z/OS

MVS Programming: Workload Management Services

Version 2 Release 2

SC34-2663-04

Note

Before using this information and the product it supports, read the information in "Notices" on page 1087.

This edition applies to Version 2 Release 2 of z/OS (5650-ZOS) and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Figures

Tables

About this information

This document supports z/OS (5650-ZOS). This document describes the z/OS® workload management services. The services are intended for programmers who write authorized programs.

This document is divided into two main parts:

- Part 1, "Using the workload management services," on page 1, which provides an overview of the WLM services and how to use them
- Part 2, "Reference: Workload Management Services," on page 141, which describes each service in detail, including syntax, parameters, and usage examples

Who should use this information

Programmers using assembly 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 the 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 V2R2 Information Roadmap*.

Title	Order number
z/OS MVS Planning: Workload Management	SC34-2662
z/OS Data Areas, Vol 1 (ABEP-DCCB)	n/a
z/OS Data Areas, Vol 2 (DCCD-IEFDOKEY)	n/a
z/OS Data Areas, Vol 3 (IEFDORC-ISGYQCBP)	n/a
z/OS Data Areas, Vol 4 (ISGYQUAA-MCHEAD)	n/a
z/OS Data Areas, Vol 5 (MCSCSA-SNAPX)	n/a
z/OS MVS Initialization and Tuning Guide	SA23-1379
z/OS MVS Initialization and Tuning Reference	SA23-1380
z/OS MVS Programming: Authorized Assembler Services Guide	SA23-1371
z/OS MVS Programming: Authorized Assembler Services Reference EDT-IXG	SA23-1373
z/OS MVS Programming: Authorized Assembler Services Reference SET-WTO	SA23-1375
z/OS MVS System Management Facilities (SMF)	SA38-0667
z/OS XL C/C++ Runtime Library Reference	SC14-7314

Information updates on the web

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

`http://www.ibm.com/systems/z/os/zos/features/wlm/`

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.

For users accessing the Information Center using a screen reader, syntax diagrams are provided in dotted decimal format.

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**

Required
: Required items are displayed on the main path of the horizontal line.

Optional
: 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

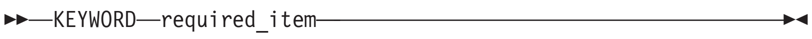
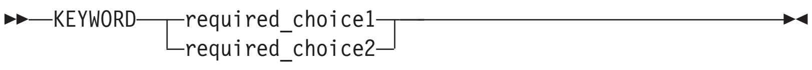
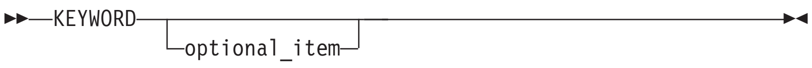
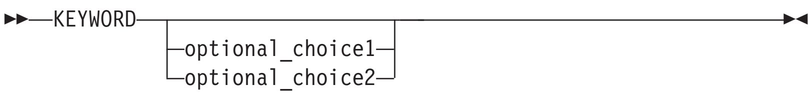
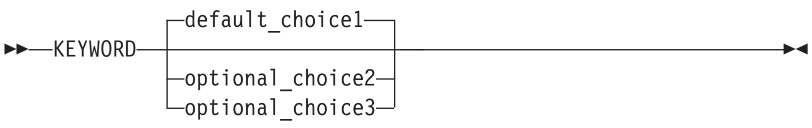
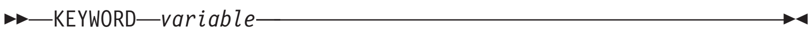
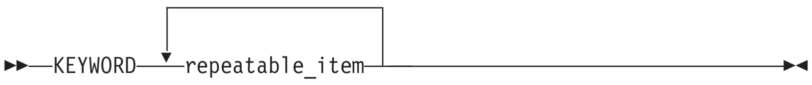
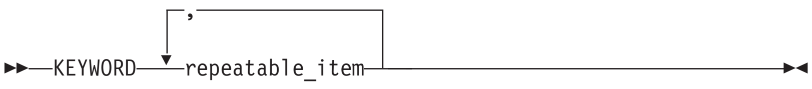
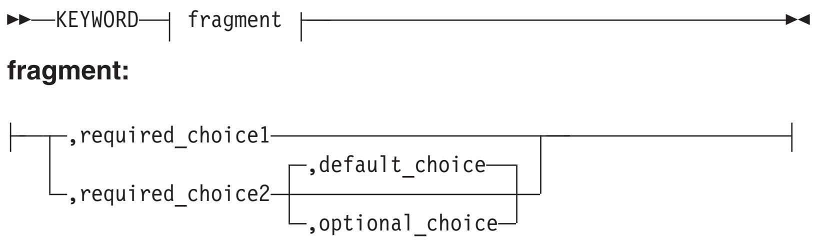
Item	Syntax example
Required item. Required items appear on the main path of the horizontal line. You must specify these items.	
Required choice. A required choice (two or more items) appears in a vertical stack on the main path of the horizontal line. You must choose one of the items in the stack.	
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.	 default_choice1 KEYWORD optional_choice2 optional_choice3
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
Fragment. 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.	 KEYWORD fragment fragment: ,required_choice1 ,required_choice2 ,default_choice ,optional_choice

How to send your comments to IBM

We appreciate your input on this documentation. Please provide us with any feedback that you have, including comments on the clarity, accuracy, or completeness of the information.

Use one of the following methods to send your comments:

Important: If your comment regards a technical problem, see instead "If you have a technical problem."

- Send an email to mhvrcfs@us.ibm.com.
- Send an email from the "Contact us" web page for z/OS (http://www.ibm.com/systems/z/os/zos/webqs.html).

Include the following information:
- Your name and address
- Your email address
- Your phone or fax number
- The publication title and order number:
  z/OS V2R2 MVS Programming: Workload Management Services
  SC34-2663-04
- The topic and page number or URL of the specific information to which your comment relates
- The text of your comment.

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

IBM or any other organizations use the personal information that you supply to contact you only about the issues that you submit.

If you have a technical problem

Do not use the feedback methods that are listed for sending comments. Instead, take one or more of the following actions:
- Visit the IBM Support Portal (support.ibm.com).
- Contact your IBM service representative.
- Call IBM technical support.

Summary of changes

This information includes terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations for the current edition are indicated by a vertical line to the left of the change.

Changes made in z/OS Version 2 Release 2 (V2R2) as updated March 2017

Changed

"IWMPACT — Activate service policy" on page 292 is updated to reflect the new CHECKHISTORY parameter, which specifies the criteria that WLM applies to decide whether or not to discard historical data for service class periods, due to changes that come into effect with a reactivated WLM policy.

Appendix C, "Structure of the XML service definition (DTD)," on page 817 is updated to reflect HonorPriority and MemoryLimit, which support new function that allows WLM administrators to prevent the overflow of specialty-engine-intensive work, in individual service classes, to standard processors.

Changes made in z/OS Version 2 Release 2 (V2R2) as updated December 2015

Changed

Several topics are updated for WLM support of mobile pricing. This support allows WLM administrators to classify transactions in the WLM service definition so that they can benefit from mobile application pricing. The updated topics include the following:

- "Suggested services for a single address space transaction manager" on page 22
- "Suggested services for a work manager that calls a data manager" on page 26
- "Services for multiple address space work managers" on page 27
- "Execution delay monitoring services for multiple address space work managers" on page 29
- "Using the information in IWMWRCAA" on page 103
- "Using the response time information" on page 109
- Appendix C, "Structure of the XML service definition (DTD)," on page 817

Summary of changes for z/OS Version 2 Release 2 (V2R2)

The following changes are made for z/OS Version 2 Release 2 (V2R2). All technical changes for z/OS V2R2 are indicated by a vertical line to the left of the change.

New

The following new information is added in this publication:

- Service "IWM4OPTQ — Query IEAOPT*xx* parameters" on page 677
- Service "IWM4QHLT — Query server health indicators" on page 689

Changes made in z/OS Version 2 Release 1 (V2R1) as updated December 2013

Changed

- The order number for z/OS Version 2 Release 1 (5650-ZOS) has been corrected.
- New SMF type 99 action codes 111-123 and 125-127 have been added in Appendix A, “SMF type 99 action codes,” on page 775.

Changes made in z/OS Version 2 Release 1

See the following publications for specific enhancements for z/OS Version 2 Release 1:

- *z/OS Summary of Message and Interface Changes*, SA23-2300
- *z/OS Introduction and Release Guide*, GA32-0887
- *z/OS Planning for Installation*, GA32-0890
- *z/OS Migration*, GA32-0889

This document contains information previously presented in *z/OS MVS™ Programming: Workload Management Services, SA22-7619-21* which supports z/OS Version 1 Release 13.

New information

The descriptions of the following new macros supporting 64-bit addressing were added.

- IWM4CLSY (see “IWM4CLSY — Classify work” on page 464)
- IWM4EQRY (see “IWM4EQRY — Query an enclave” on page 528)
- IWM4MABN (see “IWM4MABN — Monitor environment abnormal event” on page 543)
- IWM4MDEL (see “IWM4MDEL — Delete delay monitoring environment” on page 571)
- IWM4MXTR (see “IWM4MXTR — Monitoring environment extract service” on page 670)
- IWM4MNTF (see “IWM4MNTF — Notify of work execution completion” on page 606)
- IWM4MRLT (see “IWM4MRLT — Relate monitoring environments (PBs)” on page 623)
- IWM4MSTO (see “IWM4MSTO — Stops a work unit” on page 633)
- IWM4MSTR (see “IWM4MSTR — Indicate the start of a work unit” on page 640)
- IWM4MSWC (see “IWM4MSWC — Monitoring environment switch” on page 648)
- IWM4MUPD (see “IWM4MUPD — Update data for a work unit” on page 655)
- IWM4MXFR (see “IWM4MXFR — Monitoring environment transfer” on page 662)
- IWM4RPT (see “IWM4RPT — Report response time” on page 708)

Updated information

- The descriptions of the IWMDEXTR, IWMESTRT, IWMPACT, and IWM4ECRE macros are updated.

Moved information

The descriptions of the following macros were moved to Appendix E, “WLM services supporting 31-bit addressing only,” on page 827.

- IWMCLSFY (see “IWMCLSFY — Classify work request” on page 837).
- IWMECQRY (see “IWMECQRY — Query enclave classification attributes” on page 879).
- IWMRPT (see “IWMRPT — Report on work request completion” on page 1028).
- IWMMABNL (see “IWMMABNL — Record abnormal event” on page 911).
- IWMMDELE (see “IWMMDELE — Delete the monitoring environment” on page 932).
- IWMMEXTR (see “IWMMEXTR — Monitoring environment extract” on page 936).
- IWMMNTFY (see “IWMMNTFY — Notify of work execution completion” on page 958).
- IWMMRELA (see “IWMMRELA — Relate monitoring environment service” on page 967).
- IWMMSWCH (see “IWMMSWCH — Switch monitoring environment” on page 990).
- IWMMXFER (see “IWMMXFER — Transfer monitoring environment” on page 1001).
- IWMMSTOP (see “IWMMSTOP — Stop a work unit” on page 976).
- IWMMSTRT (see “IWMMSTRT — Indicate the start of a work unit” on page 983).
- IWMMXFER (see “IWMMXFER — Transfer monitoring environment” on page 1001).

Part 1. Using the workload management services

These topics provide an overview of the workload management services.

Chapter 1. Introduction to 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 introduction 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, multiprogramming level (MPL), dispatching, and I/O queueing.

The services for subsystem work managers can be grouped 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 any of the following objectives, consider using either the queueing manager, the routing manager, or enclave services instead of the work manager services:

- 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,

Table 2 summarizes the work manager services.

Table 2. Work manager services

Service	Purpose	Information
IWM4CLSY	Associate an arriving work request with a service class defined in a service policy.	"IWM4CLSY — Classify work" on page 464
IWMWMCON	Override the subsystem name and type previously provided on IWM4CON.	"IWMWMCON — WLM modify connect" on page 428
IWMWQRY	Obtain a service class goal.	"IWMWQRY — Query service" on page 435
IWM4CON	Obtain token that authorizes caller to use other work manager services, and optionally, to supply additional topology information.	"IWM4CON — Connect to workload management" on page 480
IWM4DIS	Disconnect from workload management.	"IWM4DIS — Disconnect from workload management" on page 499

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 summarizes the execution delay monitoring services.

Table 3. Execution delay monitoring services

Service	Purpose	Information
IWMWQWRK	Identify where transactions are executing.	“IWMWQWRK — Query work service” on page 439
IWM4ECRE	Create an enclave.	“IWM4ECRE — Create an enclave” on page 506
IWM4EQRY	Query enclave attributes.	“IWM4EQRY — Query an enclave” on page 528
IWM4MABN	Indicate that an abnormal event has occurred for the work request represented by the input monitoring environment.	“IWM4MABN — Monitor environment abnormal event” on page 543
IWM4MCHS	Record the state (such as ready, waiting, idle) of a work request.	“IWM4MCHS — Change the state of a work request” on page 548
IWM4MCRE	Create a monitoring environment, also called performance block.	“IWM4MCRE — Create delay monitoring environment” on page 559
IWM4MDEL	Delete a delay monitoring environment	“IWM4MDEL — Delete delay monitoring environment” on page 571

Table 3. Execution delay monitoring services (continued)

Service	Purpose	Information
IWM4MINI	Initialize monitoring environment with information about a work request.	"IWM4MINI — Monitoring environment initialization" on page 590
IWM4MNTF	Notify MVS that the execution phase for a work request associated with a monitoring environment has just completed.	"IWM4MNTF — Notify of work execution completion" on page 606
IWM4MRLT	Relate two different monitoring environments that are associated with the same work request.	"IWM4MRLT — Relate monitoring environments (PBs)" on page 623
IWM4STBG	Begin a request from a caller's work manager queue.	"IWM4STBG — WLM begin server transaction service" on page 752
IWM4MSTO	Stop a work unit which has been started by IWM4MSTR.	"IWM4MSTO — Stops a work unit" on page 633
IWM4MSTR	Indicate that a work unit is beginning execution.	"IWM4MSTR — Indicate the start of a work unit" on page 640
IWM4MSWC	Indicate that the delay information for a work request may now also reside in another monitoring environment which is not related to the current environment (Continue) or that there is no further information for the current work request beyond the current environment (Return).	"IWM4MSWC — Monitoring environment switch" on page 648
IWM4MUPD	Update data about a work unit which has been started by IWM4MSTR.	"IWM4MUPD — Update data for a work unit" on page 655
IWM4MXFR	Indicate 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).	"IWM4MXFR — Monitoring environment transfer" on page 662
IWM4MXTR	Extract information about the monitoring environment which was previously passed through IWM4MINI/IWM4MRLT.	"IWM4MXTR — Monitoring environment extract service" on page 670
IWM4RPT	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.	"IWM4RPT — Report response time" on page 708

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 will not 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 9.

Table 4 summarizes the enclave services.

Table 4. Enclave services

Service	Purpose	Information
IEAMSCHD	Schedule an SRB into the enclave.	*z/OS MVS Programming: Authorized Assembler Services Guide*
IWM4EQRY	Query the classification information associated with an enclave.	“IWM4EQRY — Query an enclave” on page 528
IWMEDREG	Deregister an enclave.	“IWMEDREG — Deregister a WLM enclave” on page 187
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.	“IWMEJOIN — Join WLM enclave” on page 200

Table 4. Enclave services (continued)

Service	Purpose	Information
IWMELEAV	Leave an enclave (task only).	"IWMELEAV — Leave WLM enclave" on page 208
IWMEREG	Register an enclave.	"IWMEREG — Register a WLM enclave" on page 222
IWMESQRY	Query whether or not the current dispatchable unit is associated with an enclave.	"IWMESQRY — Query enclave state" on page 236
IWMEXPT	Export an enclave to all systems in a parallel sysplex.	"IWMEXPT — Export a WLM enclave" on page 263
IWMGCORF	Check whether or not certain Application Response Measurement (ARM) flags in a provided EWLM correlator are set.	"IWMGCORF — Get correlator flags" on page 270
IWMIMPT	Import an enclave that has been exported to all systems in a parallel sysplex.	"IWMIMPT — Import an enclave" on page 273
IWM4MSTO	Stop a work unit which has been started by IWMMSTRT.	"IWM4MSTO — Stops a work unit" on page 633
IWM4MSTR	Indicate that a work unit is beginning execution.	"IWM4MSTR — Indicate the start of a work unit" on page 640
IWM4MUPD	Update data about a work unit which has been started by IWMMSTRT.	"IWM4MUPD — Update data for a work unit" on page 655
IWMRQRY	Obtain information about enclave resource consumption and delays.	"IWMRQRY — Collect address space delay information" on page 328
IWMSCORF	Set or clear certain Application Response Measurement (ARM) correlator flags in a provided EWLM correlator.	"IWMSCORF — Set correlator flags" on page 335
IWMUEXPT	Undo an export of an enclave to all systems in a parallel sysplex.	"IWMUEXPT — WLM undo export" on page 416
IWMUIMPT	Undo an import of an enclave.	"IWMUIMPT — WLM undo import" on page 422
IWM4ECRE	Create an enclave.	"IWM4ECRE — Create an enclave" on page 506
IWM4EDEL	Delete a previously created enclave.	"IWM4EDEL — Delete an enclave" on page 520
SYSEVENT ENCASSOC	Allows an enclave running SRBs to be associated with an address space. This way the storage-related resources of the server address space can be managed to the enclave's performance goal.	*z/OS MVS Programming: Authorized Assembler Services Reference SET-WTO*
SYSEVENT ENCSTATE	Indicate that an enclave will be idle for an extended period of time, exempting the enclave from active resource management.	*z/OS MVS Programming: Authorized Assembler Services Reference SET-WTO*

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 *queuing 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 summarizes the queueing manager services.

Table 5. Queueing manager services

Service	Purpose	Information
IWM4AEDF	Define a dynamic application environment.	"IWM4AEDF — WLM define dynamic application environments" on page 454
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.	"IWM4CON — Connect to workload management" on page 480

Table 5. Queueing manager services (continued)

Service	Purpose	Information
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.	"IWM4CON — Connect to workload management" on page 480
IWM4DIS	Remove the caller as a queueing manager or server manager.	"IWM4DIS — Disconnect from workload management" on page 499
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.	"IWM4ECRE — Create an enclave" on page 506
IWM4EDEL	Delete an enclave.	"IWM4EDEL — Delete an enclave" on page 520
IWMESQRY	Query whether or not the current dispatchable unit is associated with an enclave.	"IWMESQRY — Query enclave state" on page 236
IWM4QDE	Delete a work request from the queue for an execution address space.	"IWMQDEL — Delete a request from the queue for an execution address space" on page 1009
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.	"IWM4QIN — Insert a request onto the queue for an execution address space" on page 696
IWMSINF	Obtain the number of server instances to be started by workload management.	"IWMSINF — WLM server manager inform service" on page 368
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.	"IWM4SLI — Application environment limit service" on page 720
IWM4SSL	Select a work request from workload management queues for execution in a server address space. This must be done under a task.	"IWM4SSL — Select a request from a caller's work manager queue" on page 739
IWM4SSM	Select the next secondary work request from the queue associated with the caller's server task.	"IWM4SSM — WLM server select secondary service" on page 746

Table 5. Queueing manager services (continued)

Service	Purpose	Information
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.	"IWM4STBG — WLM begin server transaction service" on page 752
IWM4STEN	Leave the enclave that was joined in IWM4STBG. The server address space has completed its processing of the work request.	"IWM4STEN — End a request from a caller's work manager queue" on page 760
IWM4TAF	Tell workload management when a temporal affinity begins and when it ends.	"IWM4TAF — WLM temporal affinity service" on page 766

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 summarizes 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.	"IWM4CON — Connect to workload management" on page 480
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.	"IWM4CON — Connect to workload management" on page 480
IWM4DIS	Remove the caller as a routing manager.	"IWM4DIS — Disconnect from workload management" on page 499
IWM4ECRE	Create an enclave. This is done in a server address space.	"IWM4ECRE — Create an enclave" on page 506

Table 6. Routing manager services (continued)

Service	Purpose	Information
IWM4EDEL	Delete an enclave.	"IWM4EDEL — Delete an enclave" on page 520
IWMSRFSV	Find the best server for a work request. If no server exists for a request, start one.	"IWMSRFSV — Sysplex routing find server routine" on page 388

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.	"IWMSEDES — Scheduling environments determine execution service" on page 344
IWMSEQRY	Obtain scheduling environment definitions and status.	"IWMSEQRY — Scheduling environments query service" on page 350
IWMSESET	Modify the state setting of a resource.	"IWMSESET — Scheduling environments set resource service" on page 356
IWMSEVAL	Validate a scheduling environment name.	"IWMSEVAL — Scheduling environments validate service" on page 362

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 summarizes 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.	"IWMSRDNS — Get sysplex routing location list" on page 376
IWMSRDRS	Deregister a server.	"IWMSRDRS — Deregister a server for sysplex routing" on page 382
IWMSRSRG	Register an eligible server.	"IWMSRSRG — Register a server for sysplex routing" on page 396
IWMSRSRS	Provide the caller with a list of registered servers and the number of requests that should be routed to each server.	"IWMSRSRS — Sysplex routing information" on page 405
IWM4SRSC	Provide the caller with server-specific routing information to allow for balanced routing decisions.	"IWM4SRSC — Obtain server-specific routing information" on page 728

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.

