successful completion.
4	—	IwmRetCodeWarning: Successful completion, unusual conditions noted.
4	xxxx0402	<b>Equate Symbol:</b> IwmRsnCodeNoMonEnv <b>Meaning:</b> Monitoring token indicates that no monitoring environment exists.
8	—	IwmRetCodeInvocError: Invalid invocation environment or parameters
8	xxxx081E	<b>Equate Symbol:</b> IwmRsnCodeBadLU62TknLen <b>Meaning:</b> The length byte of the LU62 token has an invalid value. Only values 1-36 (decimal) are valid.
8	xxxx0820	<b>Equate Symbol:</b> IwmRsnCodeBadMonEnv: <b>Meaning:</b> Monitoring environment does not pass short form verification.

## IWMMINIT Macro

Table 100. Return and Reason Codes for the IWM4MINI Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0894	<p><b>Equate Symbol:</b> IwmRsnCodeInvalidEWLMCorr</p> <p><b>Meaning:</b> Passed correlator information (EWLM_PACORR or EWLM_PACTKN) does not pass validity checking, that means: the architected ARM correlator length field in the first two Bytes of the correlator (token) is either less than 4 ('0004'x) or greater than 512 ('0200'x).</p> <p><b>Action:</b> Check the specification of the correlator information.</p>
8	xxxx0895	<p><b>Equate Symbol:</b> IwmRsnCodeEWLMSErvNotEnabled</p> <p><b>Meaning:</b> Service is not enabled because the Monitoring Environment was not created with EWLM=YES (either on IWM4CON or IWM4MCRE).</p> <p><b>Action:</b> Specify the parameter EWLM_PACORR or EWLM_PACTKN only when the Monitoring Environment was created with EWLM=YES (either on IWM4CON or IWM4MCRE).</p>
C	—	IwmRetCodeEnvError: Environmental error
C	xxxx0C07	<p><b>Equate Symbol:</b> IwmRsnCodeNoArrTime:</p> <p><b>Meaning:</b> No arrival time was supplied to the service and STCK gave a non-zero condition code.</p>
C	xxxx0C08	<p><b>Equate Symbol:</b> IwmRsnCodeNoExTime:</p> <p><b>Meaning:</b> No execution start time was supplied to the service and STCK gave a non-zero condition code.</p>

## Examples

```
IWMMINIT MONTKN=(R9),ARRIVALTIMEP=YES,
ARRIVALTIME=(R3),EXSTARTTIMEP=YES,
EXSTARTTIME=(R4),DISPTYPE=TCB,TCB=(R7),
SCOPE=SINGLE,TRXNAME=WLTRXNAME,SOURCELU=SOURCELU,
CONTINUEP=YES,LU62TKN_FMT=OTHER,LU62TKN_LEN=LU62TKNLEN,
LU62TKN=LU62TKN1,MODE=RESET,FROM=NONE,
REPORTONLY=NO,RETCODE=RCODE,RSNCODE=RSN
```

## IWMQDEL – Deleting a Request from the Queue for An Execution Address Space

This service deletes a work request that was previously inserted using the IWMQINS service, if it has not been selected using the IWMSSEL service.

### Note

It is recommended to use the new equivalent service IWM4QDE introduced with z/OS V1R6. IWM4QDE also supports 64-bit address space. For further information refer to Chapter 75, "IWM4QDE – Deleting a Request from the Queue for An Execution Address Space," on page 635.



## Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Supervisor state or program key mask (PKM) allowing keys 0-7.
<b>Dispatchable unit mode:</b>	Task or SRB
<b>Cross memory mode:</b>	Any HASN, any SASN. PASN must be the address space which connected to WLM (i.e. the address space that was home when IWMCONN was issued for Queue_Manager=Yes, or Server_Manager=Yes with Server_Type=Queue).
<b>AMODE:</b>	31-bit
<b>ASC mode:</b>	Primary
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks may be held.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

## Programming Requirements

1. The macro CVT must be included to use this macro.
2. The macro IWMYCON must be included to use this macro.
3. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
4. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, IWMSRNCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.

## Restrictions

None.

## Input Register Information

Before issuing the IWMQDEL macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

## Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Reason code if GR15 return code is non-zero
1	Used as work registers by the system
2-13	Unchanged
14	Used as work registers by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged

## IWMQDEL Macro

**14-15** Used as work registers by the system

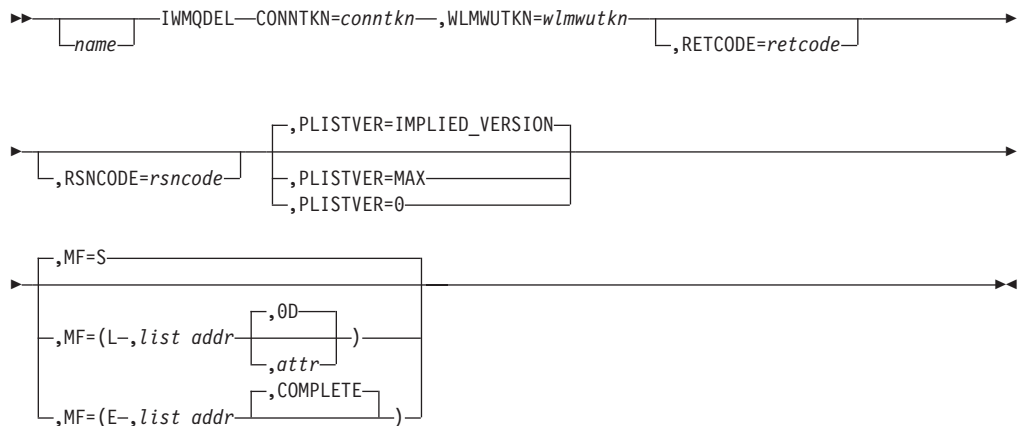
Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

## Performance Implications

None.

## Syntax

### main diagram



## Parameters

The parameters are explained as follows:

### *name*

An optional symbol, starting in column 1, that is the name on the IWMQDEL macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

### **CONNTKN=conntkn**

A required input parameter, which contains the connect token associated with the use of WLM Work Queuing services as returned by IWMCONN (specifying Queue\_Manager=Yes, or Server\_Manager=Yes with Server\_Type=Queue).

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32 bit field.

### **,MF=S**

**,MF=(L,list addr)**

**,MF=(L,list addr,attr)**

**,MF=(L,list addr,0D)**

**,MF=(E,list addr)**

**,MF=(E,list addr,COMPLETE)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

**,list addr**

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

**,attr**

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

**,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

**,PLISTVER=IMPLIED\_VERSION**

**,PLISTVER=MAX**

**,PLISTVER=0**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form, when both are assembled with the same level of the system. In this way, MAX ensures that the parameter list does not overwrite nearby storage.

- **0**, if you use the currently available parameters.

**To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX
- A decimal value of 0

**,RETCODE=retcode**

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

## IWMQDEL Macro

**,RSNCODE=***rsncode*

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,WLMWUTKN=***wlmwutkn*

A required input parameter, specifying the work unit to be deleted. This token must be a token that was returned on a prior IWMQINS request.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 16-character field.

## ABEND Codes

None.

## Return and Reason Codes

When the IWMQDEL macro returns control to your program:

- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, if you coded RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 101. Return and Reason Codes for the IWMQDEL Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> <code>IwmRetCodeOk</code> <b>Meaning:</b> Successful completion. <b>Action:</b> None required.
4	—	<b>Equate Symbol:</b> <code>IwmRetCodeWarning</code> <b>Meaning:</b> Successful completion, unusual conditions noted.
4	xxxx0408	<b>Equate Symbol:</b> <code>IwmRsnCodeWorkNotFound</code> <b>Meaning:</b> No work matching the input search criteria was found. <b>Action:</b> None required.
8	—	<b>Equate Symbol:</b> <code>IwmRetCodeInvocError</code> <b>Meaning:</b> Invalid invocation environment or parameters.
8	xxxx0803	<b>Equate Symbol:</b> <code>IwmRsnCodeDisabled</code> <b>Meaning:</b> Caller is disabled. <b>Action:</b> Avoid requesting this function while disabled.
8	xxxx0804	<b>Equate Symbol:</b> <code>IwmRsnCodeLocked</code> <b>Meaning:</b> Caller is locked. <b>Action:</b> Avoid requesting this function while locked.

Table 101. Return and Reason Codes for the IWMQDEL Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx080B	<p><b>Equate Symbol:</b> IwmRsnCodeBadPI</p> <p><b>Meaning:</b> Error accessing parameter list.</p> <p><b>Action:</b> Check for possible storage overlay. Also check if you call this macro in 64-bit address mode. Refer to the description of reason code xxxx089E for further information.</p>
8	xxxx0821	<p><b>Equate Symbol:</b> IwmRsnCodeBadConn</p> <p><b>Meaning:</b> Input connect token does not pass validity checking.</p> <p><b>Action:</b> Make sure to use the connect token returned by the IWMCONN service requesting Queue_Manager=Yes, or Server_Manager=Yes with Server_Type=Queue.</p>
8	xxxx0823	<p><b>Equate Symbol:</b> IwmRsnCodeDatoff</p> <p><b>Meaning:</b> Caller invoked service while DATOFF</p> <p><b>Action:</b> Avoid requesting this function in this environment.</p>
8	xxxx0824	<p><b>Equate Symbol:</b> IwmRsnCodeAmode24</p> <p><b>Meaning:</b> Caller invoked service but was in 24 bit addressing mode.</p> <p><b>Action:</b> Request this function only when you are in 31 bit addressing mode.</p>
8	xxxx0825	<p><b>Equate Symbol:</b> IwmRsnCodeAscModeNotPrimary</p> <p><b>Meaning:</b> Caller invoked service but was not DAT on Primary ASC mod</p> <p><b>Action:</b> Avoid requesting this function in this environment.</p>
8	xxxx0827	<p><b>Equate Symbol:</b> IwmRsnCodeRsvdNot0</p> <p><b>Meaning:</b> Reserved field in parameter list was non-zero.</p> <p><b>Action:</b> Check for use of keywords that are not supported by the MVS release on which the program is running.</p>
8	xxxx0828	<p><b>Equate Symbol:</b> IwmRsnCodeBadVersion</p> <p><b>Meaning:</b> Version number in parameter list is not valid.</p> <p><b>Action:</b> Check for possible storage overlay of the parameter list.</p>
8	xxxx083F	<p><b>Equate Symbol:</b> IwmRsnCodePrimaryNotOwnConn</p> <p><b>Meaning:</b> Primary address space does not own the passed connect token.</p> <p><b>Action:</b> Avoid requesting this function while primary address space does not own the connect token.</p>
8	xxxx0840	<p><b>Equate Symbol:</b> IwmRsnCodeServiceNotEnabled</p> <p><b>Meaning:</b> Caller's space connection is not enabled for this service</p> <p><b>Action:</b> Make sure that Queue_Manager=Yes, or Server_Manager=Yes with Server_Type=Queue is specified on the IWMCONN request to enable this service.</p>

## IWMQDEL Macro

Table 101. Return and Reason Codes for the IWMQDEL Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0848	<p><b>Equate Symbol:</b> IwmRsnCodeBadWorkUnitToken</p> <p><b>Meaning:</b> The work unit token is not valid.</p> <p><b>Action:</b> Check the specification of the WLMWUTKN parameter.</p>
8	xxxx089E	<p><b>Equate Symbol:</b> IwmRsnCodeServiceAModeMismatch</p> <p><b>Meaning:</b> The caller is in 64-bit address mode and tried to invoke a service macro that is only enabled for a 31-bit environment.</p> <p><b>Action:</b> Use the 64-bit enabled service macro (IWM4QDE) or change the address mode of the caller to 31-bit.</p>
10	—	<p><b>Equate Symbol:</b> IwmRetCodeCompError</p> <p><b>Meaning:</b> Component error.</p> <p><b>Action:</b> Contact your system programmer.</p>

## Example

To delete a work request from the WLM queue manager queues:

```

IWMQDEL  CONNTKN=CONNTOKEN,          X
          WLMWUTKN=WLMWUTKN, RETCODE=RC, RSNCODE=RSN
*
* Storage areas
*
CONNTOKEN DS   FL4           Contains the connect token
*                               associated with the use of WLM
*                               Queuing services as returned by
*                               IWMCONN
*                               (specifying QUEUE_MANAGER=YES
*                               or SERVER_MANAGER=YES
*                               SERVER_TYPE=QUEUE
WLMWUTKN DS   CL16          Work unit token
RC         DS   F           Return code
RSN        DS   F           Reason code

```

---

## IWMQINS – Inserting a Request Onto the Queue for An Execution Address space

The IWMQINS service inserts a work request onto workload management queues so its execution in a server address space can be managed by WLM.

Before using this service, the caller must connect to WLM using the IWMCONN service, specifying Queue\_Manager=Yes, or Server\_Manager=Yes with Server\_Type=Queue.

The IWMQINS service requires the use of enclaves to manage the performance goals and reporting of work. It requires the use of application environments to associate types of work requests with servers capable of processing them.

**Note**

It is recommended to use the new equivalent service IWM4QIN introduced with z/OS V1R6. IWM4QIN also supports 64-bit address space. For further information refer to Chapter 76, “IWM4QIN – Inserting a Request Onto the Queue for An Execution Address space,” on page 641.

## Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Supervisor state or program key mask (PKM) allowing keys 0-7.
<b>Dispatchable unit mode:</b>	Task or SRB
<b>Cross memory mode:</b>	Any HASN, any SASN. PASN must be the address space which connected to WLM (i.e. the address space that was home when IWMCONN was issued for Queue_Manager=Yes, or Server_Manager=Yes with Server_Type=Queue).
<b>AMODE:</b>	31-bit
<b>ASC mode:</b>	Primary
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks may be held.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

## Programming Requirements

1. The macro CVT must be included to use this macro.
2. The macro IWMYCON must be included to use this macro.
3. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
4. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, IWMSRNCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.
5. All character data, unless otherwise specified, is assumed to be left justified and padded with blanks on the right, as needed, to occupy the specified number of bytes.

## Restrictions

None.

## Input Register Information

Before issuing the IWMQINS macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

## Output Register Information

When control returns to the caller, the GPRs contain:

## IWMQINS Macro

Register	Contents
0	Reason code if GR15 return code is non-zero
1	Used as work registers by the system
2-13	Unchanged
14	Used as work registers by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

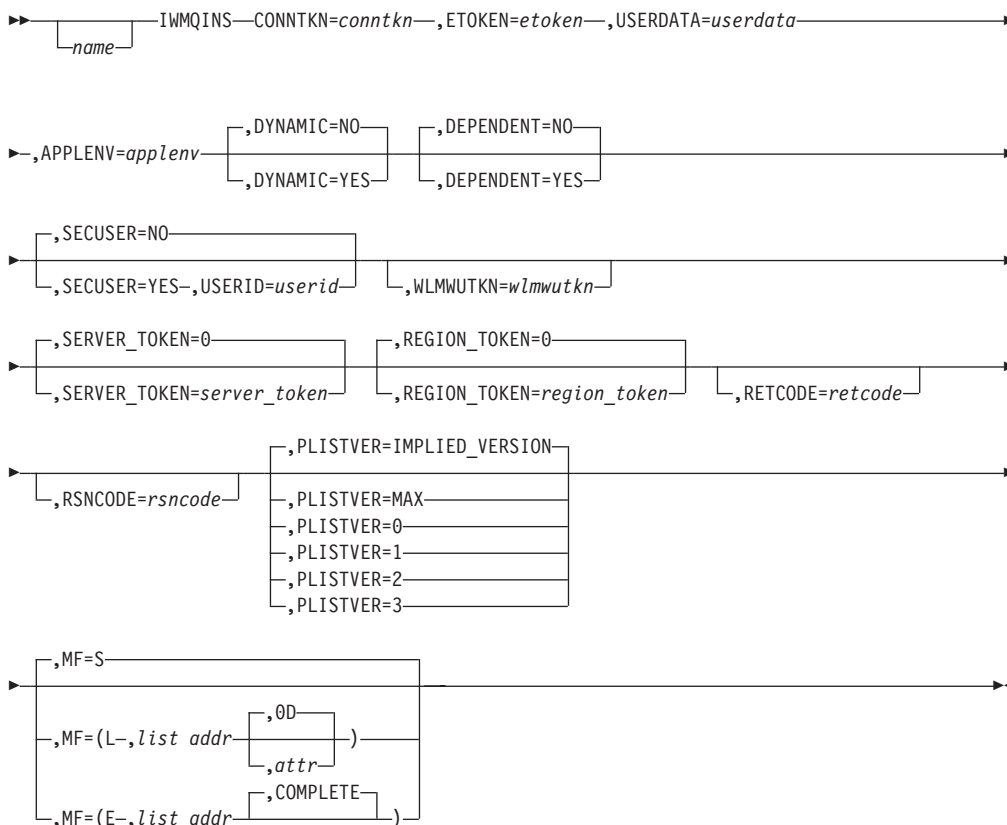
Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

## Performance Implications

None.

## Syntax

### main diagram





## Parameters

The parameters are explained as follows:

*name*

An optional symbol, starting in column 1, that is the name on the IWMQINS macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

**,APPLENV=applenv**

A required input parameter, which contains an application environment name. An application environment is defined in the workload manager service definition and instructs WLM how to create a server address space.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32-character field.

**CONNTKN=conntkn**

A required input parameter, which contains the connect token returned by the IWMCNN macro.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32 bit field.

**,DEPENDENT=NO**

**,DEPENDENT=̄YES**

An optional parameter indicating whether the insert is for a dependent or a standard request. The default is DEPENDENT=NO.

**,DEPENDENT=NO**

The request is for an independent (standard) work request (default).

**,DEPENDENT=YES**

The insert is for a dependent work request which is required by already active server tasks to complete their processing. The request is prioritized above requests which are not marked as dependent.

**,DYNAMIC=NO**

**,DYNAMIC=̄YES**

An optional parameter indicating whether the insert is for a dynamic or static application environment. The default is DYNAMIC=NO.

**,DYNAMIC=NO**

The server manager connects to a static application environment according to the WLM service definition. This is the default.

**,DYNAMIC=YES**

The server manager connects to a dynamic application environment according to a prior definition via IWMAEDEF (IWM4AEDF for 64-bit environments) service.

**,ETOKEN=etoken**

A required input parameter, which contains the enclave token associated with the work request. An enclave token is obtained using either the IWMECREA or IWMEQRY macro.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

**,MF=S**

**,MF=(L,list addr)**

**,MF=(L,list addr,attr)**

**,MF=(L,list addr,OD)**

**,MF=(E,*list addr*)**

**,MF=(E,*list addr*,**COMPLETE**)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

**,*list addr***

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

**,*attr***

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

**,**COMPLETE****

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

**,PLISTVER=IMPLIED\_VERSION**

**,PLISTVER=MAX**

**,PLISTVER=0**

**,PLISTVER=1**

**,PLISTVER=2**

**,PLISTVER=3**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form, when both are assembled with the same level of the system. In this way, MAX ensures that the parameter list does not overwrite nearby storage.

- **0**, if you use the currently available parameters.

- **1**, which supports both the following parameters and those from version 0:

SERVER\_TOKEN

- **2**, which supports both the following parameters and those from version 0 and 1:

REGION\_TOKEN

- **3**, which supports both the following parameters and those from version 0, 1, and 2:

DYNAMIC

**To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX
- A decimal value of 0, 1, 2, or 3

**,REGION\_TOKEN=***region\_token*

**,REGION\_TOKEN=0**

An optional input parameter, which contains a region token returned by the IWMSEL macro. Use REGION\_TOKEN to queue a work request to a specific server region. Such a work request is considered to be part of a set of work requests which all need access to the same status information which is kept in the virtual storage of the server region.

The following qualifications apply when specifying a region token:

- The application is responsible for passing the region token to the queueing manager so that it can insert the work request to the region.
- WLM has to know that temporal affinities for work requests to a specific server region exist in order not to stop the server region. The application must use the IWMTAFF macro to tell WLM when a temporal affinity starts and when it ends.

Coding REGION\_TOKEN=0 is equivalent to omitting the REGION\_TOKEN keyword. The default is 0.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 16-character field.

**,RETCODE=***retcode*

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,RSNCODE=***rsncode*

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,SECUSER=NO**

**,SECUSER=YES**

An optional parameter, which specifies whether the security environment of the user should be associated with the request at run time. The default is SECUSER=NO.

## IWMQINS Macro

### **,SECUSER=NO**

No security environment to be established.

### **,SECUSER=YES**

Use the specified userid to establish a security environment.

### **,SERVER\_TOKEN=*server\_token***

### **,SERVER\_TOKEN=0**

An optional input parameter, which contains a server token returned by the IWMSSSEL macro. Use SERVER\_TOKEN to queue a secondary work request to the same server task that selected a prior work request. A secondary work request is considered to be an extension of the prior work request.

The following qualifications apply when specifying a server token:

- The server task is responsible for passing the server token to the queueing manager so that it can insert a secondary work request.
- Coordination is required between the queueing manager and the server task so that the server task knows when to expect secondary work requests. The server task uses the IWMSSEM macro to select secondary work requests. It must select all secondary work requests before it can resume normal selection using IWMSSSEL.
- The same application environment and enclave token passed for the original work request must be passed for each secondary work request.
- A secondary work request cannot be deleted using the IWMQDEL macro. IWMQINS does not return a work unit token (WLMWUTKN).
- The SECUSER keyword is ignored.

Coding SERVER\_TOKEN=0 is equivalent to omitting the SERVER\_TOKEN keyword. The default is 0.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32-character field.

### **,USERDATA=*userdata***

A required input parameter, which contains data to pass to the server address space. This user data is returned to the caller of the IWMSSSEL or IWMSSEM macro. The format is undefined to MVS.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 16-character field.

### **,USERID=*userid***

When SECUSER=YES is specified, a required input parameter, which contains the requester's userid.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

### **,WLMWUTKN=*wlmwutkn***

An optional output parameter, which will receive the work unit token. This token can be passed to the IWMQDEL service to delete the work request.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 16-character field.

## ABEND Codes

None.

## Return and Reason Codes

When the IWMQINS macro returns control to your program:

- GPR 15 (and *retcode*, when you code RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, when you code RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 102. Return and Reason Codes for the IWMQINS Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> <code>IwmRetCodeOk</code> <b>Meaning:</b> Successful completion. <b>Action:</b> None required.
4	—	<b>Equate Symbol:</b> <code>IwmRetCodeWarning</code> <b>Meaning:</b> Successful completion, unusual conditions noted.
4	xxxx042E	<b>Equate Symbol:</b> <code>IwmRsnCodeServerNotFound</code> <b>Meaning:</b> The server token does not identify an existing server task. The server task may have terminated since the token was obtained. <b>Action:</b> If the server task has not terminated, check that the correct token is specified.
4	xxxx043A	<b>Equate Symbol:</b> <code>IwmRsnCodeRegionNotFound</code> <b>Meaning:</b> The region token does not identify a valid server region. <b>Action:</b> Please specify the correct region token.
8	—	<b>Equate Symbol:</b> <code>IwmRetCodeInvocError</code> <b>Meaning:</b> Invalid invocation environment or parameters.
8	xxxx0803	<b>Equate Symbol:</b> <code>IwmRsnCodeDisabled</code> <b>Meaning:</b> Caller is disabled. <b>Action:</b> Avoid requesting this function while disabled.
8	xxxx0804	<b>Equate Symbol:</b> <code>IwmRsnCodeLocked</code> <b>Meaning:</b> Caller is locked. <b>Action:</b> Avoid requesting this function while locked.
8	xxxx080B	<b>Equate Symbol:</b> <code>IwmRsnCodeBadPI</code> <b>Meaning:</b> Error accessing parameter list. <b>Action:</b> Check for possible storage overlay. Also check if you call this macro in 64-bit address mode. Refer to the description of reason code xxx089E for further information.
8	xxxx0821	<b>Equate Symbol:</b> <code>IwmRsnCodeBadConn</code> <b>Meaning:</b> Input connect token does not pass validity checking. <b>Action:</b> Make sure to use the connect token returned by the IWMCONN service requesting <code>Queue_Manager=Yes</code> , or <code>Server_Manager=Yes</code> with <code>Server_Type=Queue</code> .

Table 102. Return and Reason Codes for the IWMQINS Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0823	<b>Equate Symbol:</b> IwmRsnCodeDatoff <b>Meaning:</b> Caller invoked service while DATOFF <b>Action:</b> Avoid requesting this function in this environment.
8	xxxx0824	<b>Equate Symbol:</b> IwmRsnCodeAmode24 <b>Meaning:</b> Caller invoked service but was in 24 bit addressing mode. <b>Action:</b> Request this function only when you are in 31 bit addressing mode.
8	xxxx0825	<b>Equate Symbol:</b> IwmRsnCodeAscModeNotPrimary <b>Meaning:</b> Caller invoked service but was not in primary ASC mode. <b>Action:</b> Avoid requesting this function in this environment.
8	xxxx0827	<b>Equate Symbol:</b> IwmRsnCodeRsvdNot0 <b>Meaning:</b> Reserved field in parameter list was non-zero. <b>Action:</b> Check for use of keywords that are not supported by the MVS release on which the program is running.
8	xxxx0828	<b>Equate Symbol:</b> IwmRsnCodeBadVersion <b>Meaning:</b> Version number in parameter list is not valid. <b>Action:</b> Check for possible storage overlay of the parameter list.
8	xxxx083A	<b>Equate Symbol:</b> IwmRsnCodeBadEnclave <b>Meaning:</b> Enclave token in parameter list is not valid. <b>Action:</b> Check the specification of the ETOKEN parameter.
8	xxxx083F	<b>Equate Symbol:</b> IwmRsnCodePrimaryNotOwnConn <b>Meaning:</b> Primary address space does not own the passed connect token. <b>Action:</b> Ensure that the primary address space has previously connected to WLM using the IWMCONN macro. Ensure that the connect token returned by the IWMCONN macro is passed to the IWMQINS macro.
8	xxxx0840	<b>Equate Symbol:</b> IwmRsnCodeServiceNotEnabled <b>Meaning:</b> Caller's space connection is not enabled for this service <b>Action:</b> Make sure that Queue_Manager=Yes, or Server_Manager=Yes with Server_Type=Queue is specified on the IWMCONN request to enable this service.
8	xxxx0842	<b>Equate Symbol:</b> IwmRsnCodeNoWLMConnect <b>Meaning:</b> Caller's space disconnected from WLM during processing of the insert request. <b>Action:</b> None.