See "IWMWSYSQ — Query system information" on page 446 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 restarted. At such times, performance monitors should also stop and restart their reporting intervals. For each time that data collection is stopped and restarted in workload management, an event notification facility (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 V1R3, 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 summarizes the workload reporting services, and where the information about them is documented.

Table 9. Summary of workload reporting services

Service	Purpose	Information
IWMPQRY	• Provides the active service policy. • Use it with IWMRCOLL for goal vs. actual information	"IWMPQRY — Query active service policy" on page 300

Table 9. Summary of workload reporting services (continued)

Service	Purpose	Information
IWMRCOLL	Collects: • Workload activity information • Response time information • General delay information • Execution delay state information	"IWMRCOLL — Collect workload activity data" on page 312
IWMRQRY	Provides address space related information: • Server information • Velocity information • General delay information – MPL delay – Swap-in – Resource group capping – CPU delay • Enclave information	"IWMRQRY — Collect address space delay information" on page 328
SYSEVENTs REQASD and REQFASD	Provide information about an address space: • Whether it is a server • Whether its goal is being honored • Whether it was quiesced • Service class, report class • Performance group, report performance group	*z/OS MVS Programming: Authorized Assembler Services Reference SET-WTO*
SYSEVENT REQSRMST	To quickly check: • Active service policy • Installed service definition	*z/OS MVS Programming: Authorized Assembler Services Reference SET-WTO*

Getting information from SMF type 99 records

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. 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 123 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 summarizes the administrative application services.

Table 10. Administrative application services

Service	Purpose	Information
IWMCQRY	Query the classification rules in effect.	"IWMCQRY — Query classification attributes" on page 154
IWMDINST	Install a service definition on the WLM couple data set.	"IWMDINST — Install a service definition" on page 170
IWMDEXTR	Extract a service definition from the WLM couple data set.	"IWMDEXTR — Extract WLM service definition" on page 161
IWMPACT	Activate a service policy.	"IWMPACT — Activate service policy" on page 292

WLM services that support 64-bit addressing

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 program to invoke the 64-bit versions of the services. For example, change IWMCONN to IWM4CON. The prefix of all 64-bit services names is `IWM4`.

Generally, the services that run in 64-bit address mode support the same parameters as their equivalents in 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
         IWM4xxxx 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 that support 64-bit addressing and their equivalents for 31-bit addressing only:

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

WLM service name (31-bit only)	**WLM service name (31-bit and 64-bit)**	**Purpose**
IWMAEDEF. For more information, see "IWMAEDEF — Defining Dynamic Application Environments to Workload Management" on page 828.	IWM4AEDF. For more information, see "IWM4AEDF — WLM define dynamic application environments" on page 454.	Define a dynamic application environment to Workload Manager.
IWMCONN. For more information, see "IWMCONN — Connect to workload management" on page 853.	IWM4CON. For more information, see "IWM4CON — Connect to workload management" on page 480.	Connect to Workload Manager.
IWMCLSFY. For more information, see "IWMCLSFY — Classify work request" on page 837.	IWM4CLSY. For more information, see "IWM4CLSY — Classify work" on page 464	Associate an arriving work request with a service class defined in a service policy.
IWMDISC. For more information, see "IWMDISC — Disconnect from workload management" on page 872.	IWM4DIS. For more information, see "IWM4DIS — Disconnect from workload management" on page 499.	Disconnect from Workload Manager.
IWMECREA. For more information, see "IWMECREA — Create an enclave" on page 885.	IWM4ECRE. For more information, see "IWM4ECRE — Create an enclave" on page 506.	Create an enclave.
IWMEDELE. For more information, see "IWMEDELE — Delete an enclave" on page 896.	IWM4EDEL. For more information, see "IWM4EDEL — Delete an enclave" on page 520.	Delete an enclave.
IWMEQRY. For more information, see "IWMEQRY — Enclave query" on page 904.	IWM4EQRY. For more information, see "IWM4EQRY — Query an enclave" on page 528.	Query enclave attributes.
IWMMABNL. For more information, see "IWMMABNL — Record abnormal event" on page 911.	IWM4MABN. For more information, see "IWM4MABN — Monitor environment abnormal event" on page 543	Indicate that an abnormal event has occurred for the work request represented by the input monitoring environment.
IWMMCHST. For more information, see "IWMMCHST — Monitor change state of work unit" on page 915.	IWM4MCHS. For more information, see "IWM4MCHS — Change the state of a work request" on page 548.	Change state of work request service.
IWMMCREA. For more information, see "IWMMCREA — Create delay monitoring environment" on page 922.	IWM4MCRE. For more information, see "IWM4MCRE — Create delay monitoring environment" on page 559.	Create monitoring environment service.

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

WLM service name (31-bit only)	**WLM service name (31-bit and 64-bit)**	**Purpose**
IWMMDELE. For more information, see "IWMMDELE — Delete the monitoring environment" on page 932.	IWM4MDEL. For more information, see "IWM4MDEL — Delete delay monitoring environment" on page 571	Delete a delay monitoring environment.
IWMMINIT. For more information, see "IWMMINIT — Initialize monitoring environment" on page 943.	IWM4MINI. For more information, see "IWM4MINI — Monitoring environment initialization" on page 590.	Monitor environment initialization.
IWMMNTFY. For more information, see "IWMMNTFY — Notify of work execution completion" on page 958.	IWM4MNTF. For more information, see "IWM4MNTF — Notify of work execution completion" on page 606	Notify MVS that the execution phase for a work request associated with a monitoring environment has just completed.
IWMMRELA. For more information, see "IWMMRELA — Relate monitoring environment service" on page 967.	IWM4MRLT. For more information, see "IWM4MRLT — Relate monitoring environments (PBs)" on page 623	Relate two different monitoring environments that are associated with the same work request.
IWMMSTOP. For more information, see "IWMMSTOP — Stop a work unit" on page 976	IWM4MSTO. For more information, see "IWM4MSTO — Stops a work unit" on page 633	Stop a work unit which has been started by IWMMSTRT.
IWMMSTRT. For more information, see "IWMMSTRT — Indicate the start of a work unit" on page 983	IWM4MSTR. For more information, see "IWM4MSTR — Indicate the start of a work unit" on page 640	Indicate that a work unit is beginning execution.
IWMMSWCH. For more information, see "IWMMSWCH — Switch monitoring environment" on page 990	IWM4MSWC. For more information, see "IWM4MSWC — Monitoring environment switch" on page 648	Indicate that the delay information for a work request may now also reside in another monitoring environment which is not Related to the current environment (Continue) OR that there is no further information for the current work request beyond the current environment (Return).
IWMMUPD. For more information, see "IWMMUPD — Update data for a work unit" on page 995	IWM4MUPD. For more information, see "IWM4MUPD — Update data for a work unit" on page 655	Update data about a work unit which has been started by IWMMSTRT.

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

WLM service name (31-bit only)	**WLM service name (31-bit and 64-bit)**	**Purpose**
IWMMXFER. For more information, see “IWMMXFER — Transfer monitoring environment” on page 1001	IWM4MXFR. For more information, see “IWM4MXFR — Monitoring environment transfer” on page 662	Indicate 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).
IWMMEXTR. For more information, see “IWMMEXTR — Monitoring environment extract” on page 936	IWM4MXTR. For more information, see “IWM4MXTR — Monitoring environment extract service” on page 670	Extract information about the monitoring environment which was previously passed through IWM4MINI/ IWM4MRLT.
IWMQDEL. For more information, see “IWMQDEL — Delete a request from the queue for an execution address space” on page 1009.	IWM4QDE. For more information, see “IWM4QDE — Delete a request from the queue for an execution address space” on page 683.	Delete a request from the queue for an execution address space.
IWMQINS. For more information, see “IWMQINS — Insert a request onto the queue for an execution address space” on page 1016.	IWM4QIN. For more information, see “IWM4QIN — Insert a request onto the queue for an execution address space” on page 696.	Insert a request to the queue for an execution address space.
IWMRPT. For more information, see “IWMRPT — Report on work request completion” on page 1028	IWM4RPT. For more information, see “IWM4RPT — Report response time” on page 708	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.
IWMSLIM. For more information, see “IWMSLIM — Application environment limit service” on page 1040.	IWM4SLI. For more information, see “IWM4SLI — Application environment limit service” on page 720.	Application environment limit service.
IWMSSEL. For more information, see “IWMSSEL — Select a request from a caller's work manager queue” on page 1047.	IWM4SSL. For more information, see “IWM4SSL — Select a request from a caller's work manager queue” on page 739.	Select a request from a caller's work manager queue.
IWMSSEM. For more information, see “IWMSSEM — WLM server select secondary service” on page 1055.	IWM4SSM. For more information, see “IWM4SSM — WLM server select secondary service” on page 746.	WLM server select secondary service.
IWMSTBGN. For more information, see “IWMSTBGN — Begin a request from a caller's work manager queue” on page 1062.	IWM4STBG. For more information, see “IWM4STBG — WLM begin server transaction service” on page 752.	Begin a request from a caller's work manager queue.

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

WLM service name (31-bit only)	WLM service name (31-bit and 64-bit)	Purpose
IWMSTEND. For more information, see "IWMSTEND — End a request from a caller's work manager queue" on page 1070.	IWM4STEN. For more information, see "IWM4STEN — End a request from a caller's work manager queue" on page 760.	End a request from a caller's work manager queue.
IWMTAFF. For more information, see "IWMTAFF — WLM temporal affinity service" on page 1076.	IWM4TAF. For more information, see "IWM4TAF — WLM temporal affinity service" on page 766.	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 33.

Considerations for using the services

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

- What "type" of subsystem work manager do you need?
  - __ – 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 transactions?
  - __ – Do you plan to get reporting information for 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 functions shown in Figure 1. It goes through address space initialization and startup 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 cleanup and termination routines.

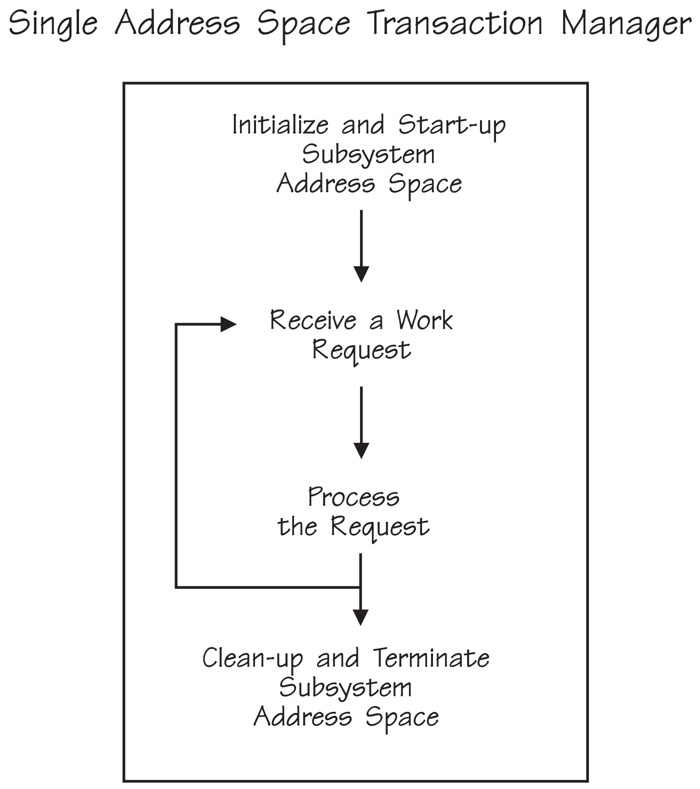

Single Address Space Transaction Manager
Initialize and Start-up Subsystem Address Space
Receive a Work Request
Process the Request
Clean-up and Terminate Subsystem Address Space

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

If you have a single address space transaction manager, consider using the services that are shown in Figure 2 on page 23 for the following functions in workload management:

- Associate work that is coming into the subsystem with a service class
- Have MVS match resources to the work to meet the service class goal
- Provide goal versus actual information for reporting
- Provide response time information and, optionally, processor consumption data for work requests

The services that are shown in Figure 2 on page 23 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 listed objectives.

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 33.

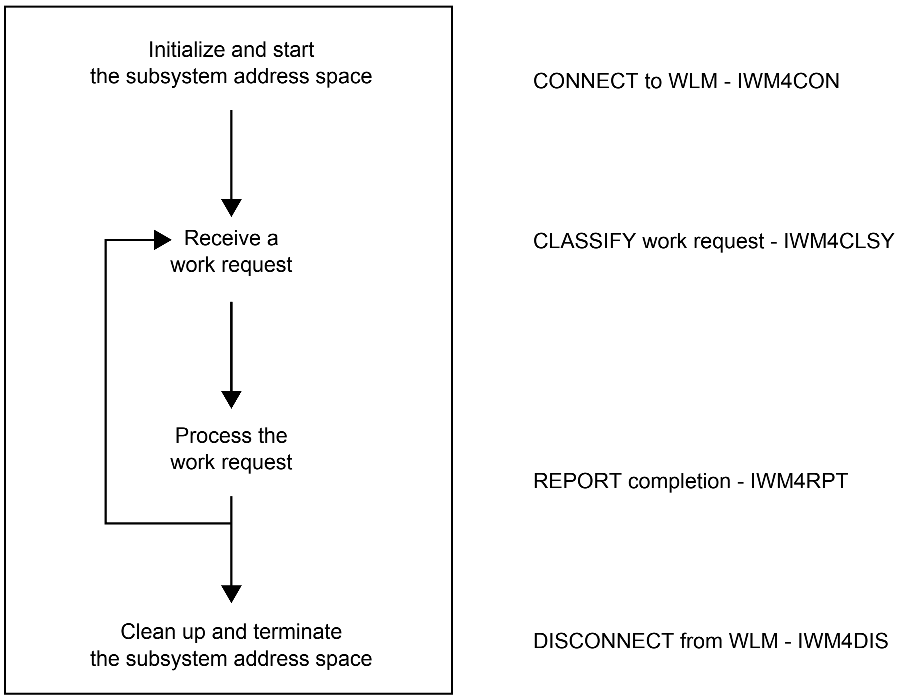

Initialize and start the subsystem address space
CONNECT to WLM - IWM4CON
Receive a work request
CLASSIFY work request - IWM4CLSY
Process the work request
REPORT completion - IWM4RPT
Clean up and terminate the subsystem address space
DISCONNECT from WLM - IWM4DIS

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 that is required by the IWM4CLSY and IWM4RPT services. When the address space receives a work request, it should issue the classify (IWM4CLSY) macro to associate arriving work with a service class. The subsystem work manager can issue IWM4CLSY 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.

IWM4CLSY passes WLM information that identifies the work request. This information includes, among other things, the following:

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

These are called work *qualifiers*. For a complete list of work qualifiers, see the topic about defining classification rules in *z/OS MVS Planning: Workload Management*. IWM4CLSY also supports a product-specific parameter, called the subsystem parameter. If none of the available qualifier types defines 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 IWM4CLSY service, so that a customer knows how to define the *classification rules* that are defined in the workload management ISPF application. Also, because your transaction manager might 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 IWM4RPT. The report service provides the arrival time and, optionally, completion information about the work request and processor consumption data. Only normal completions are included in the response time information, so the information is not skewed by abnormal completions. The transaction manager should issue IWM4RPT only once per work request.

Then, at address space cleanup 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, consider using the services shown in Figure 3 on page 25:

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

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 33

It is important to use the IWM4CHST service together with the IWM4RPT service, otherwise the delay information is not meaningful.

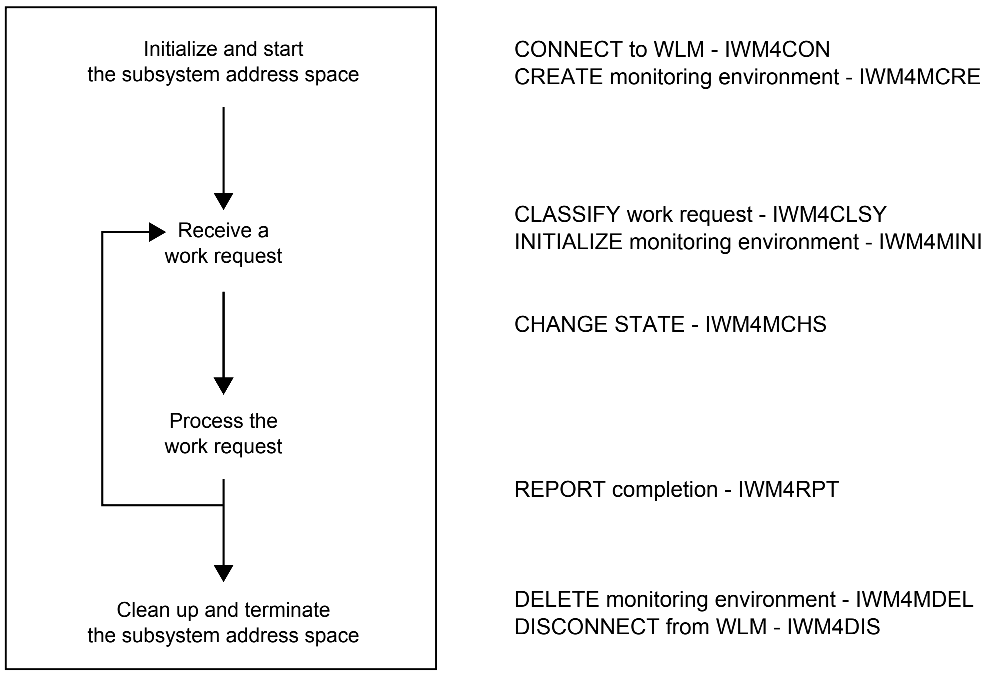

Initialize and start the subsystem address space
Receive a work request
Process the work request
Clean up and terminate the subsystem address space
CONNECT to WLM - IWM4CON
CREATE monitoring environment - IWM4MCRE
CLASSIFY work request - IWM4CLSY
INITIALIZE monitoring environment - IWM4MINI
CHANGE STATE - IWM4MCHS
REPORT completion - IWM4RPT
DELETE monitoring environment - IWM4MDEL
DISCONNECT from WLM - IWM4DIS

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. Because 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 IWM4CLSY 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 IWM4RPT service. The transaction manager should delete all created monitoring environments at address space cleanup and termination, and disconnect from workload management services.