Table 102. Return and Reason Codes for the IWMQINS Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0845	<p><b>Equate Symbol:</b> IwmRsnCodeWrongEnclave</p> <p><b>Meaning:</b> The caller tried to queue a secondary work request to a specific server task using the SERVER_TOKEN parameter. The caller's enclave token does not match the enclave token of the last work request selected by the server task.</p> <p><b>Action:</b> Check that the correct enclave token was specified. Check that the server task is invoking the IWMSSEL and IWMSSEM macros in the correct sequence.</p>
8	xxxx089E	<p><b>Equate Symbol:</b> IwmRsnCodeServiceAModeMismatch</p> <p><b>Meaning:</b> The caller is in 64-bit address mode and tried to invoke a service macro that is only enabled for a 31-bit environment.</p> <p><b>Action:</b> Use the 64-bit enabled service macro (IWM4QIN) or change the address mode of the caller to 31-bit.</p>
C	—	<p><b>Equate Symbol:</b> IwmRetCodeEnvError</p> <p><b>Meaning:</b> Environmental error.</p>
C	xxxx0C01	<p><b>Equate Symbol:</b> IwmRsnCodeNoStg</p> <p><b>Meaning:</b> Storage is not available for the request.</p> <p><b>Action:</b> Contact your system programmer. There is a common storage shortage.</p>
C	xxxx0C16	<p><b>Equate Symbol:</b> IwmRsnCodeServerUnavail</p> <p><b>Meaning:</b> A server cannot be started to process the work request. This is probably caused by one of the following:</p> <ol style="list-style-type: none"> <li>1. An error in the JCL procedure used to start the server address space.</li> <li>2. Repeated, unexpected terminations of the server address space.</li> </ol> <p>In either of these cases, workload management stops the application environment. A DISPLAY WLM command shows this state as INTERNALLY STOPPED.</p> <p><b>Action:</b> Look at the system log to determine what caused the error:</p> <ol style="list-style-type: none"> <li>1. If it is a JCL error, correct the error in the procedure.</li> <li>2. If it is repeated terminations of the server address space, correct the application error causing the termination.</li> </ol> <p>In either case, the server environment can then be resumed using the VARY operator command: V WLM,APLLENV=nnn,RESUME where nnn is the applicable application environment name.</p> <p><b>Note:</b> A re-IPL of some or all of the systems in the sysplex does not reset the stopped state of the application environment. The VARY command is the only way to resume the environment.</p>

Table 102. Return and Reason Codes for the IWMQINS Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
C	xxxx0C1A	<p><b>Equate Symbol:</b> IwmRsnCodeApplNotDefined</p> <p><b>Meaning:</b> The application environment name is not defined in the active WLM policy.</p> <p><b>Action:</b> Check whether the correct application environment name is being used. If so, a service administrator must define the application environment in the WLM service definition.</p>
C	xxxx0C1B	<p><b>Equate Symbol:</b> IwmRsnCodeApplNotSST</p> <p><b>Meaning:</b> The application environment name is defined for use by a different subsystem type in the active WLM policy.</p> <p><b>Action:</b> Check whether the correct application environment name is being used. If so, a service administrator must change the application environment in the WLM service definition to specify the correct subsystem type.</p>
C	xxxx0C1D	<p><b>Equate Symbol:</b> IwmRsnCodeQMGrNotActive</p> <p><b>Meaning:</b> The required Queue Manager is not active.</p> <p><b>Action:</b> The Queue Manager with the same subsystem type and name as the server must be started and connected to workload management before the request can be honored.</p>
C	xxxx0C22	<p><b>Equate Symbol:</b> IwmRsnCodeApplEnvQuiesced</p> <p><b>Meaning:</b> For server applications connecting to WLM with subsystem type IWEB only: The application environment has been quiesced. The work request is not inserted to the WLM work queue.</p> <p><b>Action:</b> Resume the application environment.</p>
C	xxxx0C40	<p><b>Equate Symbol:</b> IwmRsnCodeNoSafCheckPossible</p> <p><b>Meaning:</b> MLS is active but a security check could not be performed probably for one of the following reasons:</p> <ul style="list-style-type: none"> <li>•</li> <li>• No security decision could be made. The RACF router was not loaded; the request, resource, subsystem combination could not be found in the RACF ROUTER table,...</li> <li>• A resource or class name is not defined to RACF or the class has not been RACListed.</li> <li>• The class was RACListed, but the data space cannot be accessed due to an ALESERV failure.</li> <li>• The class was RACListed, but the data space has been deleted.</li> <li>• No security decision could be made. The RACF router was not loaded; the request, resource, subsystem combination could not be found in the RACF ROUTER table.</li> </ul> <p><b>Action:</b> Contact your RACF Security Administrator. Check if RACF is properly installed, configured and tuned. Correct the eventual problems.</p>



Table 102. Return and Reason Codes for the IWMQINS Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
C	xxxx0C41	<p><b>Equate Symbol:</b> IwmRsnCodeSafCheckFailed</p> <p><b>Meaning:</b> MLS is active. Queue Manager and Server Manager are not authorized to communicate.</p> <p><b>Action:</b> Normally none. If QM and SM really must communicate, conta your RACF Security Administrator. Set the appropriate Security Labels.</p>
C	xxxx0C42	<p><b>Equate Symbol:</b> IwmRsnCodeAletError</p> <p><b>Meaning:</b> Error while accessing access list with ALESERV probably because of one of the following</p> <ol style="list-style-type: none"> <li>1.</li> <li>2. The current access list cannot be expanded. There are no free access list entries and the maximum size has been reached.</li> <li>3. ALESERV could not obtain storage for an expanded access list.</li> </ol> <p><b>Action:</b> Delete unused entries and reissue the request in first case. Free some storage and retry the request in second case. Contact your System Programmer if none works</p>
10	—	<p><b>Equate Symbol:</b> IwmRetCodeCompError</p> <p><b>Meaning:</b> Component error.</p> <p><b>Action:</b> Contact your system programmer.</p>

## Example

To insert a work request onto the WLM queue manager queues:

```

IWMQINS  CONNTKN=CONNTOKEN,ETOKEN=ENCTOKEN,          X
          USERDATA=USERDATA,APPLENV=APPLENV,SECUSER=NO, X
          WLMWUTKN=WLMWUTKN,RETCODE=RC,RSNCODE=RSN
*
* Storage areas
*
CONNTOKEN DS   FL4           Contains the connect token
*                               associated with the use of WLM
*                               Queuing services as returned by
*                               IWMCONN
*                               (specifying QUEUE_MANAGER=YES
*                               or SERVER_MANAGER=YES
*                               SERVER_TYPE=QUEUE
ENCTOKEN DS   CL8           Contains the Enclave token
*                               associated with the work
*                               request as returned by IWMECREA
USERDATA DS   CL16          Contains data maintained by the
*                               user
APPLENV  DS   CL32          Contains the application
*                               environment name
WLMWUTKN DS   CL16          Work unit token
RC       DS    F            Return code
RSN     DS    F            Reason code

```

## IWMSLIM – Application Environment Limit Service

The IWMSLIM service should be used to tell WLM the total number of server instances which are supported by the application. WLM will ensure that no more server instances will be started in the system.

In addition the caller can define a minimum number of servers which should be made available by WLM regardless of whether work is available to execute or not. If the user defines multiple service classes to give the work of the application different service goals, the caller can define that the minimum number of servers is spread across these service classes to ensure that servers are available for all work executed by the application.

The caller must have previously connected to WLM using the IWMCONN service specifying SERVER\_MANAGER=YES and SERVER\_TYPE=QUEUE. It is recommended to use the IWMSLIM service directly after IWMCONN. If any server uses this service to define limits, the limits apply for all servers of the application environment regardless of whether other servers use the service or not.

If a server defines new limits during execution, WLM attempts to meet the new limit definitions as soon as possible. If the maximum limit for servers is reduced during execution it is not predictable when WLM is able to meet the new maximum definition. This depends highly on the execution time of the running work requests. Therefore changing the limits during execution should be used very carefully and primarily during times of low application utilization.

### Note

It is recommended to use the new equivalent service IWM4SLI introduced with z/OS V1R6. IWM4SLI also supports 64-bit address space. For further information refer to Chapter 77, "IWM4SLI – Application Environment Limit Service," on page 653.

## Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Problem state. Any PSW key
<b>Dispatchable unit mode:</b>	Task
<b>Cross memory mode:</b>	PASN=HASN=SASN
<b>AMODE:</b>	31-bit
<b>ASC mode:</b>	Primary
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks may be held.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

## Programming Requirements

1. Make sure no EUT FRRs are established.
2. The macro CVT must be included to use this macro.
3. The macro IWMYCON must be included to use this macro.
4. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.

- Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, IWMRSNCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.

## Restrictions

- This macro may not be used during task/address space termination.
- Only a single invocation is allowed to be active for a given address space at any given time.
- Before using this macro the caller must connect to WLM via IWMCONN Server\_Manager=YES, Server\_Type=Queue.
- The macro must be used directly after using IWMCONN.

## Input Register Information

Before issuing the IWMSLIM macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

## Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Reason code if GR15 return code is non-zero
1	Used as work registers by the system
2-13	Unchanged
14	Used as work registers by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

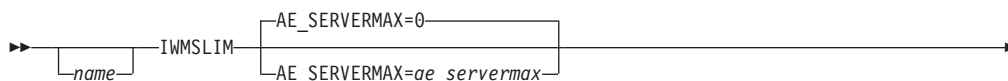
Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

## Performance Implications

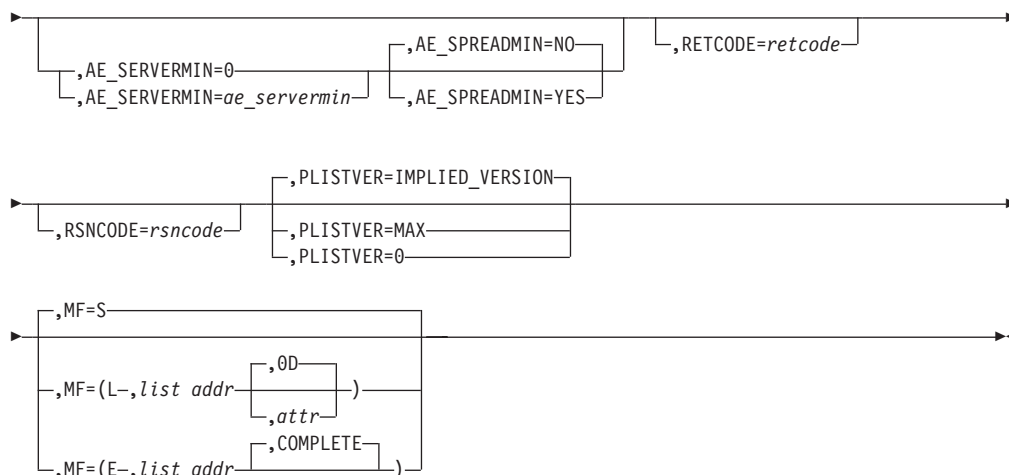
None.

## Syntax

main diagram



## IWMSLIM Macro



## Parameters

The parameters are explained as follows:

### *name*

An optional symbol, starting in column 1, that is the name on the IWMSLIM macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

### **,AE\_SERVERMAX=ae\_servermax**

#### **,AE\_SERVERMAX=0**

An optional input parameter, which indicates the architectural limit for the total number of server instances which can run concurrently across the application environment for a given subsystem type and subsystem name.

This parameter represents a physical limit, such as the maximum number of available connections to a back-end subsystem. WLM will not start more than this number of server instances, even if goals cannot be met because of the limit. This value should be an integral multiple of the PARALLEL\_EU value defined on the IWMCONN service. If AE\_SERVERMAX is not an even multiple of PARALLEL\_EU, WLM will round this value down to the next integral multiple.

The maximum limit and the number of service classes to execute work requests should be defined carefully. If the number of service classes exceeds the quotient of AE\_SERVERMAX divided by PARALLEL\_EU WLM cannot start enough server address spaces to execute the work requests for all service classes. The default is 0, indicating that no maximum limit has been specified

**To code:** Specify the RS-type address of a halfword field.

### **,AE\_SERVERMIN=ae\_servermin**

#### **,AE\_SERVERMIN=0**

An optional input parameter, which indicates the minimum number of servers which should be up and running at all times.

This parameter can be used to tell WLM that a certain amount of server tasks should always be kept available to select work. This value should be an integral multiple of the PARALLEL\_EU value defined on IWMCONN service. If AE\_SERVERMIN is not an even multiple of PARALLEL\_EU, WLM will round this value down to the next integral multiple. The default is 0, indicating that no limit has been specified

**To code:** Specify the RS-type address of a halfword field.

**,AE\_SPREADADMIN=NO****,AE\_SPREADADMIN=YES**

When `AE_SERVERMIN=ae_servermin` is specified, an optional parameter, which indicates whether WLM will distribute the minimum number of servers as evenly as possible across the service classes being used to process the work requests. The default is `AE_SPREADADMIN=NO`.

**,AE\_SPREADADMIN=NO**

The server tasks specified in `AE_SERVERMIN` will be distributed to service classes as needed in order to meet goals.

**,AE\_SPREADADMIN=YES**

The server tasks specified in `AE_SERVERMIN` will be distributed as evenly as possible to all service classes being used to execute work requests.

**,MF=S****,MF=(L,list addr)****,MF=(L,list addr,attr)****,MF=(L,list addr,0D)****,MF=(E,list addr)****,MF=(E,list addr,COMPLETE)**

An optional input parameter that specifies the macro form.

Use `MF=S` to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. `MF=S` is the default.

Use `MF=L` to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the `PLISTVER` parameter may be coded with the list form of the macro.

Use `MF=E` to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

**,list addr**

The name of a storage area to contain the parameters. For `MF=S` and `MF=E`, this can be an RS-type address or an address in register (1)-(12).

**,attr**

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of `0F` to force the parameter list to a word boundary, or `0D` to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of `0D`.

**,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

**,PLISTVER=IMPLIED\_VERSION****,PLISTVER=MAX****,PLISTVER=0**

An optional input parameter that specifies the version of the macro. `PLISTVER` determines which parameter list the system generates. `PLISTVER` is an optional input parameter on all forms of the macro, including the list form. When using `PLISTVER`, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

## IWMSLIM Macro

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form, when both are assembled with the same level of the system. In this way, MAX ensures that the parameter list does not overwrite nearby storage.

- **0**, if you use the currently available parameters.

**To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX
- A decimal value of 0

**,RETCODE=retcode**

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,RSNCODE=rsncode**

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

## ABEND Codes

None.

## Return and Reason Codes

When the IWMSLIM macro returns control to your program:

- GPR 15 (and *retcode*, when you code RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, when you code RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 103. Return and Reason Codes for the IWMSLIM Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> IwmRetCodeOk <b>Meaning:</b> Successful completion. <b>Action:</b> None required.
8	—	<b>Equate Symbol:</b> IwmRetCodeInvocError <b>Meaning:</b> Invalid invocation environment or parameters.

Table 103. Return and Reason Codes for the IWMSLIM Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0801	<b>Equate Symbol:</b> lwmRsnCodeSrbMode <b>Meaning:</b> Caller is in SRB mode. <b>Action:</b> Avoid requesting this function while in SRB mode.
8	xxxx0803	<b>Equate Symbol:</b> lwmRsnCodeDisabled <b>Meaning:</b> Caller is disabled. <b>Action:</b> Avoid requesting this function while disabled.
8	xxxx0804	<b>Equate Symbol:</b> lwmRsnCodeLocked <b>Meaning:</b> Caller is locked. <b>Action:</b> Avoid requesting this function while locked.
8	xxxx080B	<b>Equate Symbol:</b> lwmRsnCodeBadPI <b>Meaning:</b> Error accessing parameter list. <b>Action:</b> Check for possible storage overlay. Also check if you call this macro in 64-bit address mode. Refer to the description of reason code xxxx089E for further information.
8	xxxx0824	<b>Equate Symbol:</b> lwmRsnCodeAmode24 <b>Meaning:</b> Caller invoked service but was in 24 bit addressing mode. <b>Action:</b> Request this function only when you are in 31 bit addressing mode.
8	xxxx0825	<b>Equate Symbol:</b> lwmRsnCodeAscModeNotPrimary <b>Meaning:</b> Caller invoked service but was not in primary ASC mode. <b>Action:</b> Avoid requesting this function in this environment.
8	xxxx0828	<b>Equate Symbol:</b> lwmRsnCodeBadVersion <b>Meaning:</b> Version number in parameter list is not valid. <b>Action:</b> Check for possible storage overlay of the parameter list.
8	xxxx0840	<b>Equate Symbol:</b> lwmRsnCodeServiceNotEnabled <b>Meaning:</b> Caller's space connection is not enabled for this service <b>Action:</b> Make sure that SERVER_MANAGER=YES and SERVER_TYPE=QUEUE is specified on the IWMCNN request to enable this service.
8	xxxx0841	<b>Equate Symbol:</b> lwmRsnCodeXmemMode <b>Meaning:</b> Caller is in cross-memory mode. <b>Action:</b> Request this function only when you are not in cross-memory mode.
8	xxxx0842	<b>Equate Symbol:</b> lwmRsnCodeNoWLMConnect <b>Meaning:</b> Caller's space is not connected to WLM. <b>Action:</b> Invoke the IWMCNN macro before invoking this macro.

## IWMSLIM Macro

Table 103. Return and Reason Codes for the IWMSLIM Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx089E	<p><b>Equate Symbol:</b> IwmRsnCodeServiceAModeMismatch</p> <p><b>Meaning:</b> The caller is in 64-bit address mode and tried to invoke a service macro that is only enabled for a 31-bit environment.</p> <p><b>Action:</b> Use the 64-bit enabled service macro (IWM4SLI) or change the address mode of the caller to 31-bit.</p>
10	—	<p><b>Equate Symbol:</b> IwmRetCodeCompError</p> <p><b>Meaning:</b> Component error.</p> <p><b>Action:</b> Contact your system programmer.</p>

## Example

To specify application limits to WLM.

```

IWMCNN WORK_MANAGER=YES,
        SERVER_MANAGER=YES,
        PARALLEL_EU=EUNITS,
        SERVER_TYPE=QUEUE,
        CONNTKN=CTKN,
        CONNTKNKEY=PSWKEY,
        RETCODE=RC,
        RSNCODE=RSN

IWMSLIM AE_SERVERMAX=MAXSRVS,
        AE_SERVERMIN=MINSRVS,
        RETCODE=RC,
        RSNCODE=RSN
*
* Storage areas
*
EUNITS  DS    F           Number of Tasks which will be started
*                               per address space.
MAXSRVS DS    H           Maximum Number of Servers supported
*                               by the application.
MINSRVS DS    H           Minimum number of servers which should
*                               be up and running all time
CTKN    DS    FL4         Connect Token
RC      DS    F           Return code
RSN     DS    F           Reason code

```



## IWMSSSEL – Selecting a Request from a Caller’s Work Manager Queue

The IWMSSSEL service selects the next work request from the queue associated with the caller’s application environment. The caller must have previously connected to WLM using the IWMCONN service specifying `SERVER_MANAGER=YES`.

If there are no queued work requests waiting for selection the calling task will be suspended, pending arrival of work to do. The caller cannot rely upon asynchronous exits receiving control while the task is suspended.

After a work request is selected, the caller uses the IWMSTBGN and IWMSTEND services to indicate the start and end of processing of the request.

### Note

It is recommended to use the new equivalent service IWM4SSL introduced with z/OS V1R6. IWM4SSL also supports 64-bit address space. For further information refer to Chapter 79, “IWM4SSL – Selecting a Request from a Caller’s Work Manager Queue,” on page 669.

## Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Problem state. Any PSW key
<b>Dispatchable unit mode:</b>	Task
<b>Cross memory mode:</b>	PASN=HASN=SASN
<b>AMODE:</b>	31-bit
<b>ASC mode:</b>	Primary
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks may be held.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

## Programming Requirements

1. The macro CVT must be included to use this macro.
2. The macro IWMYCON must be included to use this macro.
3. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
4. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, IWMRSNCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.

## Restrictions

None.

## IWMSSEL Macro

### Input Register Information

Before issuing the IWMSSEL macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

### Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Reason code if GR15 return code is non-zero
1	Used as work registers by the system
2-13	Unchanged
14	Used as work registers by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

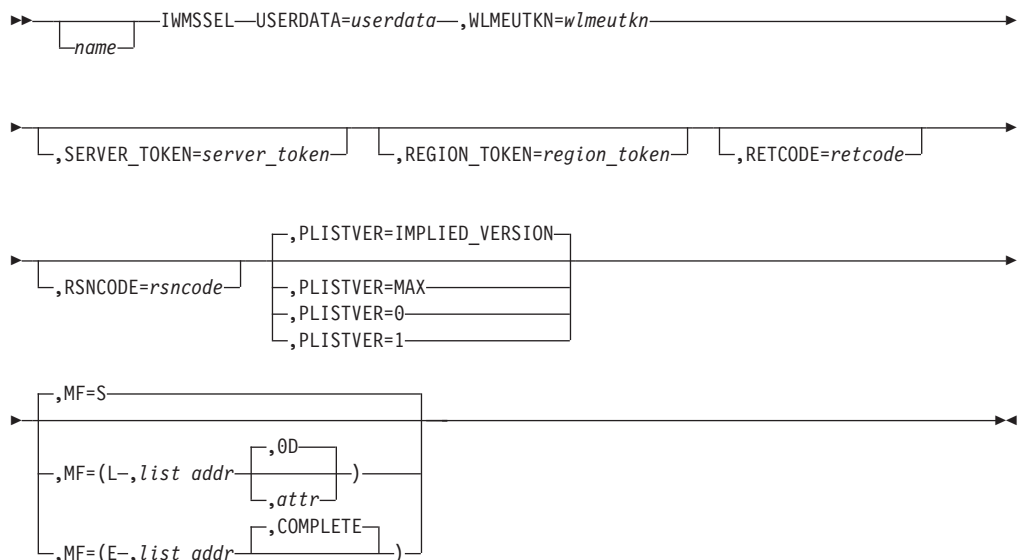
Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

### Performance Implications

None.

### Syntax

#### main diagram



## Parameters

The parameters are explained as follows:

### *name*

An optional symbol, starting in column 1, that is the name on the IWMSSSEL macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

### **,MF=S**

**,MF=(L,*list addr*)**

**,MF=(L,*list addr,attr*)**

**,MF=(L,*list addr,0D*)**

**,MF=(E,*list addr*)**

**,MF=(E,*list addr,COMPLETE*)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

### *,list addr*

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

### *,attr*

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

### **,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

### **,PLISTVER=IMPLIED\_VERSION**

**,PLISTVER=MAX**

**,PLISTVER=0**

**,PLISTVER=1**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.

## IWMSSEL Macro

- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify `PLISTVER=MAX` on the list form of the macro. Specifying `MAX` ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form, when both are assembled with the same level of the system. In this way, `MAX` ensures that the parameter list does not overwrite nearby storage.

- **0**, if you use the currently available parameters.
- **1**, which supports both the following parameters and those from version 0:

`SERVER_TOKEN`

**To code:** Specify one of the following:

- `IMPLIED_VERSION`
- `MAX`
- A decimal value of 0 or 1

**,REGION\_TOKEN=***region\_token*

An optional output parameter, which contains a region token. A queueing manager can use the region token to queue work requests to a specific server region. These work requests are considered to belong to a set of work requests all needing access to same status information which exists only in the virtual storage of the server region. They are selected using the `IWMSSEL` macro. It is assumed that the application uses the service `IWMTAFF` to tell WLM when the temporary affinity to the defined server region begins and ends.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,RETCODE=***retcode*

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,RSNCODE=***rsncode*

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a

**,SERVER\_TOKEN=***server\_token*

An optional output parameter, which contains a server token. A queueing manager can use the server token to queue secondary work requests to this server task. Secondary work requests are considered to be extensions of the work request selected by `IWMSSEL`. They are selected using the `IWMSSEM` macro. See the `IWMSSEM` macro for more information.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 32-character field.

**USERDATA=***userdata*

A required output parameter, which contains the user data previously passed to WLM via `IWMQINS`.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 16-character field.

,WLMEUTKN=*wlmeutkn*

A required output parameter, which will receive the execution unit token. This token must be passed on subsequent IWMSTBGN and IWMSTEND requests.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

## ABEND Codes

None.

## Return and Reason Codes

When the IWMSEL macro returns control to your program:

- GPR 15 (and *retcode*, when you code RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, when you code RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 104. Return and Reason Codes for the IWMSEL Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> <code>IwmRetCodeOk</code> <b>Meaning:</b> Successful completion. <b>Action:</b> None required.
8	—	<b>Equate Symbol:</b> <code>IwmRetCodeInvocError</code> <b>Meaning:</b> Invalid invocation environment or parameters.
8	xxxx0801	<b>Equate Symbol:</b> <code>IwmRsnCodeSrbMode</code> <b>Meaning:</b> Caller is in SRB mode. <b>Action:</b> Avoid requesting this function while in SRB mode.
8	xxxx0803	<b>Equate Symbol:</b> <code>IwmRsnCodeDisabled</code> <b>Meaning:</b> Caller is disabled. <b>Action:</b> Avoid requesting this function while disabled.
8	xxxx0804	<b>Equate Symbol:</b> <code>IwmRsnCodeLocked</code> <b>Meaning:</b> Caller is locked. <b>Action:</b> Avoid requesting this function while locked.
8	xxxx080B	<b>Equate Symbol:</b> <code>IwmRsnCodeBadPI</code> <b>Meaning:</b> Error accessing parameter list. <b>Action:</b> Check for possible storage overlay. Also check if you call this macro in 64-bit address mode. Refer to the description of reason code xxxx089E for further information.
8	xxxx0823	<b>Equate Symbol:</b> <code>IwmRsnCodeDatoff</code> <b>Meaning:</b> Caller invoked service while DATOFF <b>Action:</b> Avoid requesting this function in this environment.

Table 104. Return and Reason Codes for the IWMSEL Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0824	<p><b>Equate Symbol:</b> IwmRsnCodeAmode24</p> <p><b>Meaning:</b> Caller invoked service but was in 24 bit addressing mode.</p> <p><b>Action:</b> Request this function only when you are in 31 bit addressing mode.</p>
8	xxxx0825	<p><b>Equate Symbol:</b> IwmRsnCodeAscModeNotPrimary</p> <p><b>Meaning:</b> Caller invoked service but was not in primary ASC mode.</p> <p><b>Action:</b> Avoid requesting this function in this environment.</p>
8	xxxx0827	<p><b>Equate Symbol:</b> IwmRsnCodeRsvdNot0</p> <p><b>Meaning:</b> Reserved field in parameter list was non-zero.</p> <p><b>Action:</b> Check for possible storage overlay of the parameter list.</p>
8	xxxx0828	<p><b>Equate Symbol:</b> IwmRsnCodeBadVersion</p> <p><b>Meaning:</b> Version number in parameter list is not valid.</p> <p><b>Action:</b> Check for possible storage overlay of the parameter list.</p>
8	xxxx0840	<p><b>Equate Symbol:</b> IwmRsnCodeServiceNotEnabled</p> <p><b>Meaning:</b> Caller's space connection is not enabled for this service</p> <p><b>Action:</b> Make sure that SERVER_MANAGER=YES is specified on the IWMCONN request to enable this service.</p>
8	xxxx0841	<p><b>Equate Symbol:</b> IwmRsnCodeXmemMode</p> <p><b>Meaning:</b> Caller is in cross-memory mode.</p> <p><b>Action:</b> Request this function only when you are not in cross-memory mode.</p>
8	xxxx0842	<p><b>Equate Symbol:</b> IwmRsnCodeNoWLMConnect</p> <p><b>Meaning:</b> Caller's space is not connected to WLM.</p> <p><b>Action:</b> Invoke the IWMCONN macro before invoking this macro.</p>
8	xxxx0854	<p><b>Equate Symbol:</b> IwmRsnCodeTooManySelect</p> <p><b>Meaning:</b> The caller is attempting to select more work units than it has tasks to execute the work.</p> <p><b>Action:</b> Wait until an execution task has issued IWMSTEND before attempting to select more work units.</p>
8	xxxx0864	<p><b>Equate Symbol:</b> IwmRsnCodeSecondaryWorkExists</p> <p><b>Meaning:</b> There are secondary work requests queued to this server task. The caller was expected to process them using IWMSEM before calling IWMSEL.</p> <p><b>Action:</b> Select all secondary work requests before issuing IWMSEL.</p>

Table 104. Return and Reason Codes for the IWMSEL Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx089E	<b>Equate Symbol:</b> IwmRsnCodeServiceAModeMismatch  <b>Meaning:</b> The caller is in 64-bit address mode and tried to invoke a service macro that is only enabled for a 31-bit environment.  <b>Action:</b> Use the 64-bit enabled service macro (IWM4SSL) or change the address mode of the caller to 31-bit.
C	—	<b>Equate Symbol:</b> IwmRetCodeEnvError  <b>Meaning:</b> Environmental error.
C	xxxx0C01	<b>Equate Symbol:</b> IwmRsnCodeNoStg  <b>Meaning:</b> Storage is not available for the request.  <b>Action:</b> Caller must disconnect by invoking the IWMDISC macro.
C	xxxx0C14	<b>Equate Symbol:</b> IwmRsnCodeNoWorkShutDown  <b>Meaning:</b> No work selected. Caller is to shutdown.  <b>Action:</b> Caller must disconnect by invoking the IWMDISC macro.
C	xxxx0C3B	<b>Equate Symbol:</b> IwmRsnCodeStopTask  <b>Meaning:</b> WLM decided to stop the server instance.  <b>Action:</b> Calling task must shutdown, but server address space must remain active.
10	—	<b>Equate Symbol:</b> IwmRetCodeCompError  <b>Meaning:</b> Component error.  <b>Action:</b> Contact your system programmer.