Suggested services for a work manager that calls a data manager

If you have a work manager that calls a data manager on the same MVS image, you can use the combination of services that are described in this topic. If you want to do any of the following, consider using the services that are shown in Figure 4:

- Associate work that is coming into the subsystem with a service class
- Goals versus actual information for reporting
- | Response time information and, optionally, processor consumption data for work requests
- Execution delay information about work
- Track work from a work manager to a data manager

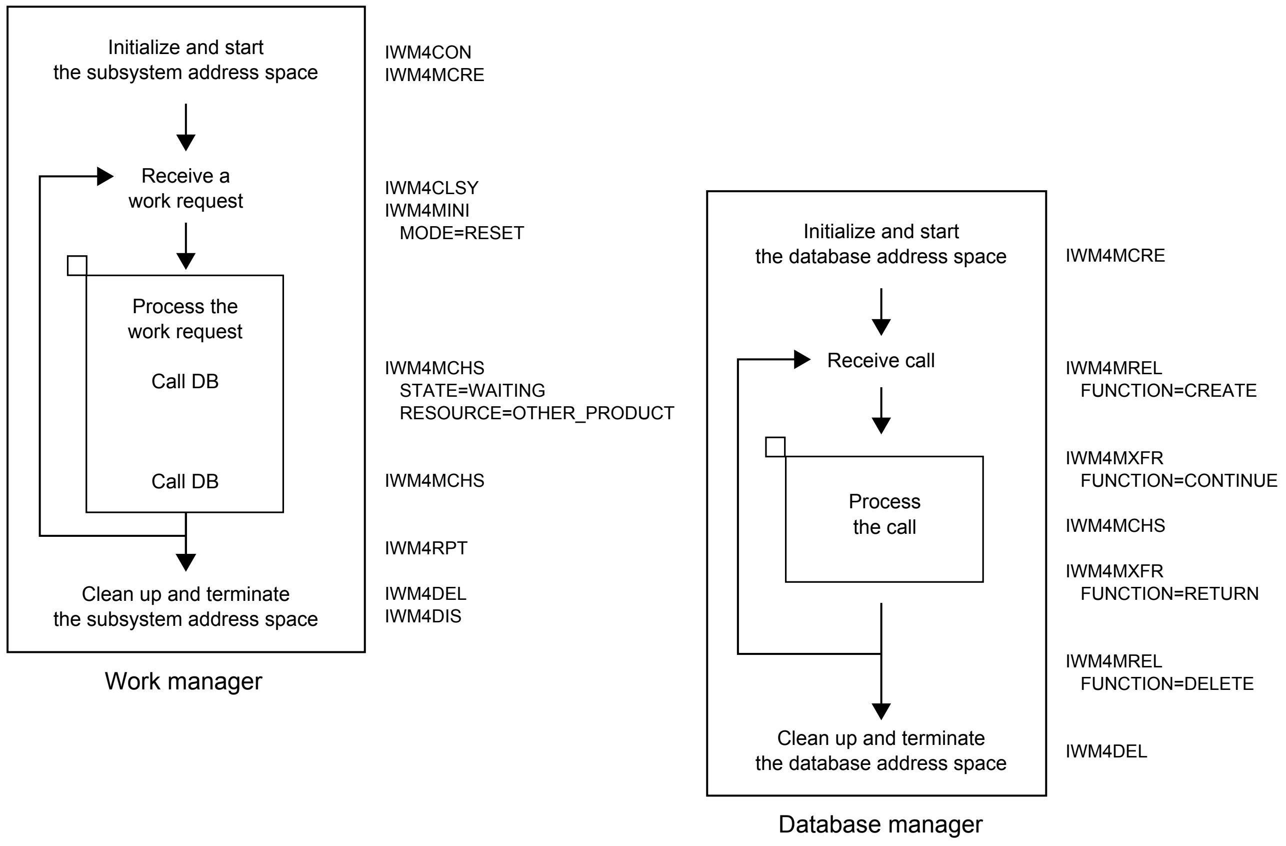


Figure 4. Services for a work manager that uses a database manager.

Because 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 might be running in different tasks, or in the same tasks. The tasks in the figure (one in the work manager, and one in the database manager) might be the same one.

However, the database manager, instead of using IWM4MINI, issues IWM4MRLT (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 that identifies 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 database request. Each such call should begin with a transfer (IWM4MXFR) service with a `FUNCTION=CONTINUE` parameter, and end with a transfer (IWM4MXFR) with a `FUNCTION=RETURN` parameter.

When the work manager issues a call to the database manager, the database manager issues an IWMMXFER. The work manager state can be either active, or waiting throughout the transfer. The IWM4MXFR `FUNCTION=CONTINUE` parameter 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 IWM4MRLT and IWM4MXFR in pairs for each database 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 IWM4MRLT 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 on page 28 shows the sequence of functions in a multiple address space work manager.

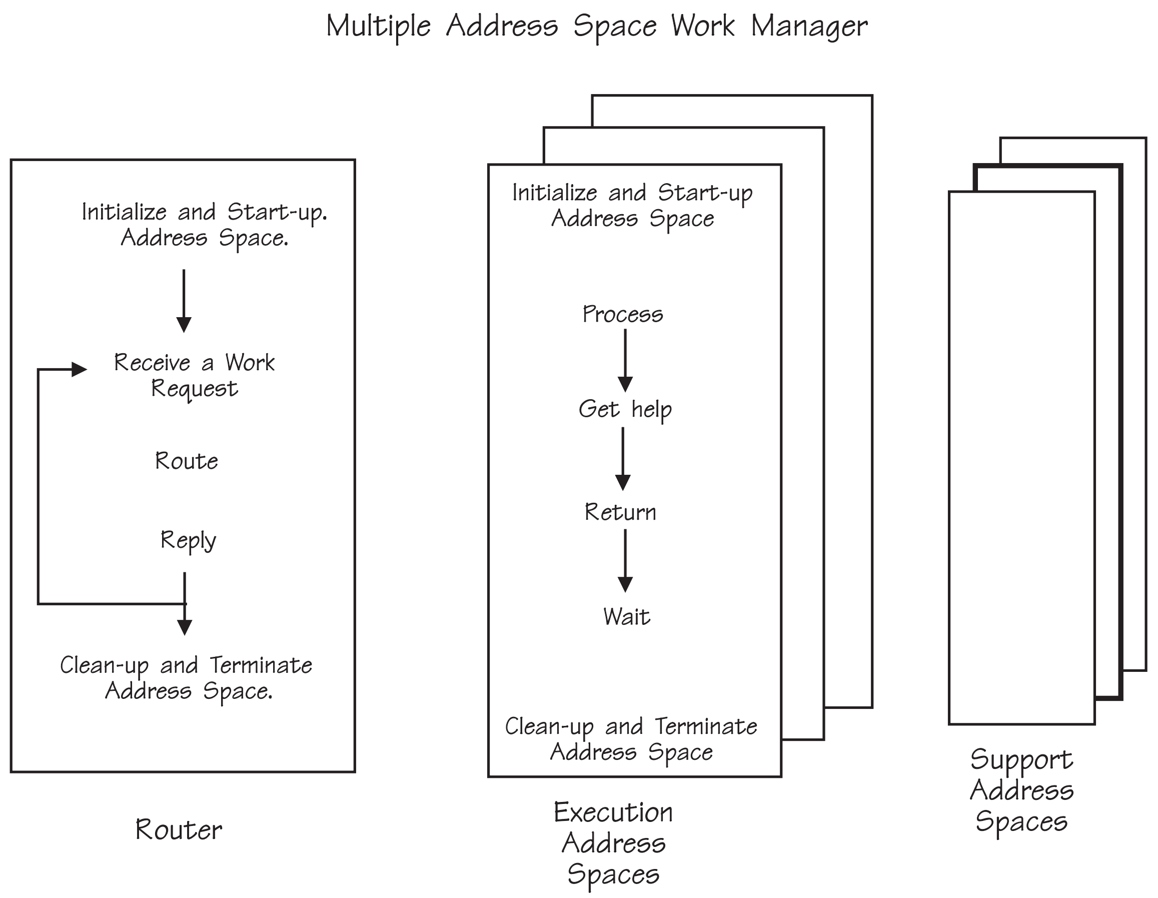

Multiple Address Space Work Manager
Initialize and Start-up. Address Space.
Receive a Work Request
Route
Reply
Clean-up and Terminate Address Space.
Router
Initialize and Start-up Address Space
Process
Get help
Return
Wait
Clean-up and Terminate Address Space
Execution Address Spaces
Support Address Spaces

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

Several groups of services are useful to a multiple address space work manager. They are described in the following topics:

- Chapter 3, “Creating and using enclaves,” on page 33

  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 69

  The queuing 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 29.

  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. Optionally, processor consumption data can be provided. 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), 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 (IWM4MRLT), transfer (IWM4MXFR), and switch (IWM4MSWC) 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 the following:

- 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. IWM4MXFR requires both monitor environments to be on the same image, and knowledge of the parent monitoring environment. IWM4MSWC indicates that 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 IWM4MRLT requires addressability to the parent monitoring environment. IWM4MXFR 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, user ID, and so on) 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 might want to pass the service class token that is returned from the IWM4CLSY service, together with the work request using their own communication methods.

Figure 6 on page 30 shows some suggested services for a multiple address space work manager that take into account the considerations that were discussed previously.

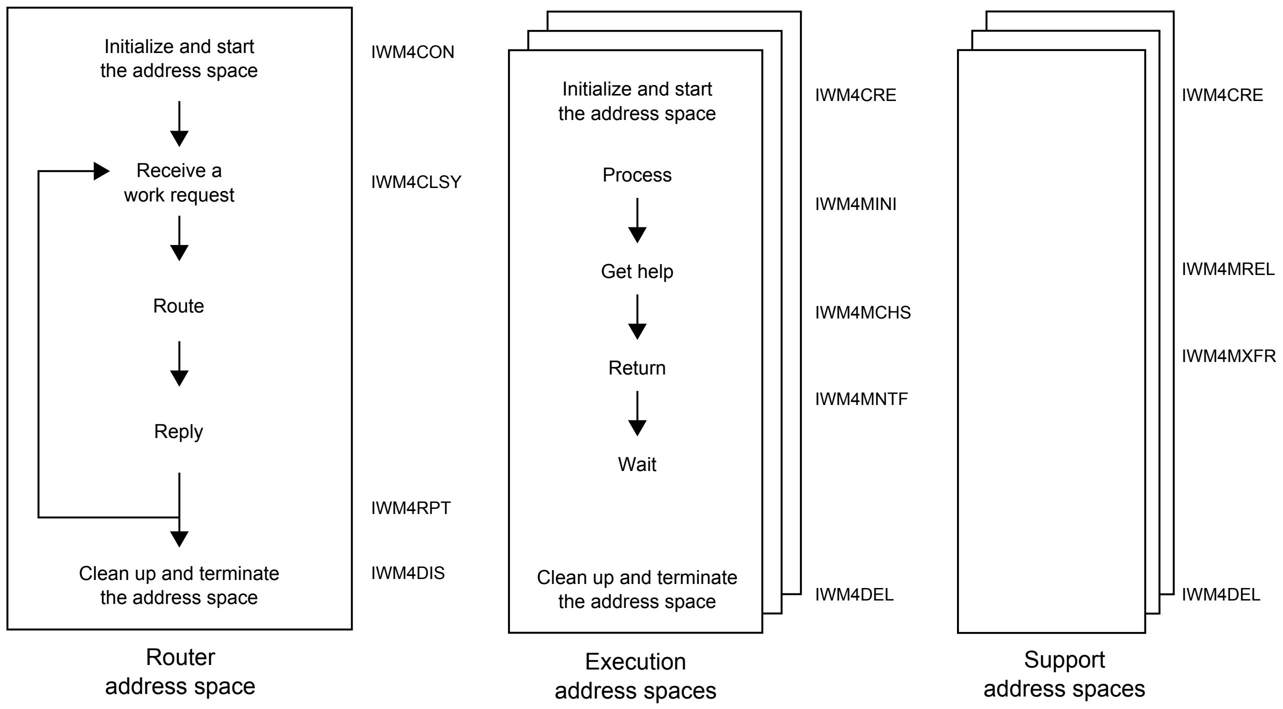


Figure 6. Example of services that monitor work across multiple address spaces.

In the figure, the router might be like a CICS® TOR that routes off the work to an AOR, the execution address space, for processing. There might 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 IWM4CLSY service to associate the arriving work with a service class. Because it also receives the work request back once it has been processed, it also issues the IWM4RPT 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 IWM4MNTF to signify that the execution phase of work request execution is complete and, optionally, to provide processor consumption data. Then, at address space termination, it would issue the IWM4MDEL service to delete the monitoring environments.

The supporting address spaces would also use monitoring environments, so they would issue IWM4MCRE and IWM4MDEL. 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 IWM4MRLT and IWM4MXFR services.

IWM4MRLT and IWM4MXFR require the supporting address space to be on the same MVS image, and that the monitoring token is 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 IWM4CLSY 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 IWM4CLSY 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:

- IWM4CLSY, which returns a service class based on the modified subsystem environment attributes. See “IWM4CLSY — Classify work” on page 464 for a complete description.
- IWM4ECRE, which creates an enclave based on the parameter list from IWM4CLSY. See Chapter 3, “Creating and using enclaves,” on page 33.
- IWM4RPT, which reports on the completion of the work associated with the service class received from IWM4CLSY. See “IWM4RPT — Report response time” on page 708 for a complete description.

For details about the IWMWMCON macro, see “IWMWMCON — WLM modify connect” on page 428.

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 33.

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 IWM4CLSY 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

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. This way the storage-related resources of the server address space can be managed to the performance goal of the enclave.

  For information about using the SYSEVENT ENCASSOC macro, see *z/OS MVS Programming: Authorized Assembler Services Reference SET-WTO*
- 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 IWM4EQRY 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 work-dependent enclave represents a continuation of an existing independent enclave's transaction. It inherits its classification and performance goals from the independent enclave. Service consumed by a work-dependent enclave is treated as if it was consumed by the independent enclave, and can cause the independent enclave (including all associated work-dependent enclaves) to switch into later periods.

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.

Note: It is *not* required to use SYSEVENT ENCASSOC when you run SRBs in enclaves. It is an improvement for storage management but it is not recommended when the address space into which the SRB is scheduled runs other significant work because the association may change the goal management for the target address space.

For more information about the SYSEVENT macro, see *z/OS MVS Programming: Authorized Assembler Services Reference SET-WTO*.

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 deleting 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 uses the IWM4ECRE service to create an enclave. You can define *independent enclaves*, *work-dependent enclaves*, or *dependent enclaves*.

Comparison between independent, dependent, and work-dependent enclaves

You use independent, dependent and work-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 type 30 record of the owning address space and the SMF type 72 record of the enclave's service class or performance group period.

  For an independent enclave, the connect token provided with the CLSFY keyword of macro IWM4ECRE must not be associated with a user key (as specified with the **`CONNTKNKEY`** parameter of IWM4CON, or IWMCONN).

  For examples showing how to use independent enclaves, see “Scheduling an SRB in an independent enclave” on page 39 and “Joining tasks to an independent enclave” on page 41.

- **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. For a dependent enclave that is created with `TYPE=DEPENDENT` specified on IWM4ECRE, the owner will become the home address space at the time the service is invoked. The owner address space of a dependent enclave, resulting from IWM4ECRE with `TYPE=WORKDEPENDENT` specified, will be the creating enclave's (that is, the enclave the TCB/SRB was running in when it called IWM4ECRE) owner. 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 type 30 record of the owning address space and the SMF type 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 43.

- **Work-dependent enclaves**

  Use a work-dependent enclave to extend an existing independent enclave's transaction. A work-dependent enclave inherits its classification and its owner address space from the independent enclave it extends. CPU service consumed by the enclave is accumulated in the SMF type 30 record of the owning address space and the SMF type 72 record of the enclave's service class.

For more specific differences between independent, work-dependent, and dependent enclaves, see Table 13 on page 49.

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, IWMEDREG, 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

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.

Note: The use of multisystem enclaves requires the definition of a coupling facility structure named SYSZWLM_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.

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 type 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 type 30 record.

A multisystem enclave begins as either an independent or dependent enclave on a single system. This enclave is called the *original enclave*. Note that WLM does not allow you to export work-dependent enclaves. 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, IWMEDREG) 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 is 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 44.

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 40 shows the two address spaces.

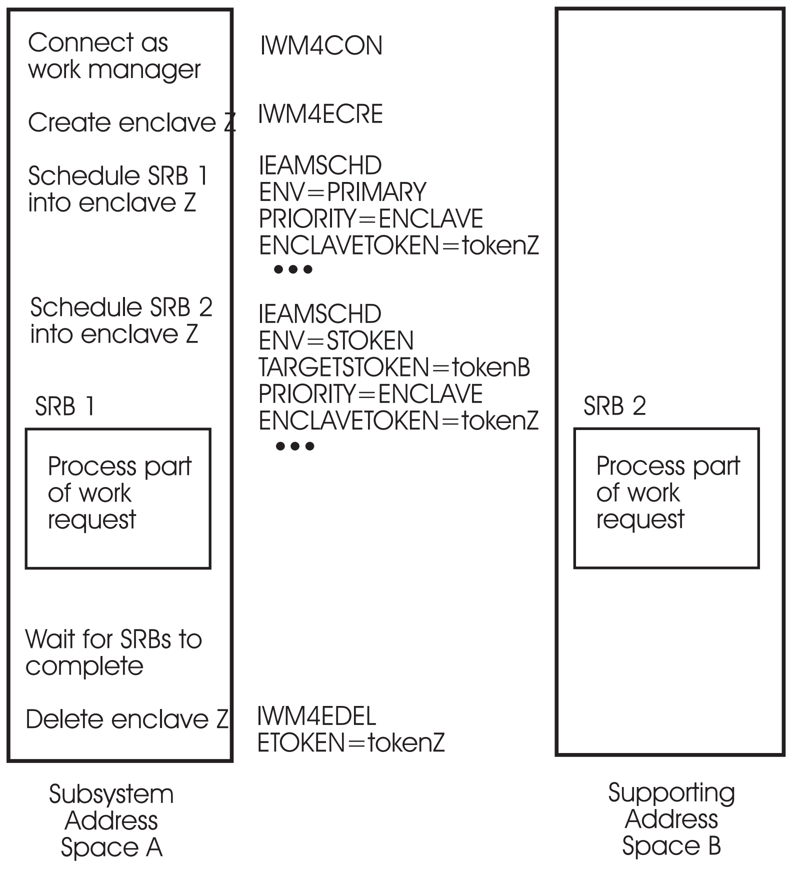


Figure 7. Creating an independent enclave and scheduling an SRB

Figure 7 illustrates the following sequence:

1. 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.

2. 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.

   For more information, refer to “Performance management of address spaces with enclaves” on page 46.

3. 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
```

4. 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
```

5. 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

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 shows how this works.

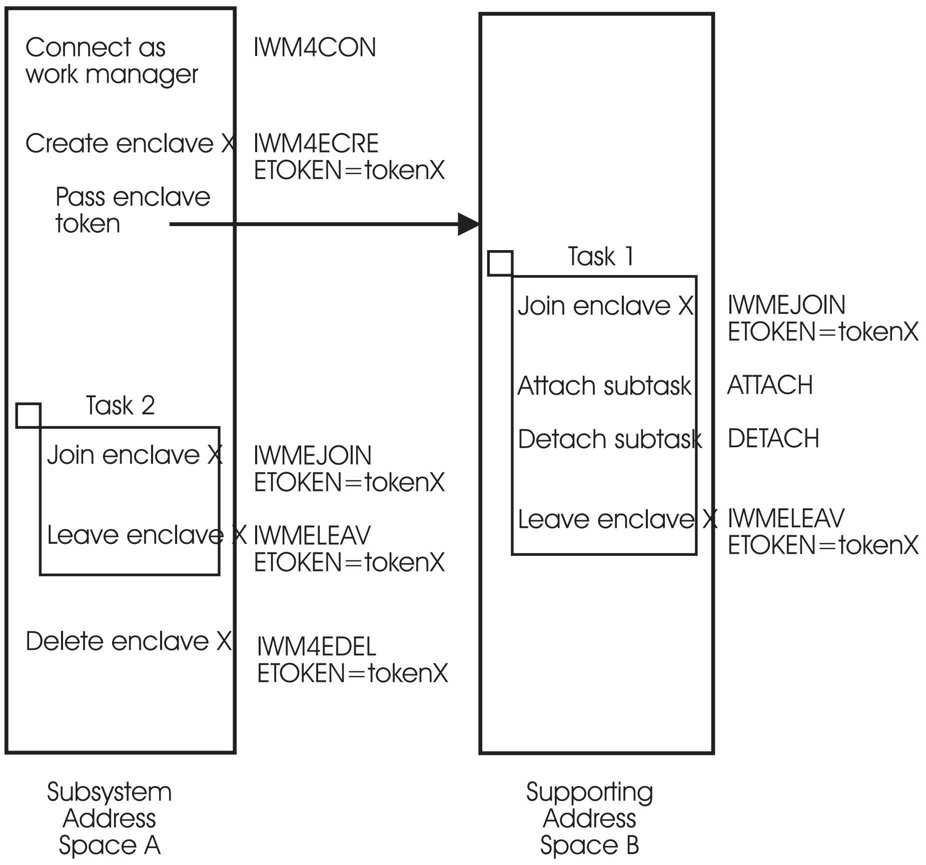


Figure 8. Creating an enclave and joining tasks to it

Figure 8 illustrates the following sequence:

1. 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.