## Example

To select a work request from the WLM queue manager queues:

```

IWMSEL USERDATA=USERDATA,          X
      WLMEUTKN=WLMEUTKN, RETCODE=RC, RSNCODE=RSN
*
* Storage areas
*
USERDATA DS    CL16          Contains the user-defined data
*                that was passed to IWMQINS
WLMEUTKN DS    CL8           Work unit token that must be
*                passed to IWMSTBGN and IWMSTEND
RC        DS    F           Return code
RSN       DS    F           Reason code

```

---

## IWMSEM – WLM Server Select Secondary Service

The IWMSEM service selects the next secondary work request from the queue associated with the caller's server task.

If there are no queued secondary work requests waiting for selection the calling task will be suspended, pending arrival of work to do. The caller cannot rely upon asynchronous exits receiving control while the task is suspended.

## IWMSSEM Macro

Secondary work requests are considered to be extensions of an original work request selected using IWMSSEL. The caller must invoke WLM services in the following sequence:

- The caller invokes the IWMSSEL macro to select an initial work request. IWMSSEL returns a token identifying the server task. The caller is responsible for passing the server token to the queueing manager so that it can insert secondary work requests.
- The caller invokes the IWMSTBGN macro to establish an environment for processing the work request selected using IWMSSEL. This environment also covers all secondary work requests.
- The caller invokes the IWMSSEM macro to select each secondary work request. The queueing manager is responsible for indicating the last secondary work request so that the server task knows when not to try to select another one.
- After the last secondary work request has been processed, the caller invokes the IWMSTEND macro to remove the environment created by IWMSTBGN.
- The caller invokes IWMSSEL to select a new primary work request, and repeats the above flow.

In the above flow, IWMSSEL, IWMSTBGN, IWMSSEM, and IWMSTEND must be invoked from the same task.

### Note

It is recommended to use the new equivalent service IWM4SSM introduced with z/OS V1R6. IWM4SSM also supports 64-bit address space. For further information refer to Chapter 80, "IWM4SSM – WLM Server Select Secondary Service," on page 677.

## Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Problem state. Any PSW key
<b>Dispatchable unit mode:</b>	Task
<b>Cross memory mode:</b>	PASN=HASN=SASN
<b>AMODE:</b>	31-bit
<b>ASC mode:</b>	Primary
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks may be held.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

## Programming Requirements

1. The macro CVT must be included to use this macro.
2. The macro IWMYCON must be included to use this macro.
3. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
4. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded



from comparison with the reason code values described above. The constant, IWMRSNCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.

## Restrictions

None.

## Input Register Information

Before issuing the IWMSSEM macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

## Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Reason code if GR15 return code is non-zero
1	Used as work registers by the system
2-13	Unchanged
14	Used as work registers by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

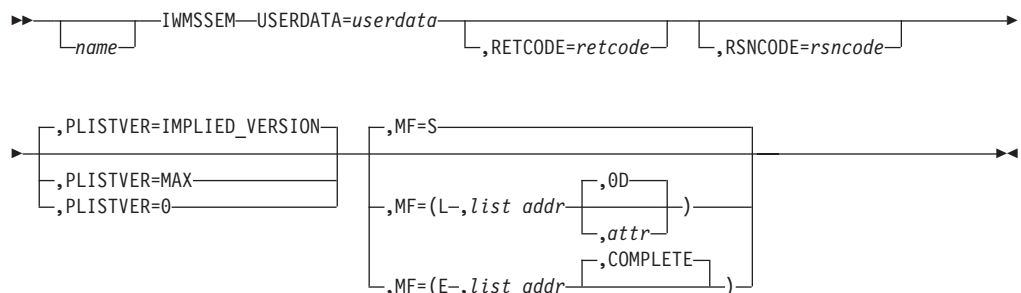
Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

## Performance Implications

None.

## Syntax

### main diagram



## Parameters

The parameters are explained as follows:

## IWMSSEM Macro

### *name*

An optional symbol, starting in column 1, that is the name on the IWMSSEM macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

### **,MF=S**

### **,MF=(L,*list addr*)**

### **,MF=(L,*list addr*,*attr*)**

### **,MF=(L,*list addr*,0D)**

### **,MF=(E,*list addr*)**

### **,MF=(E,*list addr*,COMPLETE)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

### *,list addr*

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

### *,attr*

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

### **,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

### **,PLISTVER=IMPLIED\_VERSION**

### **,PLISTVER=MAX**

### **,PLISTVER=0**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that

the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form, when both are assembled with the same level of the system. In this way, MAX ensures that the parameter list does not overwrite nearby storage.

- 0, if you use the currently available parameters.

**To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX
- A decimal value of 0

**,RETCODE=retcode**

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,RSNCODE=rsncode**

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**USERDATA=userdata**

A required output parameter, which contains the user data previously passed to WLM via IWMQINS.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 16-character field.

## ABEND Codes

None.

## Return and Reason Codes

When the IWMSSSEM macro returns control to your program:

- GPR 15 (and *retcode*, when you code RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, when you code RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 105. Return and Reason Codes for the IWMSSSEM Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> <code>IwmRetCodeOk</code> <b>Meaning:</b> Successful completion. <b>Action:</b> None required.
8	—	<b>Equate Symbol:</b> <code>IwmRetCodeInvocError</code> <b>Meaning:</b> Invalid invocation environment or parameters.
8	xxxx0801	<b>Equate Symbol:</b> <code>IwmRsnCodeSrbMode</code> <b>Meaning:</b> Caller is in SRB mode. <b>Action:</b> Avoid requesting this function while in SRB mode.

Table 105. Return and Reason Codes for the IWMSSEM Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0803	<b>Equate Symbol:</b> IwmRsnCodeDisabled <b>Meaning:</b> Caller is disabled. <b>Action:</b> Avoid requesting this function while disabled.
8	xxxx0804	<b>Equate Symbol:</b> IwmRsnCodeLocked <b>Meaning:</b> Caller is locked. <b>Action:</b> Avoid requesting this function while locked.
8	xxxx080B	<b>Equate Symbol:</b> IwmRsnCodeBadPI <b>Meaning:</b> Error accessing parameter list. <b>Action:</b> Check for possible storage overlay. Also check if you call this macro in 64-bit address mode. Refer to the description of reason code xxxx089E for further information.
8	xxxx0824	<b>Equate Symbol:</b> IwmRsnCodeAmode24 <b>Meaning:</b> Caller invoked service but was in 24 bit addressing mode. <b>Action:</b> Request this function only when you are in 31 bit addressing mode.
8	xxxx0825	<b>Equate Symbol:</b> IwmRsnCodeAscModeNotPrimary <b>Meaning:</b> Caller invoked service but was not in primary ASC mode. <b>Action:</b> Avoid requesting this function in this environment.
8	xxxx0828	<b>Equate Symbol:</b> IwmRsnCodeBadVersion <b>Meaning:</b> Version number in parameter list is not valid. <b>Action:</b> Check for possible storage overlay of the parameter list.
8	xxxx0840	<b>Equate Symbol:</b> IwmRsnCodeServiceNotEnabled <b>Meaning:</b> Caller's space connection is not enabled for this service <b>Action:</b> Make sure that SERVER_MANAGER=YES is specified on the IWMCONN request to enable this service.
8	xxxx0841	<b>Equate Symbol:</b> IwmRsnCodeXmemMode <b>Meaning:</b> Caller is in cross-memory mode. <b>Action:</b> Request this function only when you are not in cross-memory mode.
8	xxxx0842	<b>Equate Symbol:</b> IwmRsnCodeNoWLMConnect <b>Meaning:</b> Caller's space is not connected to WLM. <b>Action:</b> Invoke the IWMCONN macro before invoking this macro.
8	xxxx0862	<b>Equate Symbol:</b> IwmRsnCodeNoPriorSelect <b>Meaning:</b> The caller has not previously selected work using the IWMSSEL macro. <b>Action:</b> Invoke the IWMSSEL macro before invoking this macro.

Table 105. Return and Reason Codes for the IWMSSSEM Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0863	<p><b>Equate Symbol:</b> IwmRsnCodeNoExecEnv</p> <p><b>Meaning:</b> The caller has not established an execution environment using IWMSTBGN.</p> <p><b>Action:</b> Invoke the IWMSTBGN macro before invoking this macro.</p>
8	xxxx089E	<p><b>Equate Symbol:</b> IwmRsnCodeServiceAModeMismatch</p> <p><b>Meaning:</b> The caller is in 64-bit address mode and tried to invoke a service macro that is only enabled for a 31-bit environment.</p> <p><b>Action:</b> Use the 64-bit enabled service macro (IWM4SSM) or change the address mode of the caller to 31-bit.</p>
C	—	<p><b>Equate Symbol:</b> IwmRetCodeEnvError</p> <p><b>Meaning:</b> Environmental error.</p>
C	xxxx0C14	<p><b>Equate Symbol:</b> IwmRsnCodeNoWorkShutDown</p> <p><b>Meaning:</b> No work selected. Caller is to shutdown.</p> <p><b>Action:</b> Caller must disconnect by invoking the IW DISC macro.</p>
10	—	<p><b>Equate Symbol:</b> IwmRetCodeCompError</p> <p><b>Meaning:</b> Component error.</p> <p><b>Action:</b> Contact your system programmer.</p>

## Example

To select a secondary work request from the WLM queue manager queues:

```

IWMSEM USERDATA=USERDATA,RETcode=RC,RSNcode=RSN
*
* Storage areas
*
USERDATA DS    CL16           Contains the user-defined data
*                               that was passed to IWMQINS
RC        DS    F             Return code
RSN       DS    F             Reason code

```

---

## IWMSTBGN – Beginning a Request from a Caller’s Work Manager Queue

IWMSTBGN establishes the environment to process a work request that was previously selected using IWMSEL. The caller must invoke IWMSTBGN from the task in the server address space that will process the request. IWMSTBGN establishes a business unit-of-work relationship by joining the caller’s task to the enclave associated with the request. IWMSTBGN creates a security environment if there is a user ID associated with the request previously selected.

Use IWMSTBGN together with IWMSTEND to begin and end the processing of a work request. A task can process only one work request at a time.

Note that a task may only join an Enclave if it is not already part of an Enclave. In particular, a subtask which inherited the Enclave attribute from its mother task (which may happen either as a result of the mother task issuing IWMEJOIN or IWMSTBGN) is not allowed to use IWMEJOIN to explicitly join an Enclave. This restriction is independent of whether the specified Enclave is the same Enclave as it is in, or a different Enclave from the one it is in. Such a subtask which inherited the Enclave attribute is also not allowed to use IWMELEAV to explicitly leave the Enclave. The subtask would only leave the Enclave upon its own (task) termination or when the Enclave is deleted (IWMEDELE). Also, a task which successfully establishes a Begin environment (IWMSTBGN) may not invoke Enclave Join, nor is the task allowed to use Enclave Leave while this Begin environment exists.

### Note

It is recommended to use the new equivalent service IWM4STBG introduced with z/OS V1R6. IWM4STBG also supports 64-bit address space. For further information refer to Chapter 81, “IWM4STBG – Beginning a Request from a Caller’s Work Manager Queue,” on page 683.

## Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Problem state. Any PSW key
<b>Dispatchable unit mode:</b>	Task
<b>Cross memory mode:</b>	PASN=HASN=SASN
<b>AMODE:</b>	31-bit
<b>ASC mode:</b>	Primary
<b>Interrupt status:</b>	Enabled for I/O and external interrupts

<b>Locks:</b>	No locks may be held.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

## Programming Requirements

1. The macro CVT must be included to use this macro.
2. The macro IWMYCON must be included to use this macro.
3. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
4. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, IWMSRNCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.

## Restrictions

1. The caller cannot have an EUT FRR established.

## Input Register Information

Before issuing the IWMSTBGN macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

## Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Reason code if GR15 return code is non-zero
1	Used as work register by the system
2-13	Unchanged
14	Used as work registers by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

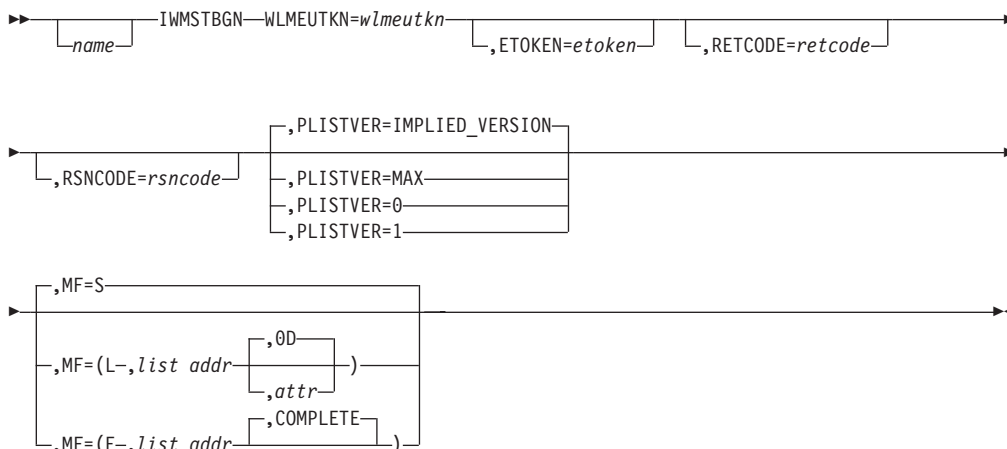
Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

## Performance Implications

None.