2. 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.

   For further information, refer to “Performance management of address spaces with enclaves” on page 46.

3. 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.

4. 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.

5. 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.

6. 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.

7. 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.

8. 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 shows how this works.

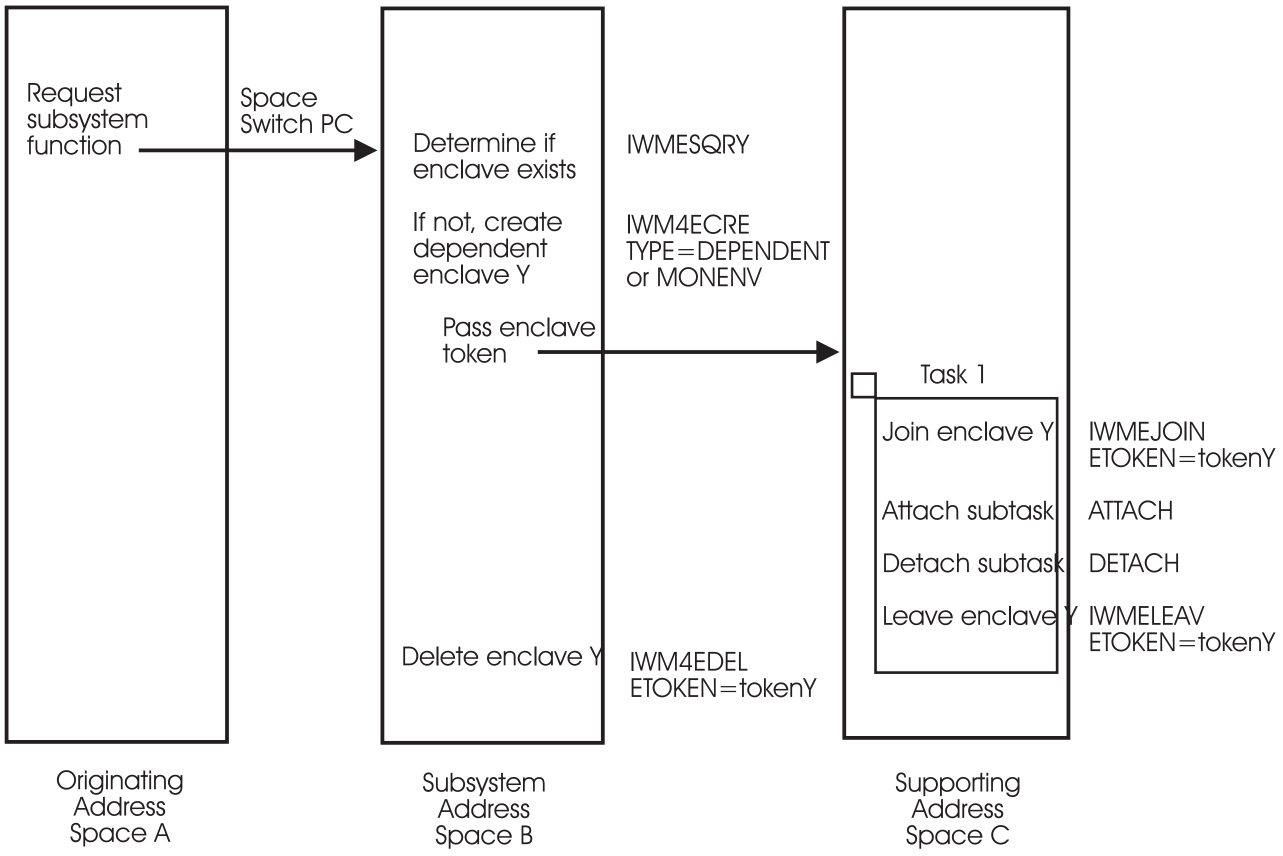

Request subsystem function
Space Switch PC
Determine if enclave exists
IWMESQRY
If not, create dependent enclave Y
IWM4ECRE TYPE=DEPENDENT or MONENV
Pass enclave token
Task 1
Join enclave Y
IWMEJOIN ETOKEN=tokenY
Attach subtask
ATTACH
Detach subtask
DETACH
Leave enclave Y
IWMELEAV ETOKEN=tokenY
Delete enclave Y
IWM4EDEL ETOKEN=tokenY
Originating Address Space A
Subsystem Address Space B
Supporting Address Space C

Figure 9. Using dependent enclaves

Figure 9 illustrates the following sequence:

1. 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.[1] 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.
2. Determine if an enclave exists.

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.

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.

3. 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.

4. 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.

5. 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.

   For further information, see "Performance management of address spaces with enclaves" on page 46.

6. 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.

7. 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.

8. 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 45 shows how this works.

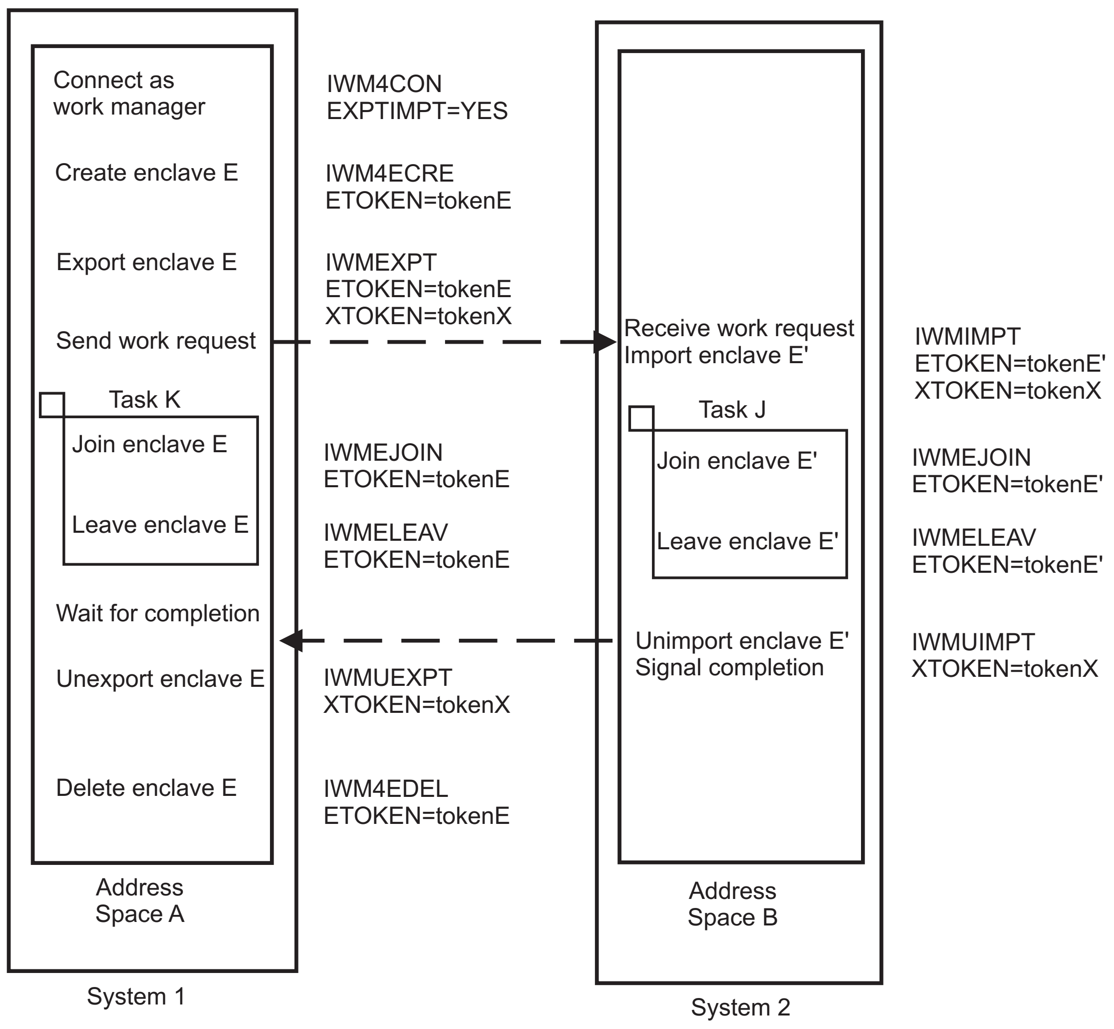


Figure 10. Using a multisystem enclave

Figure 10 illustrates the following sequence:

1. 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.

2. 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.

3. 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.

4. 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.

5. 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'*.

6. 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.

7. 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.

8. 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.

9. 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.

10. 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*.

11. 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

Table 12 on page 47 describes the performance management of address spaces with enclaves in terms of MPL level, paging, dispatching, and I/O priorities.

Address spaces with enclaves (dependent enclaves or independent enclaves) are managed either towards the performance goal of the address space or towards the performance goal of the enclave depending on how the program associates the enclave with the address space. In either case, the enclave's dispatching priority is always managed towards the performance goal of the enclave.

Table 12. Performance management of address spaces with enclaves

Managing performance towards the performance goal of the address space	**Managing performance towards the performance goal of the enclave**
Address spaces are managed towards the performance goal of the **address space** if an enclave SRB was scheduled to run in this address space without the enclave being associated to the address space (the SRB did not issue SYSEVENT ENCASSOC). In this case, the **non-enclave work** is also managed towards the performance goal of the address space. **Note:** An address space must be non-swappable if it has enclave SRBs dispatched and SYSEVENT ENCASSOC has not been issued.	Address spaces are managed towards the performance goal of the **enclave**, if one of the following is true: • At least one task of the address space has joined an enclave by the services IWMEJOIN or IWM4STBG having been issued, or • At least one enclave SRB was scheduled to run in this address space that has issued the SYSEVENT ENCASSOC to associate the enclave with this address space. Note that the performance management of the **non-enclave work** depends on the specification of the IEAOPT parameter **`ManageNonEclaveWork`**: • For `ManageNonEnclaveWork=NO` (and for releases earlier than z/OS V1R12): 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. • For `ManageNonEnclaveWork=YES`: The non-enclave work of an address space is performance managed towards the first service class period of the address space goal. Based on this expanded performance management it is recommended to verify the performance goals for the service class of the address spaces which process enclave work. See Figure 11 on page 48 for an example. **Note:** The non-enclave work of the address space is not performance managed if the service class of the address space is a system-provided service class other than SYSOTHER, even if the `ManageNonEnclaveWork=YES` IEAOPT parameter was specified. Refer to *z/OS MVS Initialization and Tuning Reference* for further information about the OPT parameter.

Figure 11 shows an example of how the non-enclave work of Task F is managed towards the first period of the service class A (SC A) which is the address space goal.

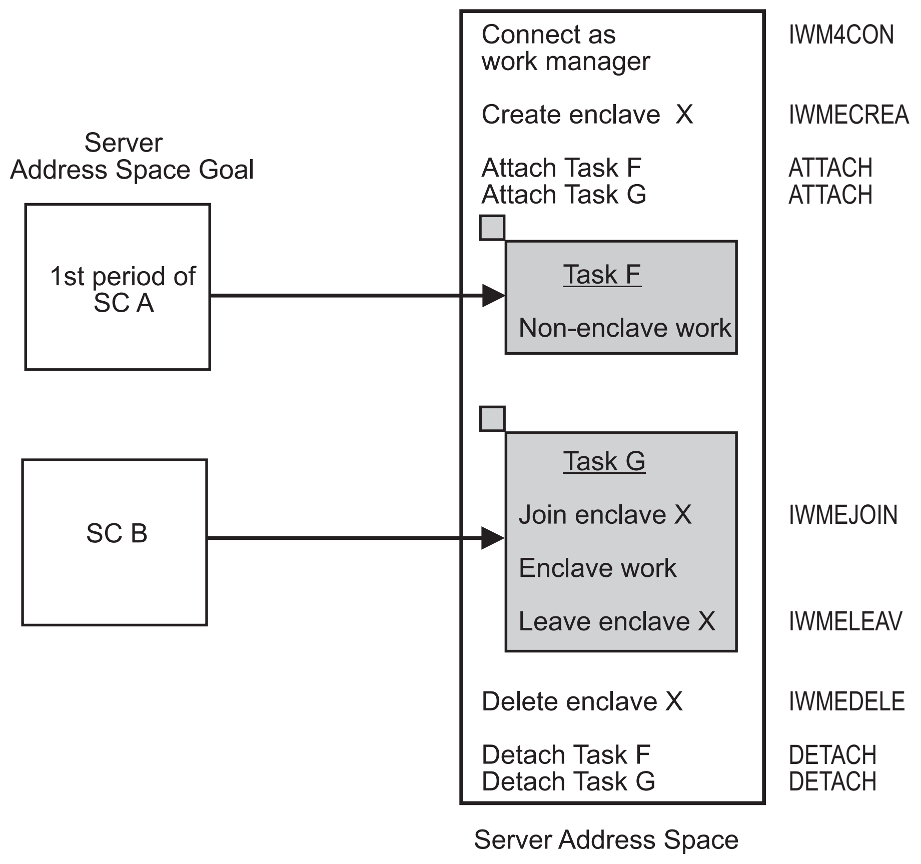


Figure 11. Example of performance management of non-enclave work with IEAOPT parameter `ManageNonEnclaveWork=YES`

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:

1. Task: Join an enclave using IWMEJOIN or IWM4STBG. SRB: Associate enclave with an address space using SYSEVENT ENCASSOC.
2. Obtain resource with ENQ or latch manager.
3. Release resource.
4. 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, work-dependent, 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 type 30 record of the owning address space, and in the SMF type 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 and for work-dependent enclaves, CPU service is included in the SMF type 30 record of the owning address space, and in the SMF type 72 record for the enclave's service class or performance group period. MSO service is not calculated for either kind of enclave.

For dependent, work-dependent and independent enclaves, IOC service is included in the SMF type 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 type 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 type 72 record on each system where the enclave executed.

Because CPU time used by foreign enclaves is included in the owner's SMF type 30 record, it is *not* included in the SMF type 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 type 97 record is written for each SMF global recording interval. This SMF type 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 type 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 type 30 records, and because SMF intervals can vary from system to system, it may not be possible to exactly match the times on SMF type 30 records with those on SMF type 97 records from one global interval to another.

Table 13 compares control characteristics and resource accounting for independent, work-dependent, dependent, and foreign enclaves.

Table 13. Enclave characteristics and resource accounting. (a.s. = address space)

	Independent enclave	Dependent enclave	Foreign enclave	Work-dependent enclave
Dispatchable unit type	SRBs and/or tasks	SRBs and/or tasks	SRBs and/or tasks	SRBs and/or tasks
New transaction?	Yes	No	No	No

Table 13. Enclave characteristics and resource accounting (continued). (a.s. = address space)

	Independent enclave	Dependent enclave	Foreign enclave	Work-dependent enclave
Owner	Home space at the time IWM4ECRE is issued	Depends on the TYPE parameter passed to IWM4ECRE: • If `TYPE=DEPENDENT`, the home a.s. at the time the service was issued. • If `TYPE=WORKDEPENDENT`, the creating (dependent) enclave's home a.s. • If `TYPE=MONENV`, the a.s. associated with the monitoring environment - see Note 1	Owner of the original enclave	Owner a.s. of the creating independent enclave
Server	a.s. where enclave work is dispatched	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.	Same as owning independent enclave's
CPU time	Owner's SMF30Cpt (total) owner's SMF30Enc (independent and work-dependent enclaves only)	Owner's SMF30Cpt (total) owner's SMF30Det (dependent enclave only)	Owner's SMF30MRI (for foreign ind. enclave) owner's SMF30MRD (for foreign dependent enclave)	Owner's SMF30Cpt (total) owner's SMF30Enc (independent and work-dependent only)
CPU service by a.s.	Owner's SMF30Csu (total) owner's SMF30Esu (independent and work-dependent enclaves only)	Owner's SMF30Csu (total)	CPU time/ SMF30MRA/256 * CPU coefficient (CPU coefficient can be obtained from SMF 72 record)	Owner's SMF30Csu (total) owner's SMF30Esu (independent and work-dependent only)
CPU service by period	Enclave's R723Ccpu	Owner's R723Ccpu	Enclave's R723Ccpu	Enclave's R723Ccpu
DASD I/O connect time by a.s. (see Note 3)	Owner's SMF30Eic (independent and work-dependent enclaves only)	Owner's SMF30Aic (dependent enclave + a.s.)	n/a	Owner's SMF30Eic (independent and work-dependent only)
DASD I/O connect time by period (see Note 3)	Enclave's R723Cict	Owner's R723Cict	Enclave's R723Cict	Enclave's R723Cict
DASD I/O counts by a.s.	Owner's SMF30Eis (independent and work-dependent enclaves only)	Owner's SMF30Eis (independent and work-dependent enclaves only)	n/a	Owner's SMF30Eis (independent and work-dependent only)
DASD I/O counts by period	Enclave's R723Circ	Owner's R723Circ	Enclave's R723Circ	Enclave's R723Circ

Table 13. Enclave characteristics and resource accounting (continued). (a.s. = address space)

	Independent enclave	**Dependent enclave**	**Foreign enclave**	**Work-dependent enclave**
Page delay samples, with storage mgt. (see Note 4)	Enclave's R723Cspv	Owner's R723Cspv	Enclave's R723Cspv	Enclave's R723Cspv
Page delay samples, without storage mgt. (see Note 4)	Enclave's R723Caxm	Owner's R723Caxm	Enclave'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	Server's SMF 30 and 72 records
SRB service	n/a	n/a	n/a	n/a
MSO service	n/a	Owner's SMF30Mso, based on owner's frame count	n/a	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 information describes how to classify the work running in enclaves:

- Using independent enclaves
- Using dependent enclaves
- Using work-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.

Using work-dependent enclaves

Work-dependent enclaves inherit their classification from the owning independent enclave and are thus managed to the independent enclave's performance goal.

Querying an enclave's classification information

A caller can use the IWM4EQRY macro to determine the classification information about an enclave. The classification information is that information passed on the IWM4CLSY macro for an independent enclave or inherited from the owning address space for a dependent enclave. For details on IWM4CLSY, see "IWM4CLSY — Classify work" on page 464.

Example

To determine the classification attributes associated with an enclave represented by ETOKEN, first issue IWM4EQRY 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 the following:

```
IWM4EQRY 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 IWM4EQRY again for the enclave classification information:

```
IWM4EQRY ETOKEN=etoken,
         ANSAREA=ansarea,
         ANSLEN=anslen,
         QUERYLEN=querylen
```

Querying a dispatchable unit's enclave status

A caller can use the IWMESQRY 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 IWM4EDEL 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. Work-dependent enclaves are implicitly deleted when the owning independent enclaves get deleted.

For information about the IWM4EDEL macro, see “IWM4EDEL — Delete an enclave” on page 520.

Example

To delete an enclave, specify the following:

```
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 eServer Information Center on the Internet at 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 33.