## Syntax

### main diagram



## Parameters

The parameters are explained as follows:

### *name*

An optional symbol, starting in column 1, that is the name on the IWMSTBGN macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

### **,ETOKEN=etoken**

An optional output parameter, which will receive the Enclave token.

**To code:** Specify the RS-type address, or register (2)-(12), of an 8-character field.

### **,MF=S**

**,MF=(L,list addr)**

**,MF=(L,list addr,attr)**

**,MF=(L,list addr,OD)**

**,MF=(E,list addr)**

**,MF=(E,list addr,COMPLETE)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.



**,list *addr***

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

**,attr**

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

**,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

**,PLISTVER=IMPLIED\_VERSION****,PLISTVER=MAX****,PLISTVER=0****,PLISTVER=1**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form, when both are assembled with the same level of the system. In this way, MAX ensures that the parameter list does not overwrite nearby storage.

- **0**, if you use the currently available parameters.
- **1**, which supports both the following parameters and those from version 0:

ETOKEN

**To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX
- A decimal value of 0

**,RETCODE=*retcode***

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,RSNCODE=*rsncode***

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

## IWMSTBGN Macro

**WLMEUTKN**=*wlmeutkn*

A required input parameter, execution unit token that was returned by a prior invocation of IWMSSSEL.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

## ABEND Codes

None.

## Return and Reason Codes

When the IWMSTBGN macro returns control to your program:

- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, if you coded RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 106. Return and Reason Codes for the IWMSTBGN Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> <code>IwmRetCodeOk</code> <b>Meaning:</b> Successful completion. <b>Action:</b> None required.
4	—	<b>Equate Symbol:</b> <code>IwmRetCodeWarning</code> <b>Meaning:</b> Successful completion, unusual conditions noted.
4	xxxx041F	<b>Equate Symbol:</b> <code>IwmRsnCodeExecEnvChanged</code> <b>Meaning:</b> The execution environment has changed while the requested function is in progress. <b>Action:</b> None required.
8	—	<b>Equate Symbol:</b> <code>IwmRetCodeInvocError</code> <b>Meaning:</b> Invalid invocation environment or parameters.
8	xxxx0801	<b>Equate Symbol:</b> <code>IwmRsnCodeSrbMode</code> <b>Meaning:</b> Caller is in SRB mode. <b>Action:</b> Avoid requesting this function while in SRB mode.
8	xxxx0803	<b>Equate Symbol:</b> <code>IwmRsnCodeDisabled</code> <b>Meaning:</b> Caller is disabled. <b>Action:</b> Avoid requesting this function while disabled.
8	xxxx0804	<b>Equate Symbol:</b> <code>IwmRsnCodeLocked</code> <b>Meaning:</b> Caller is locked. <b>Action:</b> Avoid requesting this function while locked.

Table 106. Return and Reason Codes for the IWMSTBGN Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx080B	<p><b>Equate Symbol:</b> lwmRsnCodeBadPl</p> <p><b>Meaning:</b> Error accessing parameter list.</p> <p><b>Action:</b> Check for possible storage overlay. Also check if you call this macro in 64-bit address mode. Refer to the description of reason code xxxx089E for further information.</p>
8	xxxx0810	<p><b>Equate Symbol:</b> lwmRsnCodeEutFrr</p> <p><b>Meaning:</b> Caller has EUT FRR established.</p> <p><b>Action:</b> Avoid requesting this function with an EUT FRR set.</p>
8	xxxx0823	<p><b>Equate Symbol:</b> lwmRsnCodeDatoff</p> <p><b>Meaning:</b> Caller invoked service while DATOFF</p> <p><b>Action:</b> Avoid requesting this function in this environment.</p>
8	xxxx0824	<p><b>Equate Symbol:</b> lwmRsnCodeAmode24</p> <p><b>Meaning:</b> Caller invoked service but was in 24 bit addressing mode.</p> <p><b>Action:</b> Request this function only when you are in 31 bit addressing mode.</p>
8	xxxx0825	<p><b>Equate Symbol:</b> lwmRsnCodeAscModeNotPrimary</p> <p><b>Meaning:</b> Caller invoked service but was not DAT on Primary ASC mode.</p> <p><b>Action:</b> Avoid requesting this function in this environment.</p>
8	xxxx0826	<p><b>Equate Symbol:</b> lwmRsnCodeTaskTerm</p> <p><b>Meaning:</b> Caller invoked service while task termination is in progress for the task associated with the owner.</p> <p><b>Action:</b> Avoid requesting this function while task termination is in progress.</p>
8	xxxx0828	<p><b>Equate Symbol:</b> lwmRsnCodeBadVersion</p> <p><b>Meaning:</b> Version number in parameter list is not valid or version length field is not valid.</p> <p><b>Action:</b> Check for possible storage overlay of the parameter list.</p>
8	xxxx083A	<p><b>Equate Symbol:</b> lwmRsnCodeBadEnclave</p> <p><b>Meaning:</b> Enclave token does not pass verification.</p> <p><b>Action:</b> Check for possible storage overlay of the Enclave token, or asynchronous events which may have deleted the Enclave.</p>
8	xxxx0840	<p><b>Equate Symbol:</b> lwmRsnCodeServiceNotEnabled</p> <p><b>Meaning:</b> Caller's space connection is not enabled for this service</p> <p><b>Action:</b> Make sure that SERVER_MANAGER=YES is specified on the IWMCONN request to enable this service.</p>

## IWMSTBGN Macro

Table 106. Return and Reason Codes for the IWMSTBGN Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0841	<p><b>Equate Symbol:</b> IwmRsnCodeXmemMode</p> <p><b>Meaning:</b> Caller is in cross-memory mode.</p> <p><b>Action:</b> Request this function only when you are not in cross-memory mode.</p>
8	xxxx0842	<p><b>Equate Symbol:</b> IwmRsnCodeNoWLMConnect</p> <p><b>Meaning:</b> Caller's space is not connected to WLM.</p> <p><b>Action:</b> Invoke the IWMCONN macro before invoking this macro.</p>
8	xxxx0850	<p><b>Equate Symbol:</b> IwmRsnCodeBeginEnvOutstanding</p> <p><b>Meaning:</b> Caller is already operating under an outstanding Begin environment.</p> <p><b>Action:</b> Avoid requesting this function in this environment.</p>
8	xxxx0851	<p><b>Equate Symbol:</b> IwmRsnCodeSecEnvOutstanding</p> <p><b>Meaning:</b> Caller is already operating under an outstanding security environment.</p> <p><b>Action:</b> Avoid requesting this function while there is a task level security environment outstanding.</p>
8	xxxx0852	<p><b>Equate Symbol:</b> IwmRsnCodeExecTokenNotCorrect</p> <p><b>Meaning:</b> The execution unit token does not identify a previously selected work unit.</p> <p><b>Action:</b> Verify that you have coded the WLMEUTKN parameter correctly.</p>
8	xxxx0857	<p><b>Equate Symbol:</b> IwmRsnCodeAlreadyInEnclave</p> <p><b>Meaning:</b> Current dispatchable workunit is already in an Enclave.</p> <p><b>Action:</b> Avoid requesting this function while the caller is already in an Enclave.</p>
8	xxxx085A	<p><b>Equate Symbol:</b> IwmRsnCodeSelectedWorkActive</p> <p><b>Meaning:</b> The selected work element associated with the input execution unit token is already in execution.</p> <p><b>Action:</b> You may have invoked IWMSTBGN from multiple tasks in the server address space passing the same WLMEUTKN. Avoid requesting this function in this environment.</p>
8	xxxx089E	<p><b>Equate Symbol:</b> IwmRsnCodeServiceAModeMismatch</p> <p><b>Meaning:</b> The caller is in 64-bit address mode and tried to invoke a service macro that is only enabled for a 31-bit environment.</p> <p><b>Action:</b> Use the 64-bit enabled service macro (IWM4STBG) or change the address mode of the caller to 31-bit.</p>
C	—	<p><b>Equate Symbol:</b> IwmRetCodeEnvError</p> <p><b>Meaning:</b> Environmental error.</p>

Table 106. Return and Reason Codes for the IWMSTBGN Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
C	xxxx0C17	<p><b>Equate Symbol:</b> IwmRsnCodeSecEnvCreateFailed</p> <p><b>Meaning:</b> A user security environment cannot be created.</p> <p><b>Action:</b> Verify that the userid is defined to RACF or check the SAF installation exit routine to enable the function.</p>
10	—	<p><b>Equate Symbol:</b> IwmRetCodeCompError</p> <p><b>Meaning:</b> Component error.</p> <p><b>Action:</b> Contact your system programmer.</p>

## Example

Suppose a work request was selected using IWMSSSEL and the execution unit token returned by IWMSSSEL is WLMEUTKN.

To establish the environment to process the work request:

```

IWMSTBGN WLMEUTKN=WLMEUTKN,RETCODE=RC,RSNCODE=RSN
*
* Storage areas
*
WLMEUTKN DS    CL8           Contains the execution unit
*                               token that was returned by
*                               IWMSSSEL
RC        DS    F            Return code
RSN       DS    F            Reason code

```

## IWMSTEND – End a Request from a Caller’s Work Manager Queue

IWMSTEND removes the environment which was previously established using IWMSTBGN to process a work request. The caller must invoke IWMSTEND from the same task that invoked IWMSTBGN. IWMSTEND removes the caller’s task from the enclave associated with the request. IWMSTEND deletes the security environment if one was previously established by IWMSTBGN.

### Note

It is recommended to use the new equivalent service IWM4STEN introduced with z/OS V1R6. IWM4STEN also supports 64-bit address space. For further information refer to Chapter 82, “IWM4STEN – End a Request from a Caller’s Work Manager Queue,” on page 691.

## Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Problem state. Any PSW key
<b>Dispatchable unit mode:</b>	Task
<b>Cross memory mode:</b>	PASN=HASN=SASN
<b>AMODE:</b>	31-bit
<b>ASC mode:</b>	Primary
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks may be held.

## IWMSTEND Macro

**Control parameters:** Control parameters must be in the primary address space.

## Programming Requirements

1. The macro CVT must be included to use this macro.
2. The macro IWMYCON must be included to use this macro.
3. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
4. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded from comparison with the reason code values described above. The constant, IWMSRNCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.

## Restrictions

The caller cannot have an EUT FRR established.

## Input Register Information

Before issuing the IWMSTEND macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

## Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Reason code if GR15 return code is non-zero
1	Used as work register by the system
2-13	Unchanged
14	Used as work registers by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

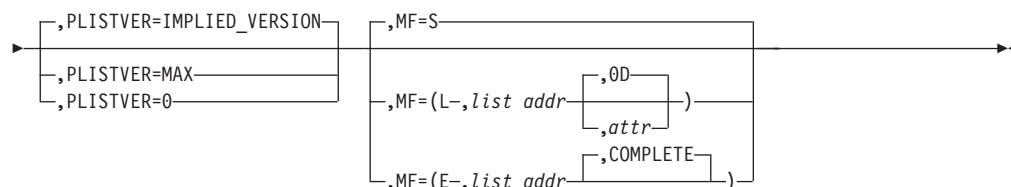
## Performance Implications

None.

## Syntax

### main diagram

```
→ [name] IWMSTEND—WLMEUTKN=wlmeutkn [ , RETCODE=retcode ] [ , RSNCODE=rsncode ] →
```



## Parameters

The parameters are explained as follows:

### *name*

An optional symbol, starting in column 1, that is the name on the IWMSTEND macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

### **,MF=S**

**,MF=(L, list addr)**

**,MF=(L, list addr, attr)**

**,MF=(L, list addr, 0D)**

**,MF=(E, list addr)**

**,MF=(E, list addr, COMPLETE)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

### *,list addr*

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

### *,attr*

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

### **,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

### **,PLISTVER=IMPLIED\_VERSION**

**,PLISTVER=MAX**

**,PLISTVER=0**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When

## IWMSTEND Macro

using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form, when both are assembled with the same level of the system. In this way, MAX ensures that the parameter list does not overwrite nearby storage.

- **0**, if you use the currently available parameters.

**To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX
- A decimal value of 0

**,RETCODE=retcode**

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**,RSNCODE=rsncode**

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

**WLMEUTKN=wlmeutkn**

A required input parameter, execution unit token that was specified on the prior invocation of IWMSTBGN.

**To code:** Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

## ABEND Codes

None.

## Return and Reason Codes

When the IWMSTEND macro returns control to your program:

- GPR 15 (and *retcode*, if you coded RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, if you coded RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.



Table 107. Return and Reason Codes for the IWMSTEND Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> IwmRetCodeOk <b>Meaning:</b> Successful completion. <b>Action:</b> None required.
4	—	<b>Equate Symbol:</b> IwmRetCodeWarning <b>Meaning:</b> Successful completion, unusual conditions noted.
4	xxxx041C	<b>Equate Symbol:</b> IwmRsnCodeNotEnclave <b>Meaning:</b> The current dispatchable workunit is not associated with an Enclave. <b>Action:</b> None required.
4	xxxx041F	<b>Equate Symbol:</b> IwmRsnCodeExecEnvChanged <b>Meaning:</b> The execution environment has changed while the requested function is in progress. <b>Action:</b> None required.
4	xxxx042F	<b>Equate Symbol:</b> IwmRsnCodeSecondaryWorkDeleted <b>Meaning:</b> There were secondary work requests queued to this server task. The caller was expected to process them using IWMSSEM before calling IWMSTEND. The secondary work requests were deleted. <b>Action:</b> Select all secondary work requests before issuing IWMSTEND
8	—	<b>Equate Symbol:</b> IwmRetCodeInvocError <b>Meaning:</b> Invalid invocation environment or parameters.
8	xxxx0801	<b>Equate Symbol:</b> IwmRsnCodeSrbMode <b>Meaning:</b> Caller is in SRB mode. <b>Action:</b> Avoid requesting this function while in SRB mode.
8	xxxx0803	<b>Equate Symbol:</b> IwmRsnCodeDisabled <b>Meaning:</b> Caller is disabled. <b>Action:</b> Avoid requesting this function while disabled.
8	xxxx0804	<b>Equate Symbol:</b> IwmRsnCodeLocked <b>Meaning:</b> Caller is locked. <b>Action:</b> Avoid requesting this function while locked.
8	xxxx080B	<b>Equate Symbol:</b> IwmRsnCodeBadPl <b>Meaning:</b> Error accessing parameter list. <b>Action:</b> Check for possible storage overlay. Also check if you call this macro in 64-bit address mode. Refer to the description of reason code xxxx089E for further information.
8	xxxx0810	<b>Equate Symbol:</b> IwmRsnCodeEutFrr <b>Meaning:</b> Caller has EUT FRR established. <b>Action:</b> Avoid requesting this function with an EUT FRR set.