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, particpation with EWLM is terminated.
- IWM4CLSY

  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 my be started explicitly but stopped implicitly. The possibilities, along with the associated WLM services, are shown in Table 14.

Table 14. Starting and stopping work requests

Start	WLM service	Stop	WLM service
Implicit	IWM4ECRE	Implicit	Work request is stopped by IWM4EDEL
		Explicit	IWMESTOP
Explicit	IWM4ECRE with `ESTRT=EXPLICIT` and subsequent IWMESTRT	Implicit	Work request is stopped by IWM4EDEL
		Explicit	IWMESTOP
	IWM4ECRE with `ESTRT=EXPLICIT_SINGLE` and subsequent IWMESTRT	Implicit	Work request is stopped by IWM4EDEL
		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 "IWM4CON — Connect to workload management" on page 480.

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 "IWM4DIS — Disconnect from workload management" on page 499.

Creating an enclave

The IWM4ECRE service creates an enclave. The optional **ESTRT** parameter indicates whether the work manager implicitly starts an EWLM work request. The **ESTRT** values can be:

- IMPLIED specifies that a work request is started implicitly when the enclave is created.
- EXPLICIT specifies that the work manager explicitly indicates the start of an EWLM work request by invoking the IWMESTRT service.
- 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 EXPLICIT_SINGLE option is specified, the CPU consumption on all EWLM enclave services (IWMEGCOR, IWMESTRT, IWMESTOP, IWMEBLK, IWMEUBLK) will be reduced.

  For details, see the corresponding macro descriptions.
- 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 **WORKREQ_HDL** parameter 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 "IWM4ECRE — Create an enclave" on page 506.

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 IWM4CLSY 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 IWM4CLSY 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 “IWM4CLSY — Classify work” on page 464.

Explicitly starting and stopping work requests

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

- To start: IWMESTRT
- To stop: IWMESTOP

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 “IWMESTRT — Start a work request” on page 248 and “IWMESTOP — Stop a work request” on page 241.

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 and 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 “IWMEBLK — Work request blocked” on page 180 and “IWMEUBLK — Work request no longer blocked” on page 256.

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 15 shows the ARM services that are invoked by the enclave services.

Table 15. 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	

Table 15. WLM Enclave Services and ARM APIs (continued)

Enclave service	Description	ARM service	Parameters
IWMESTOP	Stop a work request	arm_stop_transaction	
IWMEBLK	Indicate that processing of a work request is blocked	arm_block_transaction	
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

For information on how to instrument an application for ARM refer to http://publib.boulder.ibm.com/infocenter/eserver/v1r1/en_US/info/ewlminfo/armguide.pdf.

The topics in this section describe how to instrument, compile, bind and run an ARM-instrumented application on z/OS.

Using the ARM services for instrumenting applications and for managing ARM transactions on z/OS

This information describes the support for ARM transaction management and WLM management.

Supporting transaction management

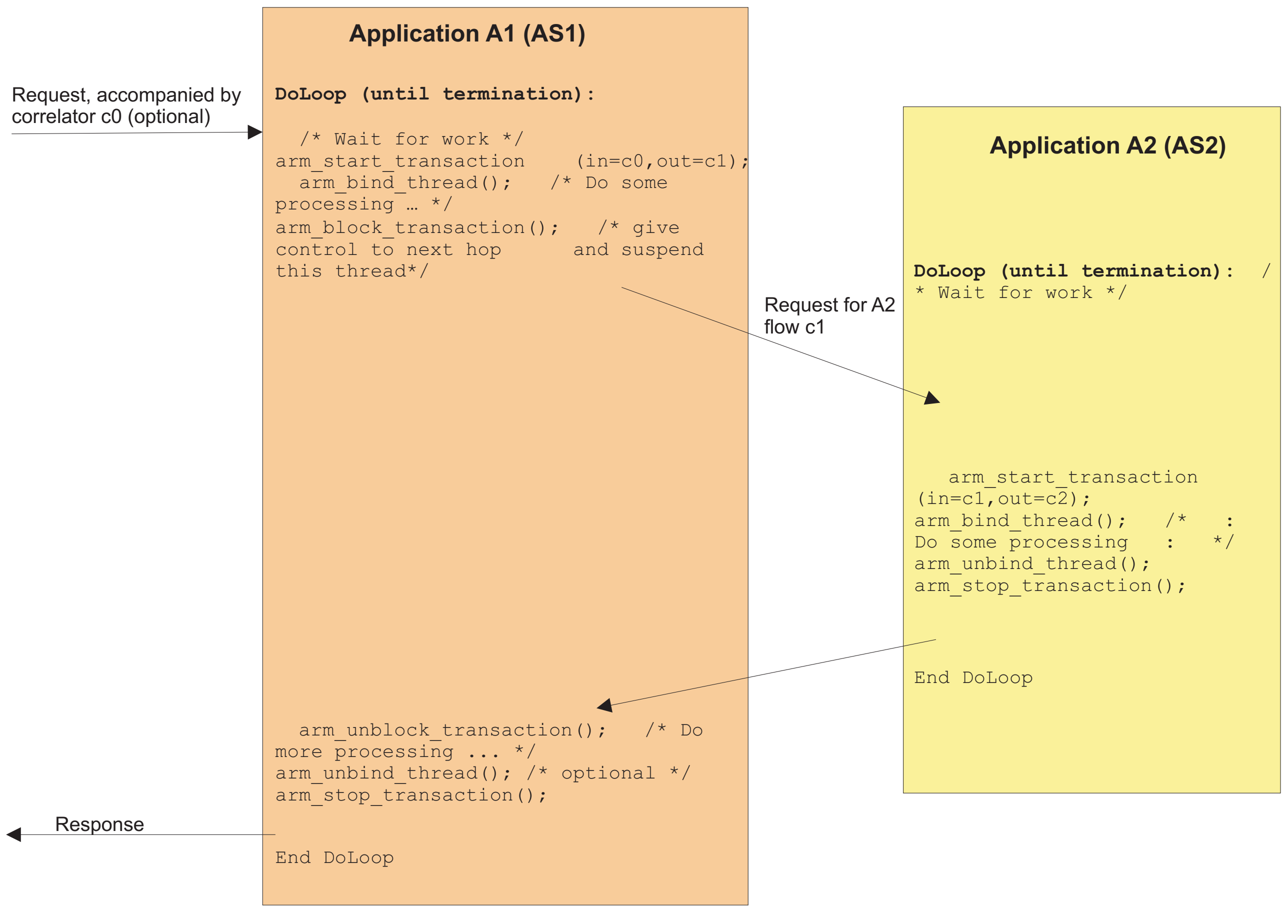
The services **`arm_start_transaction`** and **`arm_stop_transaction`** indicate the beginning and the end of an ARM transaction and support the use of enclaves. Thus, ARM transactions can be managed individually on z/OS.

To allow for transaction management, you must code the **`arm_bind_thread`** and **`arm_unbind_thread`** services in their applications to indicate that the current thread (TCB) is now performing on behalf of a specified transaction. Only when you use the **`arm_bind_thread`** and **`arm_unbind_thread`** services, WLM is able to track the thread. WLM can then assign the necessary resources to that unit of work to help the transaction to reach its goals.

Note: The z/OS implementation of **`arm_bind_thread`** and **`arm_unbind_thread`** conform to the original EWLM R1 semantic specifications. For example, threads are implicitly unbound by **`arm_stop_transaction`** or **`arm_discard_transaction`**.

A sample C-program IWMSARM4 that demonstrates the usage of the services and the new sub-buffers is available in `SYS1.SAMPLIB(IWMSARM4)`. A sample JCL to compile, bind and run this application is provided in `SYS1.SAMPLIB(IWMCARM4)`.

To understand the management of ARM transactions on z/OS it is important to understand that a distributed transaction on a single z/OS system is managed as a whole, even if it spans several address spaces, as shown in Figure 12 on page 62.



```
DoLoop (until termination):

  /* Wait for work */
arm_start_transaction    (in=c0,out=c1);
  arm_bind_thread();   /* Do some
processing … */
arm_block_transaction();   /* give
control to next hop      and suspend
this thread*/

  arm_unblock_transaction();   /* Do
more processing ... */
arm_unbind_thread(); /* optional */
arm_stop_transaction();

End DoLoop
```

```
DoLoop (until termination):  /
* Wait for work */

   arm_start_transaction
(in=c1,out=c2);
arm_bind_thread();   /*   :
Do some processing   :   */
arm_unbind_thread();
arm_stop_transaction();

End DoLoop
```

Figure 12. Typical synchronous application flow between applications on the same z/OS system

When WLM running on a z/OS system first detects of an ARM-instrumented transaction (when the first application A1 performs an **arm_start_transaction** in the example in the figure) a management control block (enclave) is created. Later on, when application A2 performs its **arm_start_transaction** call, the same control block (enclave) is used to track the transaction. Of course, it is necessary to pass the current_correlator c1 from the **arm_start_transaction** call in A1 to application A2 and use that correlator as the parent_correlator in the **arm_start_transaction** call in A2 (and equally so for additional applications A3, A4, and so on). The correlator is used by WLM to find out that A1 and A2 are working on behalf of the same transaction.

Each application in a transaction call sequence needs to indicate that it wants its transactions to be managed towards transaction goals (see next chapter). For example, if A1 has indicated transaction management, but A2 has not, and just because A1 passes a correlator to A2, A2 will not be managed towards transaction goals.

In an A1→A2→A3 scenario, where every application correlates properly but only A1 and A3 have indicated transaction management, A1 and A3 are managed together, but A2 is not.

z/OS ARM sub-buffers to support WLM management

Two z/OS-specific ARM sub-buffers support ARM-instrumented application on z/OS. The first one is for **arm_register_application** and the second is for **arm_start_transaction**.

The sub-buffers are defined in header file `<armewlm.h>` and are located in data set `SYS1.SIEAHDRV.H(ARM4EWLM)`. The sub-buffers have the following names:

- arm_subbuffer_zos_connect_t
- arm_subbuffer_zos_classify_t

These sub-buffers provide WLM with the information that it needs to connect to WLM for management purposes and to specify WLM classification attributes.

Sub-buffer arm_subbuffer_zos_connect: Use the arm_subbuffer_zos_connect sub-buffer to specify the attributes for the connection of the ARM-instrumented address space (application) to WLM. It can be provided as a sub-buffer with the `format_id= ARM_SUBBUFFER_ZOS_CONNECT` (-30400) on an **`arm_register_application`** service call. WLM needs this sub-buffer to manage individual transactions.

Figure 13 shows the layout of the sub-buffer and the parameters which can be passed to the **`arm_register_application`** service, where:

- The **`subsys`** and **`subsysName`** parameters are required
- **`groupName`** is an optional parameter

Specify all character parameters in the character set that the application uses (arm_charset_t). There is no restriction for an application using EBCDIC (IBM-1047) or US-ASCII encoding. For UTF8 and UTF16, the strings are restricted to characters from the US-ASCII (codepoints 0..127) subset.

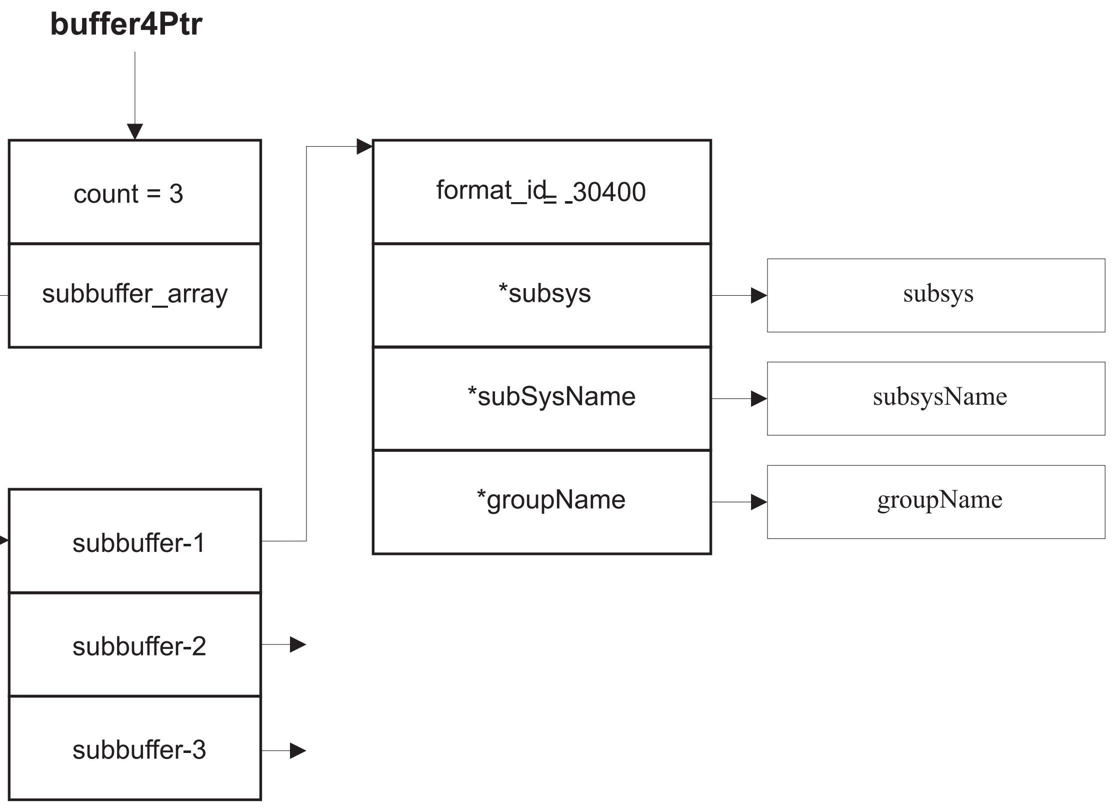
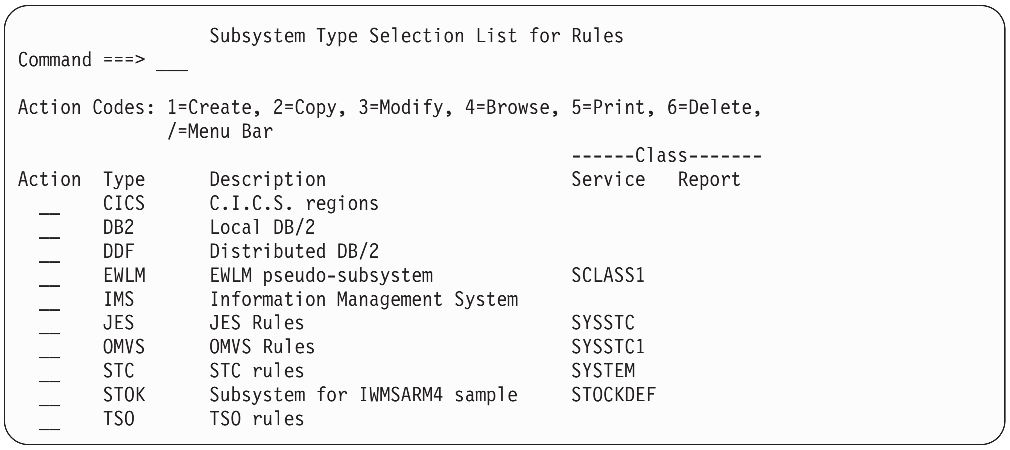


Figure 13. Sub-buffer arm_subbuffer_zos_connect_t)

The **`subsys`** parameter also determines the subsystem type parameter that can be specified in the WLM administrative application. For example, the sample application in `SYS1.SAMPLIB(IWMSARM4)` refers to a subsystem type of `STOK` that could be defined in the WLM classification rules as follows:



```
                  Subsystem Type Selection List for Rules
Command ===> ___

Action Codes: 1=Create, 2=Copy, 3=Modify, 4=Browse, 5=Print, 6=Delete,
              /=Menu Bar
                                                  ------Class-------
Action  Type      Description                     Service   Report
  __    CICS      C.I.C.S. regions
  __    DB2       Local DB/2
  __    DDF       Distributed DB/2
  __    EWLM      EWLM pseudo-subsystem           SCLASS1
  __    IMS       Information Management System
  __    JES       JES Rules                       SYSSTC
  __    OMVS      OMVS Rules                      SYSSTC1
  __    STC       STC rules                       SYSTEM
  __    STOK      Subsystem for IWMSARM4 sample   STOCKDEF
  __    TSO       TSO rules
```

Sub-buffer arm_subbuffer_zos_classify: ARM service **`arm_start_transaction`** can associate a starting transaction to a WLM enclave. The new enclave has to be classified to a WLM service class. This is accomplished by passing the WLM classification attributes in an optional z/OS sub-buffer arm_subbuffer_zos_classify with `format_id=ARM_SUBBUFFER_ZOS_CLASSIFY` (-30401). The values of the classification attributes can then be used in classification rules of the WLM administrative application to associate a specific service class with a transaction. If the sub-buffer is not present but `arm_subbuffer_zos_connect` has been specified on the **`arm_register_application`** call, the transaction will be assigned the default service class for the subsystem.

Figure 14 on page 65 shows the layout of this sub-buffer and the parameters which can be passed to the **`arm_start_transaction`** service. The parameters are the same as those specified on the WLM IWM4CLSY service. Specify all character parameters in the character set that the application uses (arm_charset_t). There is no restriction for an application using EBCDIC (IBM-1047) or US-ASCII encoding. For UTF8 and UTF16 the strings are restricted to characters from US-ASCII (codepoints 0..127) subset.

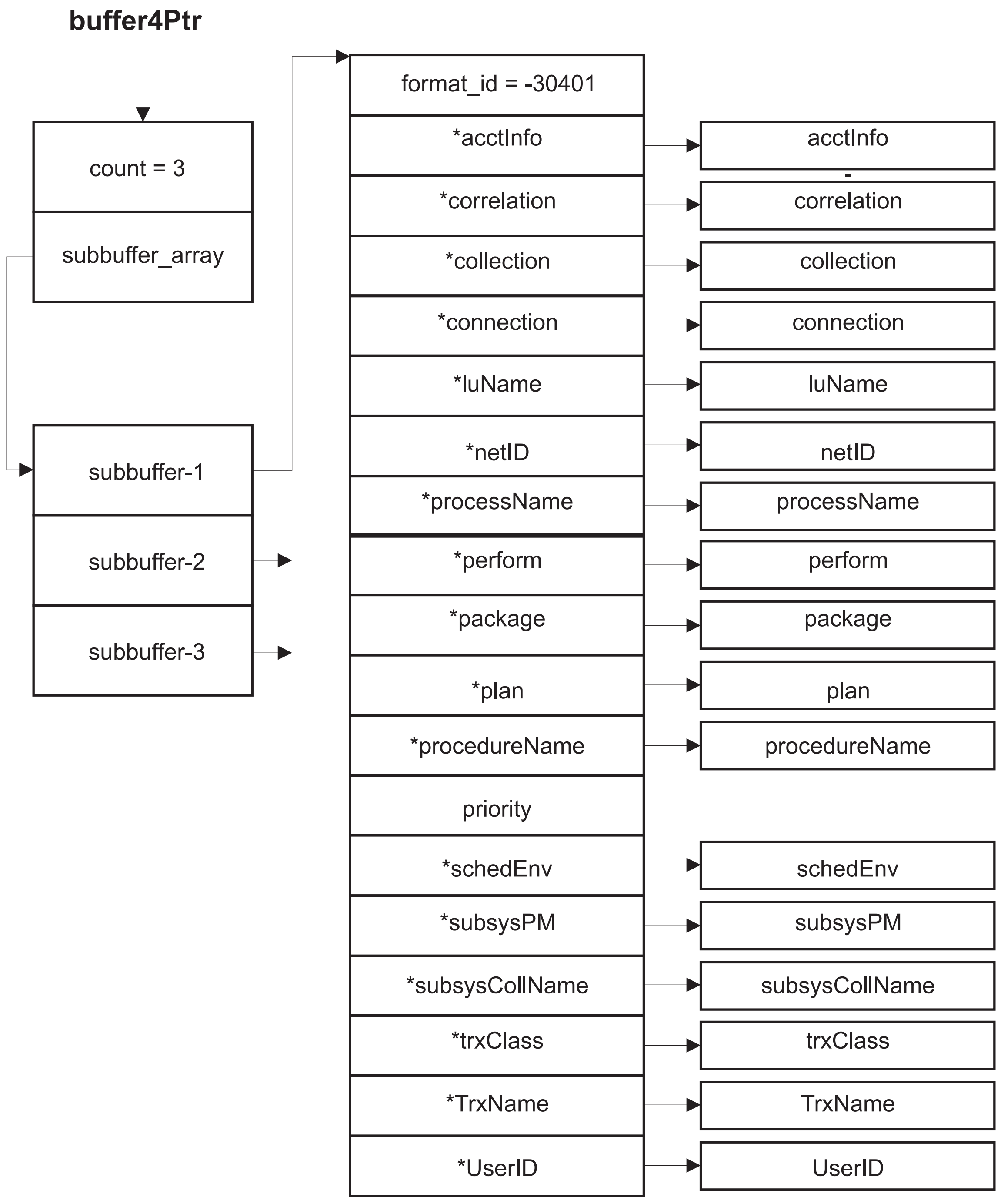


Figure 14. Sub-buffer arm_subbuffer_zos_classify

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 15 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 15. 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 C/C++ compiler on z/OS V1R6 or later.)

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 16 shows examples of compiling under z/OS UNIX System Services in the various environments.

Table 16. Sample commands for compiling applications under z/OS UNIX Systems Services

Environment	Sample command
31-bit XPLINK	`c++ –c –o mymain.o -Wc,"DLL,XPLINK,LANGLVL(EXTENDED)" \` `-I//"'SYS1.SIEAHDRV.+'" mymain.c`
31-bit non-XPLINK	`c++ -c -o mymain.o -Wc,"DLL,LANGLVL(EXTENDED)" \` `-I//"'SYS1.SIEAHDRV.+'" mymain.c`
64-bit	`c++ -c -o mymain.o -Wc,"DLL,LP64,LANGLVL(EXTENDED)" \` `-I//"'SYS1.SIEAHDRV.+'" mymain.c`

Binding an ARM-instrumented application

This information describes using the binder under z/OS UNIX System Services and in MVS batch.

Using the binder in batch

Figure 16 on page 67 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
//          BPARM='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 16. 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** or **c++** command specifying the appropriate side-deck for the ARM DLL and the correct binder options. Table 17 shows sample commands for 31-bit and 64-bit environments.

Table 17. Sample commands for binding applications under z/OS UNIX

Environment	Sample command
31-bit XPLINK	`c++ –o mymain -Wl,"DLL,XPLINK" mymain.o mymem2.o \ /usr/lib/libarm4_3x.x`
31-bit non-XPLINK	`c++ –o mymain -Wl,"DLL" mymain.o mymem2.o \ /usr/lib/libarm4_31.x`
64-bit	`c++ –o mymain -Wl,"LP64,DLL" mymain.o mymem2.o \ /usr/lib/libarm4_64.x`

Running an ARM-instrumented application

An 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 *queuing 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 managers 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 77 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 17 on page 70 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.

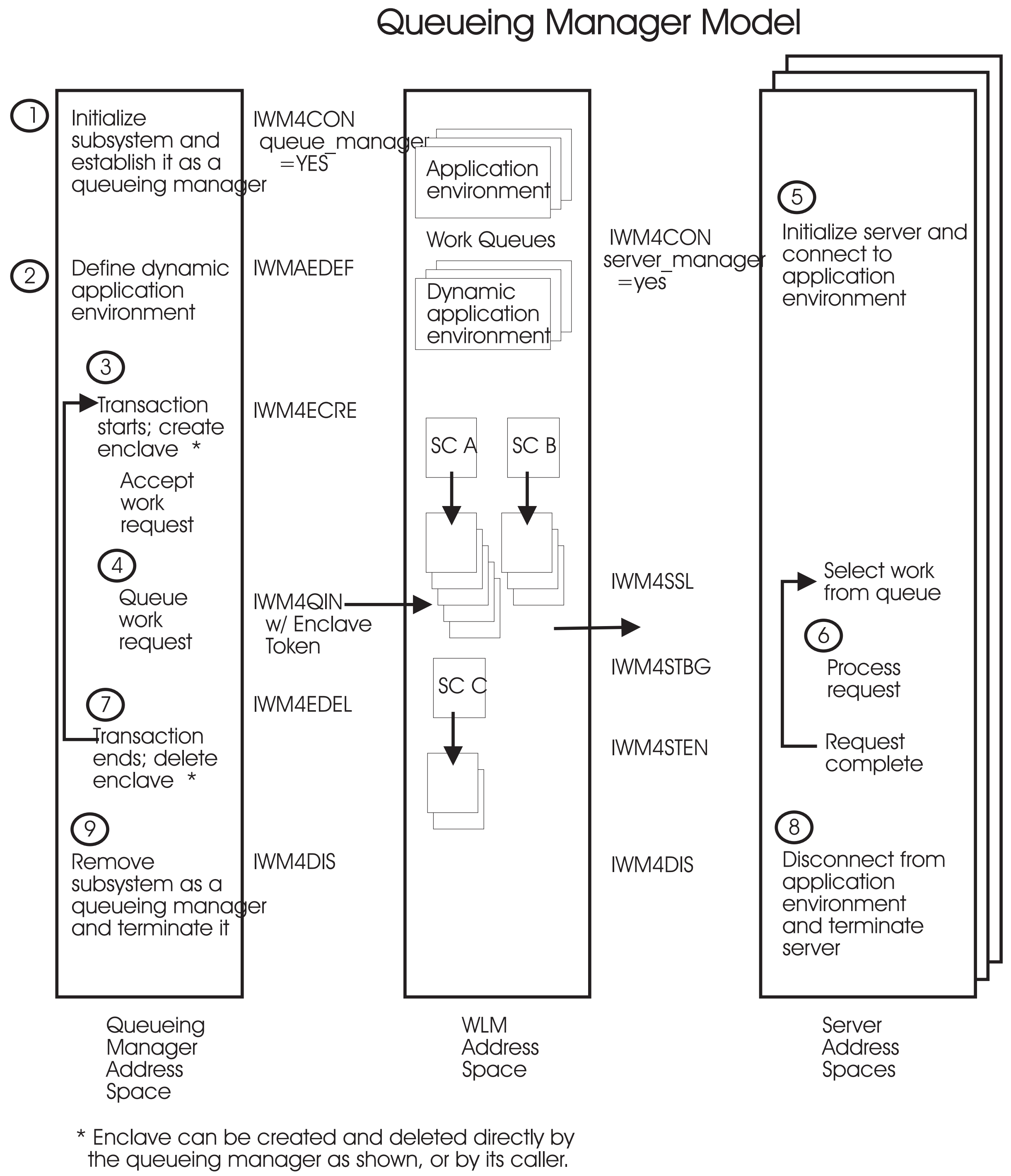

Queueing Manager Model
1
Initialize subsystem and establish it as a queueing manager
IWM4CON queue_manager =YES
Application environment
Work Queues
2
Define dynamic application environment
IWMAEDEF
Dynamic application environment
IWM4CON server_manager =yes
5
Initialize server and connect to application environment
3
Transaction starts; create enclave *
IWM4ECRE
Accept work request
SC A
SC B
4
Queue work request
IWM4QIN w/ Enclave Token
IWM4SSL
Select work from queue
6
Process request
IWM4STBG
SC C
7
Transaction ends; delete enclave *
IWM4EDEL
IWM4STEN
Request complete
8
Disconnect from application environment and terminate server
9
Remove subsystem as a queueing manager and terminate it
IWM4DIS
IWM4DIS
Queueing Manager Address Space
WLM Address Space
Server Address Spaces
* Enclave can be created and deleted directly by the queueing manager as shown, or by its caller.

Figure 17. Services for a queueing manager

Figure 17 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 17.

1. **Establish the 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 77. 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 the 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 queuing manager that invoked this service. For further information, refer to "IWM4AEDF — WLM define dynamic application environments" on page 454.

3. **Create an 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 a work 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 user ID 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*lo.

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 a 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_SPREADMIN=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 75 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 SERVER class 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 queuing manager

Workload management picks up these values from the IWM4CON parameters, **SUBSYS**, **SUBSYSNM**, **APPLENV**, and **NODENM**, respectively. For more information, see "Workload management migration" in *z/OS MVS Planning: Workload Management*.

6. **Process th 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 user ID 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 the 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 the 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=`*applenvname*`,QUIESCE`
     - `VARY WLM, DYNAPPL=`*applenvname*`,QUIESCE` or
     - `VARY WLM,APPLENV=`*applenvname*`,REFRESH`
     - `VARY WLM,DYNAPPL=`*applenvname*`,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=`*applenvname*`,RESUME` or `VARY WLM, DYNAPPL=`*applenvname*`,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 the 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 (**PARALLEL_EU** parameter on IWM4CON).

The IWMSINF interface allows the application to obtain the number of server instances from WLM. Figure 18 shows an example of how the server manager address space can use IWMSINF.

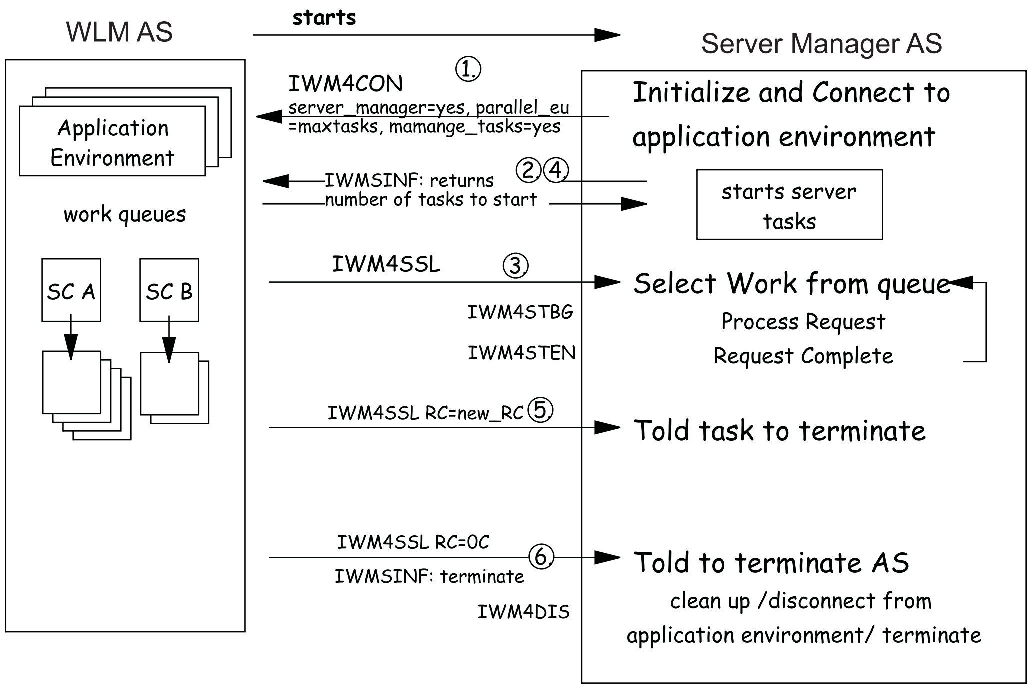


Figure 18. Exploiting IWMSINF

1. To use the IWMSINF interface, the server manager address space connects to WLM with IWM4CON specifying the MANAGE_TASK=YES option. All server address spaces that connect to the same application environment must specify the same setting of **MANAGE_TASK**. The SERVER_MANAGER=YES and SERVER_TYPE=QUEUE options must also be defined.
2. After connecting, the server manager address space must invoke the IWMSINF service 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 again invokes IWMSINF 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 a 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) or connect (IWM4CON) 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. The IWM4TAF interface 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.

Note:

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 IWM4TAF interface. WLM will not terminate a server region if it is marked of having a temporal affinity. This can significantly delay the behavior 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 runtime 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 about 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 *z/OS 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 queueing 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 queueing 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 queueing 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 queueing 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:

0	Not defined
1	Address of the input parameter list
2-13	Not defined
14	Return address
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:

0	Reason code if GR15 return code is non-zero
1-14	Not defined (need not be restored to value on entry)
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

The following example shows how to invoke 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
MYQCXITPLL 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 combine the use of the "find server" function with application environments and enclaves. See "Updating a service definition with application environment information" on page 77 for a discussion of defining policies 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 *z/OS 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 33.

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 19 on page 83 shows the suggested services for a routing manager for managing server address spaces and balancing workload.

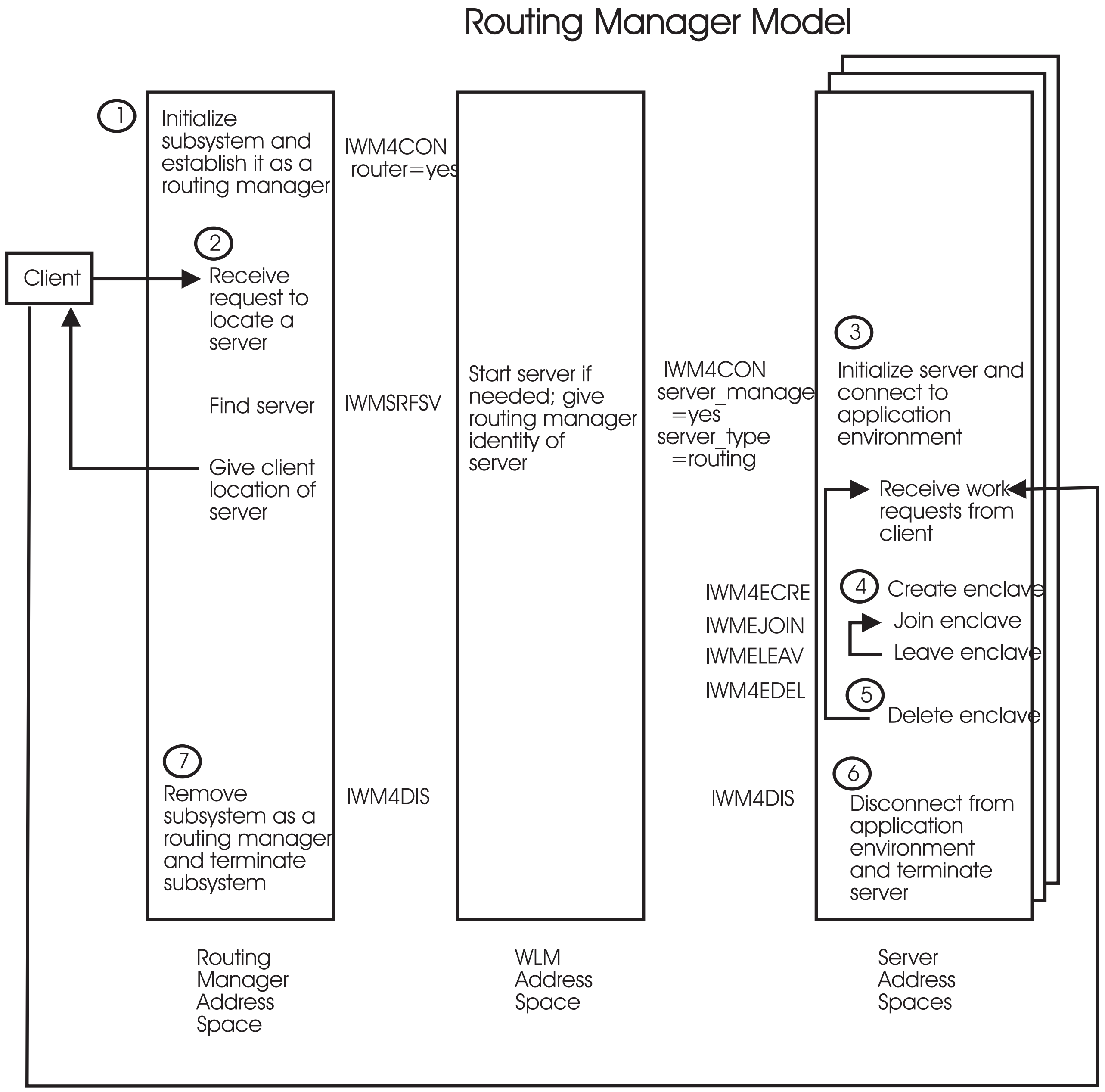

Routing Manager Model
1
Initialize subsystem and establish it as a routing manager
IWM4CON router=yes
Client
2
Receive request to locate a server
Find server
IWMSRFSV
Give client location of server
Start server if needed; give routing manager identity of server
IWM4CON server_manager =yes server_type =routing
3
Initialize server and connect to application environment
Receive work requests from client
IWM4ECRE
IWMEJOIN
IWMELEAV
IWM4EDEL
4
Create enclave
Join enclave
Leave enclave
5
Delete enclave
7
Remove subsystem as a routing manager and terminate subsystem
IWM4DIS
IWM4DIS
6
Disconnect from application environment and terminate server
Routing Manager Address Space
WLM Address Space
Server Address Spaces

Figure 19. Example of routing manager services

Figure 19 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 19 is:

1. **Establish the 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 the 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. See "Using the routing server connect exit" on page 85. 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 *z/OS MVS Planning: Workload Management*.

4. **Create an 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 the enclave.**

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

6. **Terminate the 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:

   - The operator entered a `VARY WLM,QUIESCE` or `VARY WLM,REFRESH` command.
   - A policy was activated from a new service definition which no longer includes the application environment associated with the server.
   - WLM determined 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 the 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 primary 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 IWMSRFSV (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:

0	Not defined
1	Address of the input parameter list
2-13	Not defined
14	Return address
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:

0	Reason code, if GR15 return code is non-zero
1-14	Not defined (need not be restored to value on entry)
15	Return code

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

Example

The following example shows how to invoke 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
+MYSCXITPLL 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 20 on page 90, 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.

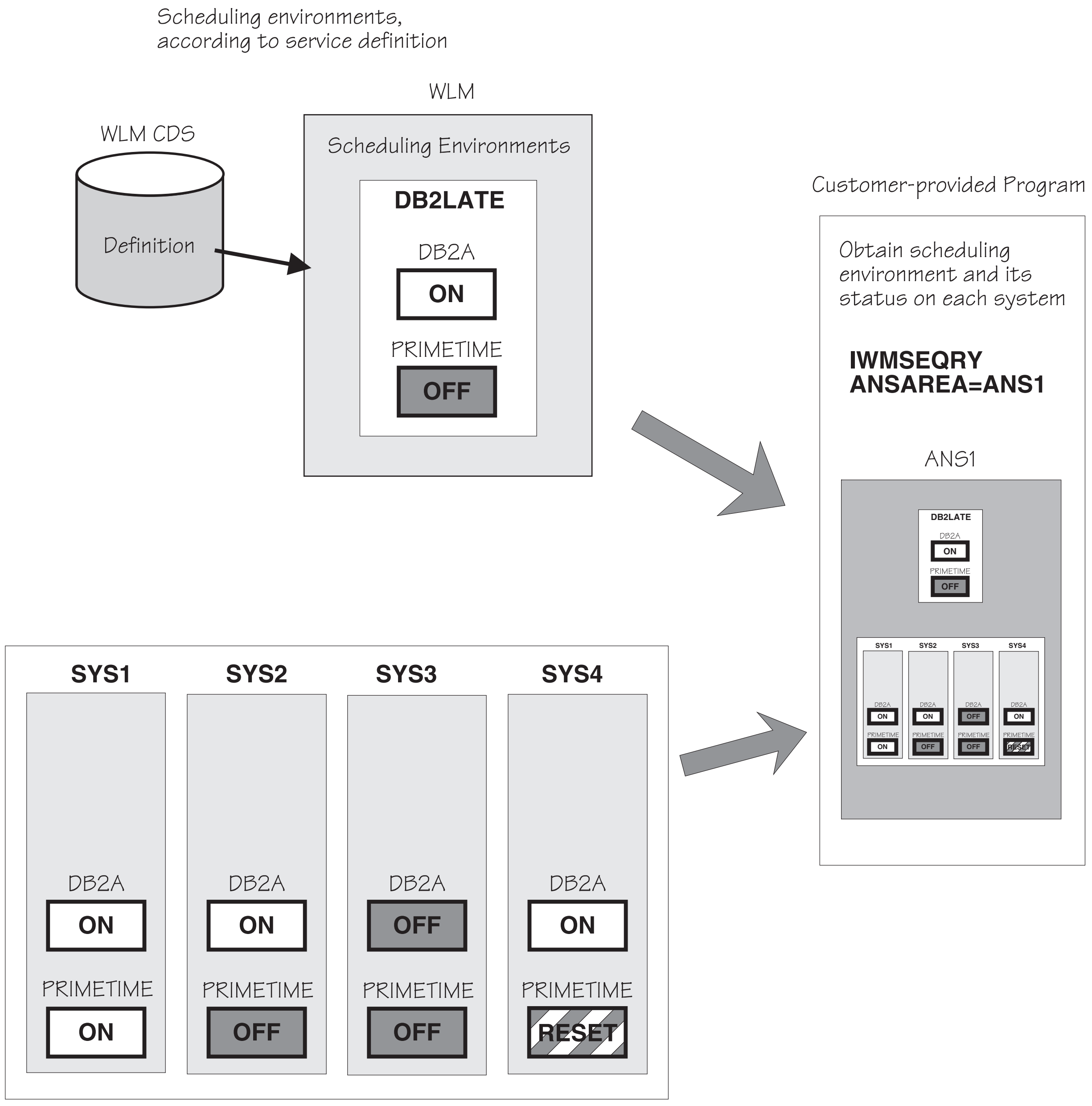


Figure 20. Obtaining scheduling environments

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 21 on page 91, 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 20 on page 90 (DB2A ON and PRIMETIME OFF).