## IWMSTEND Macro

Table 107. Return and Reason Codes for the IWMSTEND Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0823	<p><b>Equate Symbol:</b> IwmRsnCodeDatoff</p> <p><b>Meaning:</b> Caller invoked service while DATOFF</p> <p><b>Action:</b> Avoid requesting this function in this environment.</p>
8	xxxx0824	<p><b>Equate Symbol:</b> IwmRsnCodeAmode24</p> <p><b>Meaning:</b> Caller invoked service but was in 24 bit addressing mode.</p> <p><b>Action:</b> Request this function only when you are in 31 bit addressing mode.</p>
8	xxxx0825	<p><b>Equate Symbol:</b> IwmRsnCodeAscModeNotPrimary</p> <p><b>Meaning:</b> Caller invoked service but was not DAT on Primary ASC mode.</p> <p><b>Action:</b> Avoid requesting this function in this environment.</p>
8	xxxx0828	<p><b>Equate Symbol:</b> IwmRsnCodeBadVersion</p> <p><b>Meaning:</b> Version number in parameter list is not valid or version length field is not valid.</p> <p><b>Action:</b> Check for possible storage overlay of the parameter list.</p>
8	xxxx0841	<p><b>Equate Symbol:</b> IwmRsnCodeXmemMode</p> <p><b>Meaning:</b> Caller invoked service but was in cross-memory mode.</p> <p><b>Action:</b> Avoid requesting this function in cross-memory mode.</p>
8	xxxx084F	<p><b>Equate Symbol:</b> IwmRsnCodeWrongExecToken</p> <p><b>Meaning:</b> Current dispatchable workunit is not associated with the input execution unit token.</p> <p><b>Action:</b> Check for possible storage overlay of the execution unit token.</p>
8	xxxx0859	<p><b>Equate Symbol:</b> IwmRsnCodeEnclaveSubTaskExists</p> <p><b>Meaning:</b> The current dispatchable workunit has residual subtasks propagated to the Enclave which are still associated with the Enclave.</p> <p><b>Action:</b> Avoid requesting this function in this environment.</p>
8	xxxx089E	<p><b>Equate Symbol:</b> IwmRsnCodeServiceAModeMismatch</p> <p><b>Meaning:</b> The caller is in 64-bit address mode and tried to invoke a service macro that is only enabled for a 31-bit environment.</p> <p><b>Action:</b> Use the 64-bit enabled service macro (IWM4STEN) or change the address mode of the caller to 31-bit.</p>
10	—	<p><b>Equate Symbol:</b> IwmRetCodeCompError</p> <p><b>Meaning:</b> Component error.</p> <p><b>Action:</b> Contact your system programmer.</p>

## Example

To remove the environment which was previously established using IWMSTBGN:

```

IWMSTEND WLMEUTKN=WLMEUTKN,RETCode=RC,RSNCode=RSN
*
* Storage areas
*
WLMEUTKN DS    CL8           Contains the execution unit
*                               token that was specified on
*                               the prior invocation of
*                               IWMSTBGN
RC        DS    F            Return code
RSN       DS    F            Reason code

```

---

## IWMTAFF — WLM Temporal Affinity Service

The IWMTAFF service should be used to inform WLM when a temporal affinity for a specific server region starts and when it ends. WLM will ensure that server regions will not be terminated as long as temporal affinities exist.

The caller must have previously connected to WLM using the IWMCONN as server or as queue manager.

### Note

It is recommended to use the new equivalent service IWM4TAF introduced with z/OS V1R6. IWM4TAF also supports 64-bit address space. For further information refer to Chapter 83, "IWM4TAF — WLM Temporal Affinity Service," on page 697.

## Environment

The requirements for the caller are:

<b>Minimum authorization:</b>	Problem state. Any PSW key
<b>Dispatchable unit mode:</b>	Task
<b>Cross memory mode:</b>	PASN=HASN=SASN
<b>AMODE:</b>	31-bit
<b>ASC mode:</b>	Primary
<b>Interrupt status:</b>	Enabled for I/O and external interrupts
<b>Locks:</b>	No locks may be held.
<b>Control parameters:</b>	Control parameters must be in the primary address space.

## Programming Requirements

1. Make sure no EUT FRRs are established.
2. The macro CVT must be included to use this macro.
3. The macro IWMYCON must be included to use this macro.
4. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
5. Note that the high order halfword of register 0, and the reason code variable when specified, may be non-zero and represents diagnostic data which is NOT part of the external interface. The high order halfword should thus be excluded

## IWMTAFF Macro

from comparison with the reason code values described above. The constant, IWMSRNCODE\_MASK\_CONST defined in IWMYCON, may be used for this purpose.

## Restrictions

- This macro may not be used during task/address space termination.
- Before using this macro the caller must connect to WLM via IWMCONN  
Server\_Manager=YES, Server\_Type=Queue or IWMCONN  
Queue\_Manager=YES.

## Input Register Information

Before issuing the IWMTAFF macro, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

## Output Register Information

When control returns to the caller, the GPRs contain:

Register	Contents
0	Reason code if GR15 return code is non-zero
1	Used as work registers by the system
2-13	Unchanged
14	Used as work registers by the system
15	Return code

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

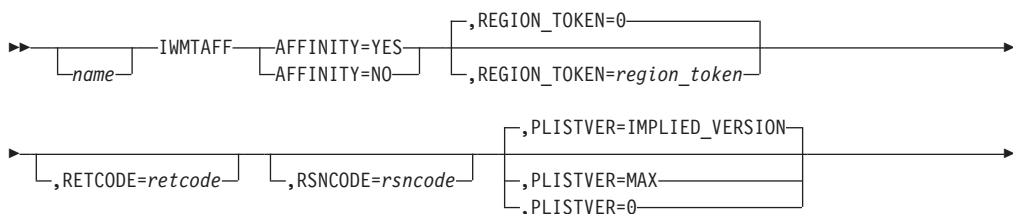
Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

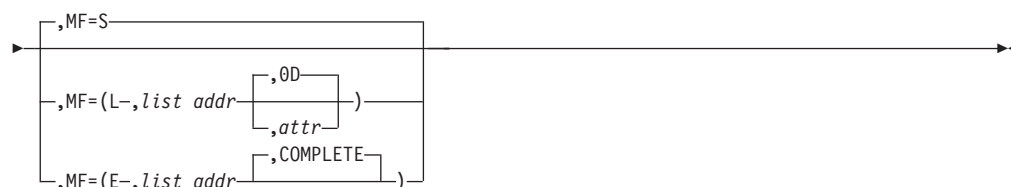
## Performance Implications

None.

## Syntax

### main diagram





## Parameters

The parameters are explained as follows:

### *name*

An optional symbol, starting in column 1, that is the name on the IWMTAFF macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

### **AFFINITY=YES**

### **AFFINITY=NO**

A required parameter indicating whether a temporal affinity begins or ends

### **AFFINITY=YES**

A new temporal affinity for the server region begins. WLM will ensure that the server regions is not terminated before all temporal affinity have ended.

### **AFFINITY=NO**

A temporal affinity for the server region has ended. WLM will start to terminate server regions if all temporal affinities have ended.

### **,MF=S**

### **,MF=(L,list addr)**

### **,MF=(L,list addr,attr)**

### **,MF=(L,list addr,OD)**

### **,MF=(E,list addr)**

### **,MF=(E,list addr,COMPLETE)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

### *,list addr*

The name of a storage area to contain the parameters. For MF=S and MF=E, this can be an RS-type address or an address in register (1)-(12).

### *,attr*

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

### **,COMPLETE**

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

### **,PLISTVER=IMPLIED\_VERSION**

### **,PLISTVER=MAX**

### **,PLISTVER=0**

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED\_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED\_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form, when both are assembled with the same level of the system. In this way, MAX ensures that the parameter list does not overwrite nearby storage.

- **0**, if you use the currently available parameters.

**To code:** Specify one of the following:

- IMPLIED\_VERSION
- MAX
- A decimal value of 0

### **,REGION\_TOKEN=region\_token**

### **,REGION\_TOKEN=0**

An optional input parameter, which contains the region token. The region token is not required if the macro is invoked from the server region for which the temporal affinity should be started or stopped. The region token must be used if the services is used from the queueing manager. The region token is returned by the IWMSSSEL macro.

The caller must be supervisor state or have PSW key mask 0-7 authority to use this service with the REGION\_TOKEN parameter.

Coding REGION\_TOKEN=0 is equivalent to omitting the REGION\_TOKEN keyword. The default is 0.

**To code:** Specify the RS-type address, or address in register (2)-(12), of a 16-character field.

### **,RETCODE=retcode**

An optional output parameter into which the return code is to be copied from GPR 15.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

### **,RSNCODE=rsncode**

An optional output parameter into which the reason code is to be copied from GPR 0.

**To code:** Specify the RS-type address of a fullword field, or register (2)-(12).

## ABEND Codes

None.

## Return and Reason Codes

When the IWMTAFF macro returns control to your program:

- GPR 15 (and *retcode*, when you code RETCODE) contains a return code.
- When the value in GPR 15 is not zero, GPR 0 (and *rsncode*, when you code RSNCODE) contains reason code.

The following table identifies the hexadecimal return and reason codes and the equate symbol associated with each reason code. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 108. Return and Reason Codes for the IWMTAFF Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	<b>Equate Symbol:</b> <code>IwmRetCodeOk</code> <b>Meaning:</b> Successful completion. <b>Action:</b> None required.
4	—	<b>Equate Symbol:</b> <code>IwmRetCodeWarning</code> <b>Meaning:</b> Successful completion, unusual conditions noted. <b>Action:</b> None required.
4	xxxx0439	<b>Equate Symbol:</b> <code>IwmRsnCodeNoAffinityFound</code> <b>Meaning:</b> The service has been invoked to tell WLM that an existing server region affinity has been terminated but WLM has no affinity defined for this server region. <b>Action:</b> If region token was not specified make sure to use the service properly at the beginning and end of each affinity. If the region token has been defined make sure that it is used for the correct server region.
4	xxxx043A	<b>Equate Symbol:</b> <code>IwmRsnCodeRegionNotFound</code> <b>Meaning:</b> The region token does not identify a valid server region. <b>Action:</b> Please specify the correct region token.
8	—	<b>Equate Symbol:</b> <code>IwmRetCodeInvocError</code> <b>Meaning:</b> Invalid invocation environment or parameters.
8	xxxx0801	<b>Equate Symbol:</b> <code>IwmRsnCodeSrbMode</code> <b>Meaning:</b> Caller is in SRB mode. <b>Action:</b> Avoid requesting this function while in SRB mode.
8	xxxx0803	<b>Equate Symbol:</b> <code>IwmRsnCodeDisabled</code> <b>Meaning:</b> Caller is disabled. <b>Action:</b> Avoid requesting this function while disabled.
8	xxxx0804	<b>Equate Symbol:</b> <code>IwmRsnCodeLocked</code> <b>Meaning:</b> Caller is locked. <b>Action:</b> Avoid requesting this function while locked.

Table 108. Return and Reason Codes for the IWMTAFF Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx080B	<p><b>Equate Symbol:</b> IwmRsnCodeBadPl</p> <p><b>Meaning:</b> Error accessing parameter list.</p> <p><b>Action:</b> Check for possible storage overlay. Also check if you call this macro in 64-bit address mode. Refer to the description of reason code xxxx089E for further information.</p>
8	xxxx0824	<p><b>Equate Symbol:</b> IwmRsnCodeAmode24</p> <p><b>Meaning:</b> Caller invoked service but was in 24 bit addressing mode.</p> <p><b>Action:</b> Request this function only when you are in 31 bit addressing mode.</p>
8	xxxx0825	<p><b>Equate Symbol:</b> IwmRsnCodeAscModeNotPrimary</p> <p><b>Meaning:</b> Caller invoked service but was not in primary ASC mode.</p> <p><b>Action:</b> Avoid requesting this function in this environment.</p>
8	xxxx0828	<p><b>Equate Symbol:</b> IwmRsnCodeBadVersion</p> <p><b>Meaning:</b> Version number in parameter list is not valid.</p> <p><b>Action:</b> Check for possible storage overlay of the parameter list.</p>
8	xxxx0840	<p><b>Equate Symbol:</b> IwmRsnCodeServiceNotEnabled</p> <p><b>Meaning:</b> Caller's space connection is not enabled for this service</p> <p><b>Action:</b> Make sure that SERVER_MANAGER=YES and SERVER_TYPE=QUEUE is specified on the IWMTAFF request to enable this service.</p>
8	xxxx0841	<p><b>Equate Symbol:</b> IwmRsnCodeXmemMode</p> <p><b>Meaning:</b> Caller is in cross-memory mode.</p> <p><b>Action:</b> Request this function only when you are not in cross-memory mode.</p>
8	xxxx0842	<p><b>Equate Symbol:</b> IwmRsnCodeNoWLMConnect</p> <p><b>Meaning:</b> Caller's space is not connected to WLM.</p> <p><b>Action:</b> Invoke the IWMTAFF macro before invoking this macro.</p>
8	xxxx084D	<p><b>Equate Symbol:</b> IwmRsnCodeNotAuthConnect</p> <p><b>Meaning:</b> The caller must be supervisor state or have PSW key mask 0-7 authority to use the requested WLM service. This applies only if the caller provides a region token for a server address space for which it wants to set the affinity.</p> <p><b>Action:</b> Avoid requesting this function in this environment.</p>
8	xxxx089E	<p><b>Equate Symbol:</b> IwmRsnCodeServiceAModeMismatch</p> <p><b>Meaning:</b> The caller is in 64-bit address mode and tried to invoke a service macro that is only enabled for a 31-bit environment.</p> <p><b>Action:</b> Use the 64-bit enabled service macro (IWMTAFF) or change the address mode of the caller to 31-bit.</p>



Table 108. Return and Reason Codes for the IWMTAFF Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
10	—	<p><b>Equate Symbol:</b> IwmRetCodeCompError</p> <p><b>Meaning:</b> Component error.</p> <p><b>Action:</b> Contact your system programmer.</p>

## Example

To start a temporal affinity from the server region, specify the following:

```

IWMTAFF AFFINITY=YES
        RETCODE=RC,
        RSNCODE=RSN
*
* Storage areas
*
RC      DS    F           Return code
RSN    DS    F           Reason code
    
```



---

## Appendix E. Notices

This information was developed for products and services offered in the USA.