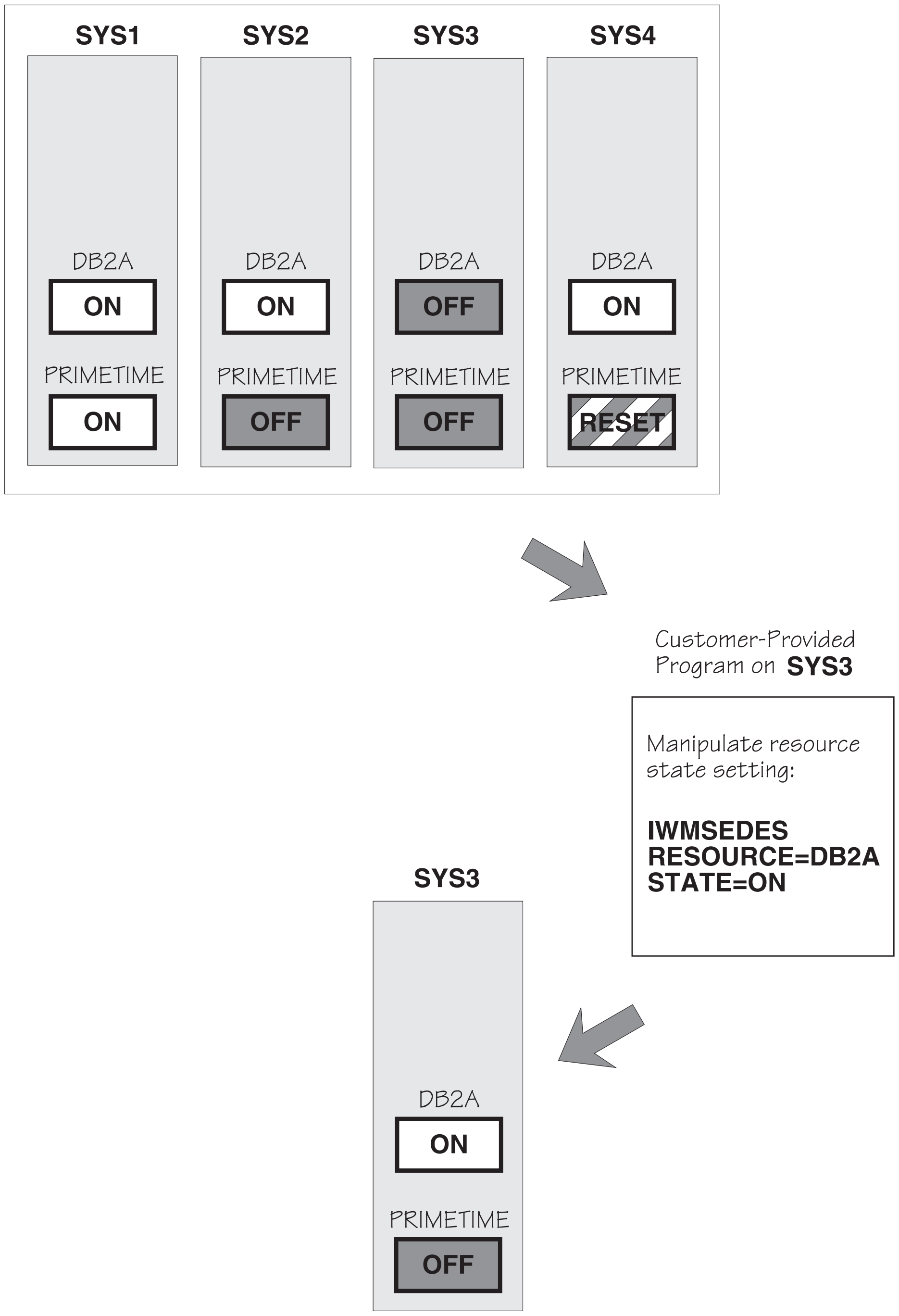


Figure 21. Manipulating resource state settings

A model work flow

Figure 22 on page 92 shows how a multisystem work scheduler can use the IWMSEVAL and IWMSEDES services to implement support of scheduling environments.

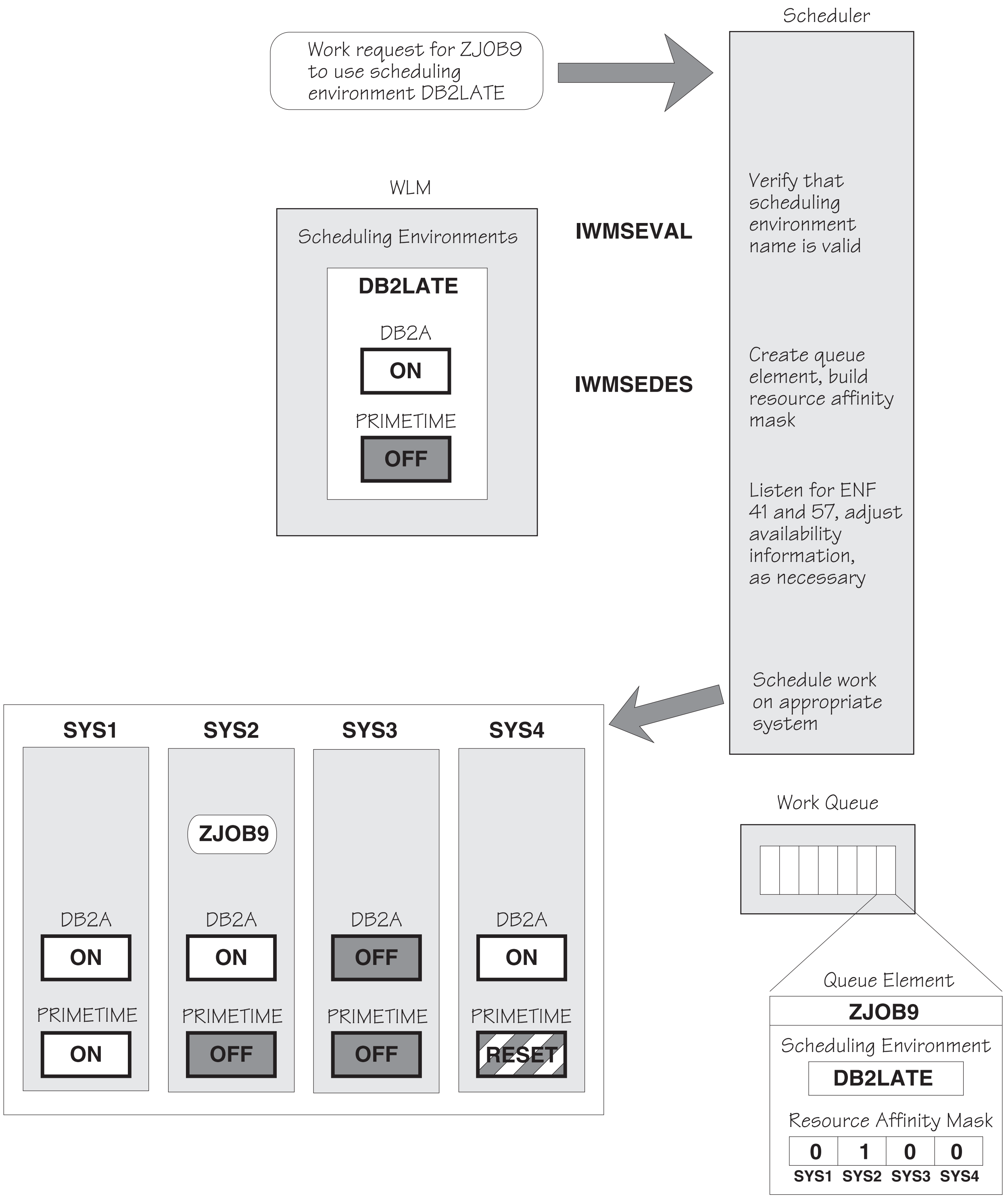

Work request for ZJOB9 to use scheduling environment DB2LATE
Scheduler
WLM
Scheduling Environments
DB2LATE
DB2A
ON
PRIMETIME
OFF
IWMSEVAL
IWMSEDES
Verify that scheduling environment name is valid
Create queue element, build resource affinity mask
Listen for ENF 41 and 57, adjust availability information, as necessary
Schedule work on appropriate system
SYS1
SYS2
SYS3
SYS4
ZJOB9
DB2A
ON
ON
OFF
ON
PRIMETIME
ON
OFF
OFF
RESET
Work Queue
Queue Element
ZJOB9
Scheduling Environment
DB2LATE
Resource Affinity Mask
0 1 0 0
SYS1 SYS2 SYS3 SYS4

Figure 22. Scheduling environment flow

In Figure 22, 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 22 on page 92, 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 event codes, 41 and 57, which signal these changes:

Code	**Event description**
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. Either it was previously not available and is now available, 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 22 on page 92, 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.

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. Because 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 "IWMSRSRG — Register a server for sysplex routing" on page 396.

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* 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, consider the following list of servers and weights:

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

The client should route the first three requests to NETID1.LUNAMEA, the next eight 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 "IWMSRSRS — Sysplex routing information" on page 405.

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 IWMSRDRS macro, see "IWMSRDRS — Deregister a server for sysplex routing" on page 382.

Example of using the sysplex routing services

Figure 23 shows an example of the sysplex routing services.

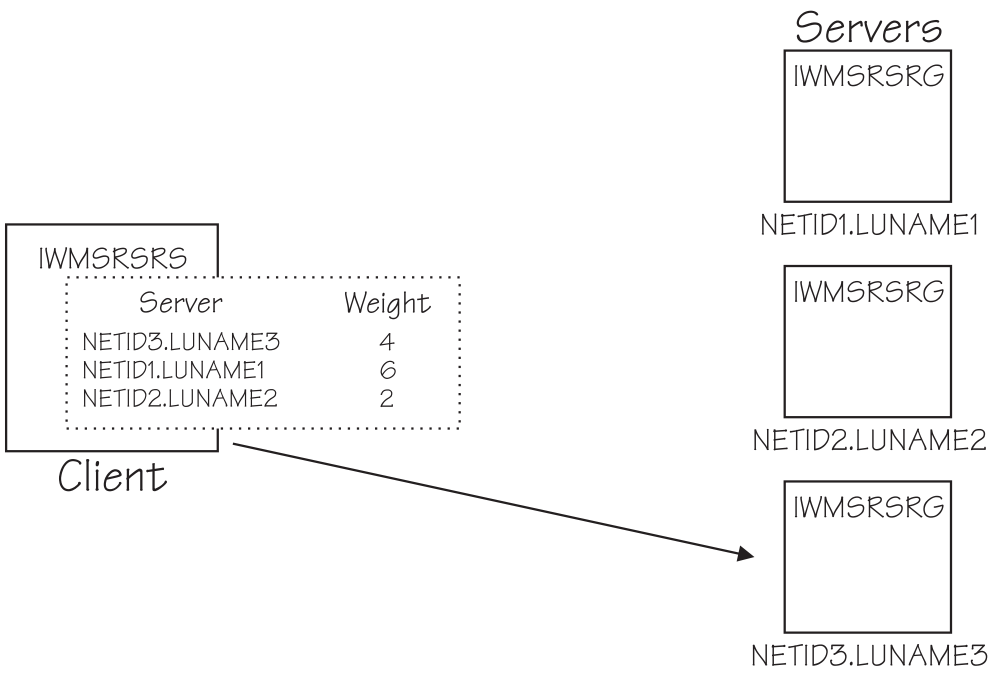


Figure 23. Example of using sysplex routing services

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-eight 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 CPU consumption table (see Table 18) 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. The levels are:

Level	Usage
0	Service consumed by the system
1-5	Externally defined importance levels
6	Discretionary work
7	Unused service (equivalent number of service units)

Each row contains the following fields:

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

WLM calculates weights as:

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. WLM calculates a system weight for each system in the sysplex using the SU value in the selected row in the table, as:

```
                      SUs for this system at selected level  ×  64
System weight   =   -----------------------------------------------
                      total SUs for all systems at selected level
```

3. WLM calculates a server weight, as:

```
                            System weight
Server weight   =   ------------------------------
                     number of servers on system
```

Example

For this example, consider the consumption table shown in Table 18.

Table 18. CPU consumption table

	System A		System B		System C	
Level	**SUs**	**Percent**	**SUs**	**Percent**	**SUs**	**Percent**
0	2000	100	2000	100	2000	100
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

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, the 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, 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 weight	Server weight
System A weight = `200 × 64 / 640 =` **`20`**	Server weight = `20 / 2 =` **`10`**
System B weight = `400 × 64 / 640 =` **`40`**	Server weight = `40 / 2 =` **`20`**
System C weight = `40 × 64 / 640 =` **`4`**	Server weight = `4 / 2 =` **`2`**

The IWMSRSRS service supports the function code `SPECIFIC`, which 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 previously described is additionally multiplied by a *Performance Index (PI) factor* and a *queue time factor*. The value of these two factors is calculated as:

PI factor
: is 1 if the server has a PI <= 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.

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.

A *health factor* is also multiplied into the weight calculation. The health factor is set for the server address space either by the IWMSRSRG registration service or by the IWM4HLTH health service. For more information about these services, see to "IWMSRSRG — Register a server for sysplex routing" on page 396 and "IWM4HLTH — Setting server health indicator" on page 535.

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, event notification facility (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

Because all performance data is continually collected until a significant change takes place in workload management, the performance monitor must know when the collection is reset. Workload management resets 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: Because 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 19. 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.

ENF event code 41

Table 19 shows the ENF event code and its qualifiers. The table also outlines what a performance monitor could do when the ENF code is heard.

Table 19. Using ENF event code 41 to guide data collection for policy changes

Event	Signal and qualifiers	Expected action
`VARY WLM, POLICY`	WLMENF11 WLMENF12 WLMENF13	**WLMENF11** 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. **WLMENF12** 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. **WLMENF13** Indicates the new policy was *not* 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.

Table 19. Using ENF event code 41 to guide data collection for policy changes (continued)

Event	Signal and qualifiers	Expected action
System failure	WLMENF31 WLMENF32 WLMENF33	**WLMENF31** Indicates workload activity reporting failed, recovery has begun. Do not issue IWMRCOLL until recovery is complete. **WLMENF32** Indicates workload activity reporting recovery was successful. Start a new reporting interval. **WLMENF33** Indicates workload activity reporting recovery was unsuccessful. Do not issue IWMRCOLL as no data is returned until the next IPL.

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 20 shows how you can use the workload reporting services together with IWMRCOLL to get workload activity information on a single system. "Using the information in IWMWRCAA" explains how to use the information returned by IWMRCOLL.

Table 20. 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 obtain the following kinds of information:

- Resource consumption information
- Response time and distribution information, and special reporting data
- General delay information
- Subsystem work manager delay state information

The information that is provided is mapped by the IWMWRCAA data area. Information is returned on a service class and report class basis. "Using the subsystem work manager delay state information" 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 might 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 that are 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 that is 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 that are encountered by service classes that represent transactions. The subsystem work manager execution delays are in the *subsystem work manager state samples* section of IWMWRCAA. The state samples are available only for subsystem work managers that use the execution delay monitoring services.

To assist in determining 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 by using the information that is 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.

| In addition, special reporting data is provided for service and report
| classes.

Using the subsystem work manager delay state information

The delay information in IWMWRCAA 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. IWMWRCAA 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 life span 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 24 on page 106 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.

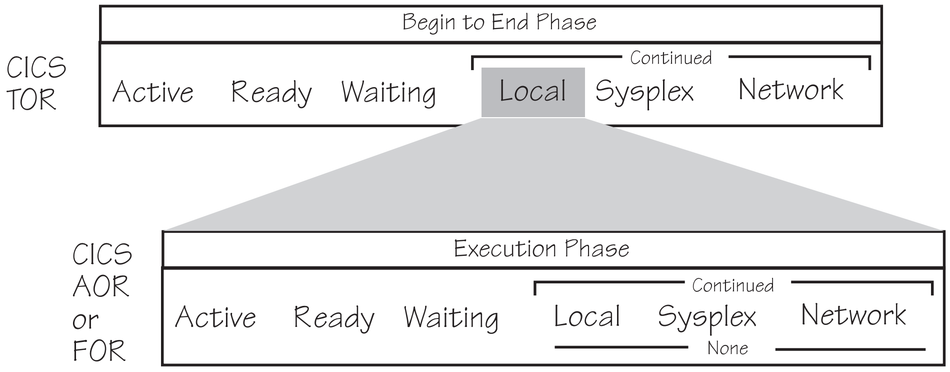


Figure 24. Using states for presenting CICS delay information

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 25 on page 107 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.

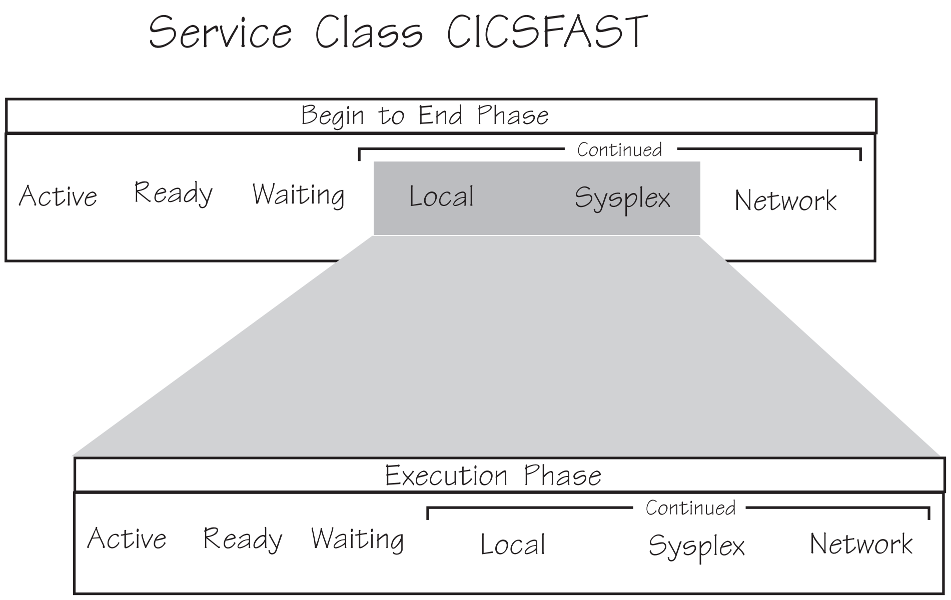


Figure 25. Combining state information for a sysplex view

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 26 on page 108, 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.