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 conditions, 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.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

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.

---

## Programming Interface Information

This book documents intended Programming Interfaces that allow the customer to write programs to obtain the services of z/OS.

---

## Trademarks

The following terms are trademarks for the IBM Corporation in the United States and/or other countries:

- AD/Cycle
- C/370
- CBIPO
- CICS
- CICS/ESA
- DB2
- DFSMS/MVS
- ESCON
- Extended Services
- Hiperbatch
- Hiperspace
- IBM
- IBMLink
- IMS/ESA
- MVS/ESA
- OS/2
- OS/390
- PR/SM
- RACF
- Resource Link
- Resource Measurement Facility
- RETAIN
- RMF
- S/390
- SAA
- Sysplex
- System/390
- Virtualization Engine
- VTAM
- z/Architecture
- z/OS
- z/OS.e
- zSeries
- z/VM

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, and service names may be trademarks or service marks of others.



---

## Appendix F. 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:

[www.ibm.com/servers/eserver/zseries/zos/bkserv/](http://www.ibm.com/servers/eserver/zseries/zos/bkserv/)





---

# Index

## Special characters

\$SRMBEST 115  
\$SRMDI00 115  
\$SRMDInn 115  
\$SRMDUMP 115  
\$SRMGOD1-5 115  
\$SRMGOOD 115  
\$SRMSnnn 115

## Numerics

64-bit address space  
WLM services supporting 14

## A

accessibility 909  
Address Space Paging Plots 124  
administrative application services  
definition 14  
installing a service definition 103  
summary 14  
using 103  
application environment  
definition 69  
audit information  
SMF type 99 13

## C

C interfaces 761  
calculation of server weights 89  
cap slices 120  
CheckSchEnv, C interface 761  
classification rules  
querying 110  
client  
definition 85  
command  
syntax diagrams xxxiii  
ConnectExportImport, C interface 761  
ConnectServer, C interface 761  
ConnectWorkMgr, C interface 761  
ContinueWorkUnit, C interface 761  
CreateWorkUnit, C interface 761

## D

DeleteWorkUnit, C interface 761  
dependent enclave 34  
using 49  
disability 909  
DisconnectServer, C interface 761  
dispatching priority 118  
distributed work  
services 27

DNS C interfaces 763  
donors 116

## E

enclave  
creating 31, 33  
delays 31  
deleting 49  
dependent 34  
dependent, using 49  
independent 33  
independent, using 48  
leaving 32  
managing work in 48  
multisystem 35  
performance management of address spaces 45  
querying classification information 31, 49  
querying enclave status of dispatchable units 49  
resource accounting 46  
resource use 31  
scheduling SRBs 31  
using 31  
enclave services  
definition 6  
summary 6  
ENF event code  
for reporting intervals 91  
ENF event code 41 91  
ENQ SYSZWLM QNAME 108  
Enterprise Workload Manager 7  
EWLM 7  
execution delay monitoring services  
definition 4  
single address space transaction manager 20  
summary 5  
expanded storage policy 119  
ExportWorkUnit, C interface 761  
extracting a service definition  
example 109  
ExtractWorkUnit, C interface 761

## I

ImportWorkUnit, C interface 761  
independent enclave 33  
using 48  
installing a service definition  
example 108  
interfaces, C 761  
internal service class  
names 115  
IWM4AEDF macro 523, 533  
IWM4CON  
using 61, 73  
IWM4CON macro 533, 553

IWM4DIS  
   using 61, 73  
 IWM4DIS macro 553, 561  
 IWM4ECRE  
   using 61, 73  
 IWM4ECRE macro 561, 573  
 IWM4EDEL  
   using 61, 73  
 IWM4EDEL macro 573, 580  
 IWM4HLTH macro 581, 587  
 IWM4MCHS macro 587, 595  
 IWM4MCRE macro 595, 604  
 IWM4MDRG macro 605, 611  
 IWM4MINI macro 611, 625  
 IWM4MREG macro 627, 635  
 IWM4QDE  
   using 61  
 IWM4QDE macro 635, 641  
 IWM4QIN  
   using 61  
 IWM4QIN macro 641, 653  
 IWM4SLI macro 653, 661  
 IWM4SRSC macro 661, 669  
 IWM4SSL  
   using 61  
 IWM4SSL macro 669, 677  
 IWM4SSM macro 677, 683  
 IWM4STBG  
   using 61  
 IWM4STBG macro 683, 691  
 IWM4STEN  
   using 61  
 IWM4STEN macro 691, 697  
 IWM4TAF  
   using 68  
 IWM4TAF macro 697, 703  
 IWMAEDEF macro 765, 773  
 IWMCLSFY macro 135, 151  
 IWMCNTN macro 151, 161  
 IWMCONN macro 773, 792  
 IWMCPAFN macro 161, 169  
 IWMCQRY macro 169, 177  
 IWMDEXTR macro 177, 185  
 IWMDINST macro 185, 195  
 IWMDISC macro 792, 798  
 IWMDNDRG, C interface 763  
 IWMDNGRP, C interface 763  
 IWMDNREG, C interface 763  
 IWMDNSRV, C interface 763  
 IWMEBLK macro 195, 201, 203  
 IWMECQRY macro 203, 209  
   example 49  
 IWMECREA macro 798, 809  
 IWMEDELE macro 809, 816  
 IWMEDELE macro 209, 215  
 IWMEGREG macro 215, 223  
 IWMEJOIN macro 223, 229  
 IWMELEAV macro 229, 235  
 IWMEQTME macro 235, 241  
 IWMEREG macro 241, 247  
 IWMERES macro 247, 255  
 IWMESEDES  
   using 79  
 IWMESEDES macro 413, 419  
 IWMESEQRY  
   using 79  
 IWMESEQRY macro 419, 425  
 IWMESESET  
   using 79  
 IWMESESET macro 425, 431  
 IWMESEVAL  
   using 79  
 IWMESEVAL macro 431, 437  
 IWMSINF  
   using 67  
 IWMSINF macro 437, 445  
 IWMSLIM macro 864, 871  
 IWMSRDNS macro 445, 451  
 IWMSRDRS macro 451, 457  
 IWMSRFSV  
   using 73  
 IWMSRFSV macro 457, 465  
 IWMSRSRG macro 465, 473  
 IWMSRSRS macro 473, 483  
 IWMSSEL macro 871, 877  
 IWMSSEM macro 877, 884  
 IWMSTBGN macro 884, 891  
 IWMSTEND macro 891, 897  
 IWMEQRY 49  
   using 61  
 IWMEQRY macro 271, 277  
 IWMESTOP macro 255, 261, 263  
 IWMESTRT macro 263, 270, 271  
 IWMEUBLK macro 277, 283, 285  
 IWMEEXPT macro 285, 293  
 IWMEIMPT macro 293, 301  
 IWMEABNL macro 301, 304  
 IWMECHST macro 816, 822  
 IWMECREA macro 822, 832  
 IWMEDELE macro 305, 309  
 IWMEEXTR macro 309, 317  
 IWMEINIT macro 832, 846  
 IWMEINTFY macro 317, 327  
 IWMERELA macro 327, 337  
 IWMESWCH macro 337, 341  
 IWMEXDC macro 343, 349  
 IWMEXFER macro 349, 356  
 IWMEXRA macro 357, 363  
 IWMEPACT macro 363, 371  
 IWMEPQRY macro 371, 377  
 IWMEQCXIT  
   using 61  
 IWMEQDEL macro 846, 852  
 IWMEQINS macro 852, 864  
 IWMEQVS, C interface 763  
 IWMECOLL macro 93, 377, 385  
   delay state information 94, 96  
   response time information 94, 98  
 IWMERESET macro 385, 393  
 IWMERPT macro 393, 405  
 IWMERQRY macro 405, 413  
   address space information 101  
 IWMESEDES  
   using 79  
 IWMESEDES macro 413, 419  
 IWMESEQRY  
   using 79  
 IWMESEQRY macro 419, 425  
 IWMESESET  
   using 79  
 IWMESESET macro 425, 431  
 IWMESEVAL  
   using 79  
 IWMESEVAL macro 431, 437  
 IWMSINF  
   using 67  
 IWMSINF macro 437, 445  
 IWMSLIM macro 864, 871  
 IWMSRDNS macro 445, 451  
 IWMSRDRS macro 451, 457  
 IWMSRFSV  
   using 73  
 IWMSRFSV macro 457, 465  
 IWMSRSRG macro 465, 473  
 IWMSRSRS macro 473, 483  
 IWMSSEL macro 871, 877  
 IWMSSEM macro 877, 884  
 IWMSTBGN macro 884, 891  
 IWMSTEND macro 891, 897

- IWMSVDEF data area
  - adding extensions 104
- IWMTAFF macro 897, 905
- IWMUEXPT macro 483, 489
- IWMUIMPT macro 489, 495
- IWMWDNSH header file 763
- IWMWMCON
  - description 27
- IWMWMCON macro 495, 503
- IWMWQRY macro 503, 507
- IWMWQWRK macro 507, 513
- IWMWSYSQ macro 515, 523

## J

- JoinWorkUnit, C interface 761

## K

- keyboard 909

## L

- LeaveWorkUnit, C interface 761
- LookAt message retrieval tool xxxii

## M

- message retrieval tool, LookAt xxxii
- MPL out target 119
- MPL-in target 119
- multisystem enclave 35

## N

- Notices 905
- number of server address spaces 121

## P

- performance index 116
- performance monitors
  - services 12
- period MPL delay plot 122
- period paging rate plot 123
- period proportional aggregate speed plot 123
- period ready user average plot 122
- period swap delay plot 122
- policy adjustment 116
- policy management
  - application services 14
- priority unbunching 119

## Q

- query virtual server, C interfaces 763
- QueryMetrics, C interface 761
- QuerySchEnv, C interface 761
- QueryWorkUnitClassification, C interface 761

- queueing manager
  - definition 61
  - services 61
- queueing manager services
  - definition 8
  - guidance information 61
  - summary 8, 61
  - using the queueing manager connect exit 69
- QVS 763

## R

- receivers 116
- resource adjustment 116
- resource requirements 79
- resource states 79
- routing manager
  - definition 73
  - services 73
- routing manager services
  - definition 9
  - guidance information 73
  - summary 9, 10, 73
  - using the routing manager server connect exit 77

## S

- scheduling environment services
  - definition 10, 79
  - guidance information 79
  - services 79
- server
  - calculation of server weights 89
  - de-registration 87
  - definition 85
  - registration 86
- server address spaces, number of 121
- service definition
  - adding extensions 103
  - data validation 106
  - data validity checking 106
  - extracting 109
  - installing 103
- service policy
  - activating 109
- services
  - multiple address space work managers 23
  - work manager calling data manager 22
- shortcut keys 909
- SMF record type 99
  - action codes 707
  - identifying work 114
  - management policy data 118
  - priority table data 126
  - when to use 114
- SMF record Type 99
  - examples 127
  - plots 121
- subsystem work manager
  - guidance information 17
  - services 3

- swap protect time 119
- syntax diagrams
  - how to read xxxiii
- sysplex routing services
  - de-registering 87
  - definition 10, 85
  - example 87
  - routing work 86
  - server registration 86
  - summary 11
  - using 85
  - workload distribution 88
- sysplex routing services, C interfaces 763
- system paging delay plot 121

## U

- UnDoExportWorkUnit, C interface 761
- UnDoImportWorkUnit, C interface 761

## W

- WLM services
  - supporting 64-bit address space 14
- work manager services
  - definition 3
  - single address space work manager 18
  - summary 4
- workload reporting services
  - definition 12
  - when to use 91

---

# Readers' Comments — We'd Like to Hear from You

**z/OS**  
**MVS Programming: Workload**  
**Management Services**

**Publication No. SA22-7619-14**

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 a fax to the following number: FAX (Germany): 07031+16-3456  
FAX (Other Countries): (+49)+7031-16-3456
- Send your comments via e-mail to: s390id@de.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



Fold and Tape

Please do not staple

Fold and Tape



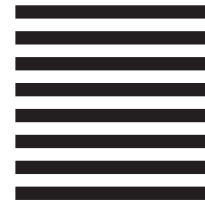
NO POSTAGE  
NECESSARY  
IF MAILED IN THE  
UNITED STATES

# BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Deutschland Entwicklung GmbH  
Department 3248  
Schoenaicher Strasse 220  
D-71032 Boeblingen  
Federal Republic of Germany



Fold and Tape

Please do not staple

Fold and Tape





Program Number: 5694-A01

Printed in USA

SA22-7619-14





Spine information:



z/OS

z/OS MVS Programming: Workload Management  
Services

SA22-7619-14