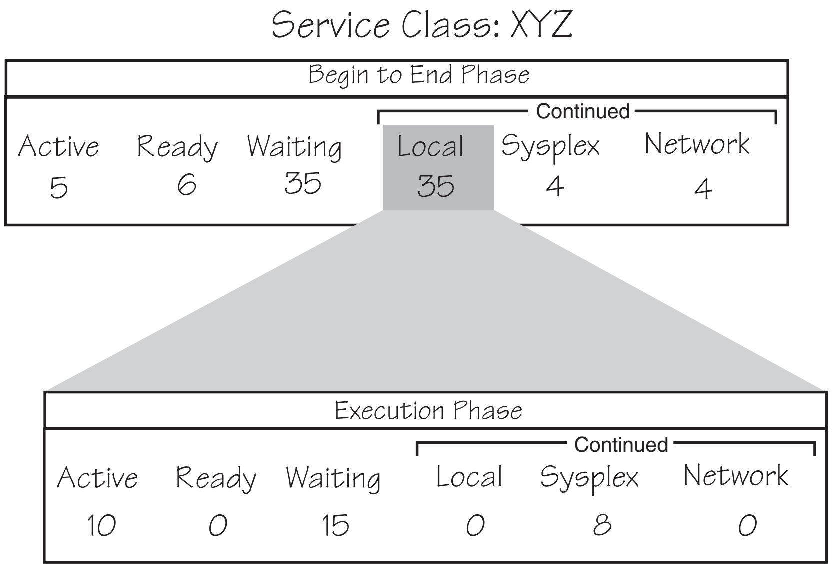


Figure 26. Combining state information for a service class

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 27, 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.

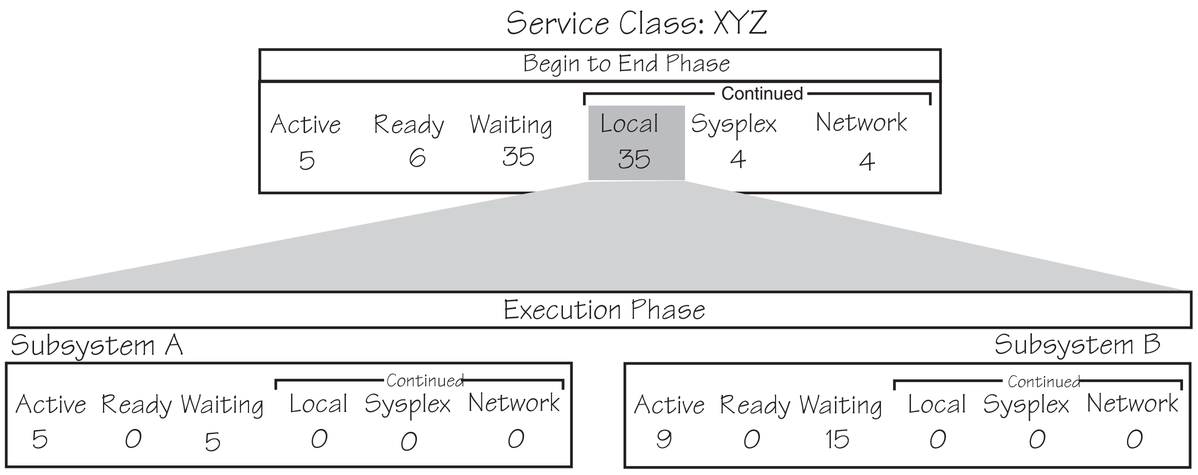


Figure 27. Combining state information across subsystems

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). One bucket always maps exactly to the specified goal, with a value of 100%.

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

In Table 21, each of the 14 buckets represents a percentage of the specified 1-second goal. For instance, bucket 2 represents all transactions that completed in 50% to 60% of the goal, or 500 - 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 (100% of goal, or 1 second). This bucket captures all those transactions that complete in 900-1000 milliseconds.

The two end buckets (buckets 1 and 14) 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.

Table 21. Self-defining response time distributions for service class XYZ

Response time distributions for service class XYZ Goal: 90% in 1 second														
Bucket	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Percent of goal	50	60	70	80	90	100	110	120	130	140	150	200	400	FFF
Transaction count	1	0	1	10	48	12	26	13	7	2	2	0	0	1

In addition to the response times, special reporting data is available in this section for service and report classes. With classification rules, you can associate your work with a special reporting attribute. This attribute can be used to report on subsets of transactions that are running in a service or report class. For more information, see the topic on defining special reporting options for workload reporting in *z/OS MVS Planning: Workload Management*. Special reporting data for a service or report class contains service consumption on general-purpose processors and offload engines for each value of the reporting attribute. In contrast to other data in the IWMWRCAA answer area, these values might not be cumulative. Due to reaccounting of transaction processor usage that arises from the WLM execution delay monitoring services, values might shrink from one invocation of IWMRCOLL to the other.

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 V1R2, WLM only returned a report class summary.

Since z/OS V1R2 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 28).

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

Figure 28 illustrates the concept of the mixed-class-indication timestamp in relation to the time interval in which a caller collects workload data.

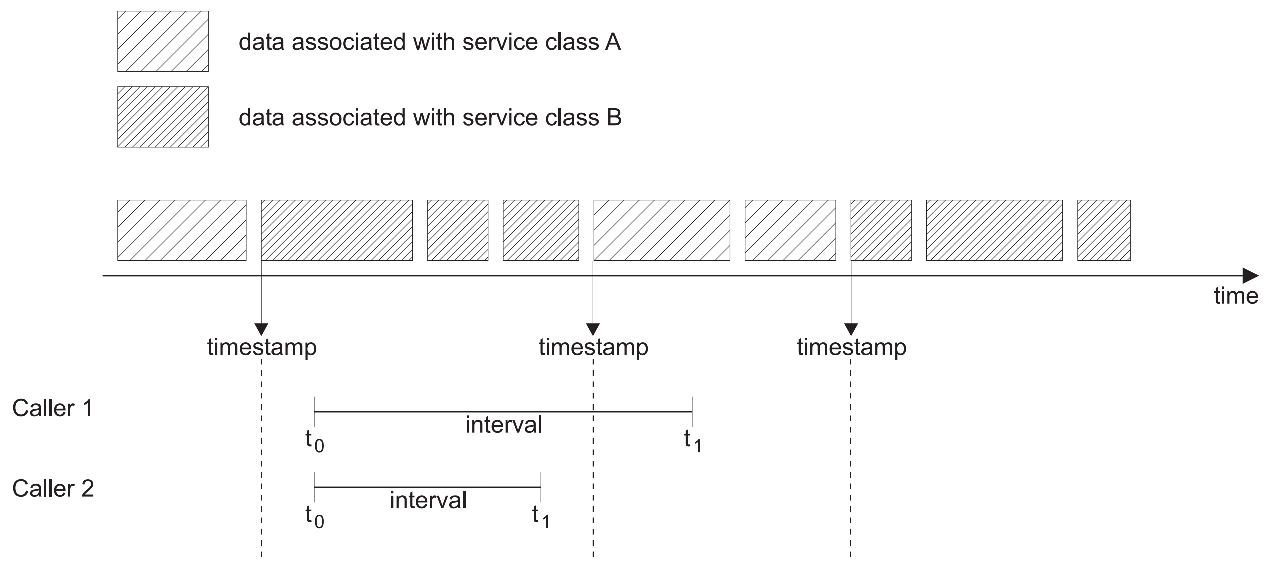


Figure 28. Mixed-class-indication timestamp in relation to the time interval

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 homogeneous 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 accumulate 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 22 shows a sample sequence of what to do to get address space information using IWMRQRY.

Table 22. Using IWMRQRY with the workload reporting services

Action	Reason
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
Issue `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 information 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 either in XML format, or as a data area mapped by the IWMSERVD mapping macro. The XML structure is defined by the DTD described in Appendix C, "Structure of the XML service definition (DTD)," on page 817. 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
- Service class attributes
- Report classes
- Resource groups
- Resource group attributes
- Application environments
- Scheduling environments
- Scheduling environment headers
- Scheduling environment resources
- Classifications
- Resources

Extensions can be specified in an XML service definition. For service definitions mapped by IWMSERVD, 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* or 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 29 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.

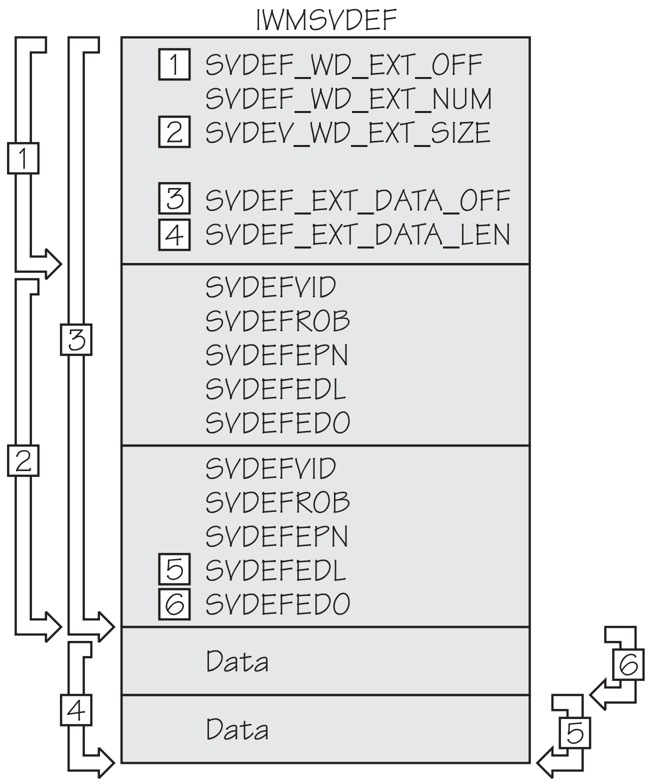


Figure 29. IWMSVDEF mapping for a service definition with workload extensions

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. Because 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. For a service definition in XML format, the macro parses the input to make sure it conforms to the DTD. Otherwise, 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 to allow for the system to complete the installation. If the IWMDINST macro finds an error during parsing or validity checking, it issues a reason code in the **`VALCHECK_RSN`** parameter. IWMDINST also returns the offset to indicate where an error was found in the IWMSVDEF mapping macro or the XML string, respectively. Note that no valid offset is returned in case the service definition type is XML and the primary reason code returned is `*xxxx*083d`. IWMDINST identifies no more than one error per call.

Appendix B, "Application validation reason codes," on page 801 contains a list of the reason codes, and their explanations. Table 23 shows an example of the information provided for reason code 1B01.

Table 23. 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.

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 in the **`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 mapped by the IWMSERVD mapping macro. To install the service definition onto the WLM couple data set, 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,
         TYPE=HEX,
         RETCODE=Module_Rc,
         RSNCODE=Module_Rsn,
         MF=E
```

With the following storage areas 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 801 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 IWMDEXTR 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 IWMDEXTR specifying **QUERYLEN** to determine the amount of storage required for the service definition The service definition returned by IWMDEXTR is not serialized against future installs, so the length returned could change before a caller can issue IWMDEXTR again. The caller should issue IWMDEXTR in a loop, checking return and reason codes, and obtain a larger storage area, if necessary.

Example of using IWMDEXTR to extract a service definition

To extract a service definition from the WLM couple data set, specify:

```
IWMDEXTR 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 IMWPACT 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 storage areas 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*.

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*. 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 storage areas 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.

SMF type 99 records have the following subtypes:

Subtype 1
Contains system level data, the trace of SRM actions, and data about resource groups. A subtype 1 record is written every policy interval.

Subtype 2
Contains 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
Contains 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
Contains 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 record is written every policy interval, which is frequent. The records contain the data, plots, and tables that 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 SMF type 99 record in the SMFPRM*xx* 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 SMFPRM*xx* parmlib member, you should include type 99 in your NOTYPE list. For example:

```
SUBSYS(STC,NOTYPE(99))
```

If you use TYPE in your SMFPRM*xx* 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 SMFPRM*xx* 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 IRASMF99 mapping macro for this record is supplied in SYS1.AMODGEN.

Identifying work in SMF type 99 records

You can identify work in SMF type 99 records 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:

$SRMDI*nn* - resource group discretionary
: Contains all work in a resource group with a discretionary goal. There is one $SRMDI*nn* 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.

$SRMS*nnn* - 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, two AORs may be serving the same three CICS service classes. Those AORs make up one $SRMS*nnn* service class.

  An address space can belong to one internal server service class, but can move from one class to another. The number of $SRMS*nnn* 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 more information, see *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 775.

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 detects and addresses underutilized resources, overutilized resources, and shortage conditions. 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 135 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 known 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.

Interpreting management policy data

This section explains the controls that SRM uses to manage work, and where these controls can be found in SMF type 99 records. 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
- Storage targets
- Cap slices
- I/O priority
- Number of server address spaces
- Buffer pool management data

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.

Storage target

To manage central storage, SRM defines a storage target of *protective central*. This specifies the number of frames that are protected in central storage. The system will not steal storage below this value.

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 ($SRMS*nnn*), 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 69) 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.

Buffer pool management data

For each buffer pool that DB2® has registered to WLM for dynamic buffer pool management, the actual size, the used size, the minimum and maximum size, the target size and the adjustment size are shown together with the buffer pool name and the owning address space.

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
- Buffer pool hitratio 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.

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.

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.

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.

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.

x axis
: The average address space size in frames.

y axis
: 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:

```
             using samples
------------------------------ × 100
 using samples + delay samples
```

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.

x axis
: The *proportional aggregate speed* of a service class period

y axis
: 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.

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.

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.

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.

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).

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.

Buffer pool hit ratio plot

SRM uses the buffer pool hit ratio plot to keep track of the relation between the size of the buffer pool and the hit ratio of DB2 buffer pool accesses. The hit ratio is percentage of hits, when looking for data in the buffer pool, related to the total number of attempts to read data from the buffer pool. There is one plot per buffer pool.

x axis
The size of the buffer pool in Megabytes. This is not the absolute size of the buffer pool, instead this is the delta to the defined minimum size.

y axis
The hitratio.

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 of interpreting the data in SMF type 99 records:
- 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 following example shows the 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 following example shows the subtype 2 service class period output:

```
CLASS    P I N  SC GT    SPI   LPI DP  MPLI MPLO  SWPT
APPC     1 2 N -99 SRT  1.10  1.10 251    0  999   488
APPC     2 2 N -99 SRT     0     0 251    0  999     0
APPC     3 2 N -72 SRT  1.10  1.10 251    1  999 76206
APPC     4 3 N -99 VEL     0     0 249    1  999     0
NRBATCH  1 3 N -99 LRT  2.16  3.04 249    1   11     0
NRBATCH  2 4 Y -98 LRT   .02   .03 247    2  999     0
NRBATCH  3 6 Y -99 DIS   .81   .81 192    0   14     0
OFFBATCH 1 6 Y -99 DIS   .81   .81 192    0   14     0
TSO      1 2 Y -64 SRT   .10   .11 251    1   11     0
TSO      2 2 Y   0 SRT  1.84  1.84 251    1  999  1464
TSO      3 2 Y  -2 SRT   .91   .93 247    2   12  2440
TSO      4 3 Y -99 VEL   .26   .26 245    7  999     0
$SRMDISC 1 6 Y -99 DIS   .81   .81 192    0   14     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	The service class period needs help.
N	The service class period does not 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

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. Because 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 here 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 in 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

The information in this part describes the Workload Management Services in detail.

Chapter 12. Workload management services

This chapter provides detailed information about the workload management services, including those that support both 31-bit and 64-bit addressing mode. Workload management services that support 64-bit addressing mode have names that start with IWM4*xxxx*.

For information about equivalent versions of some of these services that only support 31-bit addressing mode, see Appendix E, "WLM services supporting 31-bit addressing only," on page 827.

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:

Minimum authorization	Supervisor state or program key mask (PKM) allowing keys 0-7.
Dispatchable unit mode	Task or SRB
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 may be held.
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, 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. Because 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.

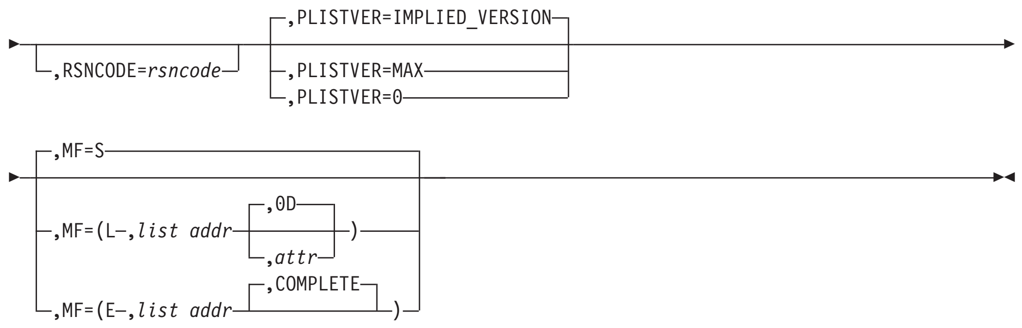
Performance implications

None.

Syntax

The syntax of the IWMCNTN macro is as follows:

```
►►──┬──────┬──IWMCNTN──SUBSYS=subsys──,SUBSYSNM=subsysnm───────────────────────────►
    └─name─┘

   ┌─,RESOURCESCOPE=SINGLESYSTEM─┐
►──┼─────────────────────────────┼──,RESOURCEID=resourceid─,RESOURCEID_LEN=resourceid_len──►
   └─,RESOURCESCOPE=MULTISYSTEM──┘

►──┬─,INVOCATIONTYPE=UPDATE─,REQUESTLIST=requestlist──┬──┬─────────────────┬──────►
   ├─,INVOCATIONTYPE=REPLACE─,REQUESTLIST=requestlist─┤  └─,RETCODE=retcode─┘
   └─,INVOCATIONTYPE=ENDOFCONTENTION──────────────────┘
```



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***,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

,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 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.

For each request the macro returns a return and a 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 Table 24:

Table 24. Work unit IDs

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
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.

4 Name: IwmRetCodeWarning Meaning: Successful completion, unusual conditions noted. Action: Check reason code

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.

0448 Name: IwmRsnCodePossibleDeadlock Meaning: The specified chronic resource contention may have caused a deadlock: The holder of resource (A) is waiting for resource (B), which is currently held by another holder, which is waiting for resource (A). Action: Check for possible deadlock.

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.
08A5	Name: IwmRsnCodeNoContention Meaning: The specified chronic resource contention is not stored in the topology. The DELETE request for this request list entry was not processed. Action: Correct the delete request.
08A8	Name: IwmRsnCodeDupContention Meaning: The specified chronic resource contention is already stored in the topology. The ADD request for this request list entry was not processed. Action: Correct the add request.
08AF	Name: IwmRsnCodeDeadlock Meaning: The specified chronic resource contention caused a deadlock: The holder of resource (A) is waiting for resource (B), which is currently held by another holder, which is waiting for resource (A). The request list entry was not processed. Action: Remove the deadlock.

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.

| **Note:** This keyword is ignored. WLM contention notifications are
| processed with `RESOURCESCOPE=SINGLESYSTEM`, even if
| `RESOURCESCOPE=MULTISYSTEM` is specified.

,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 codes 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 25. Return and Reason Codes for the IWMCNTN Macro

Return Code	Reason Code	Equate Symbol, Meaning, and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.

Table 25. Return and Reason Codes for the IWMCNTN Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
4	xxxx00447	**Equate Symbol**: IwmRsnCodeRequestListEntryWarning **Meaning**: The processing of at least one of the request list entries has caused a warning. Refer to the return code and reason code stored for this request list entry. **Action**: SRM continues with the request list processing.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: Caller is locked. **Action**: Avoid requesting this function while locked.
8	xxxx080B	**Equate Symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list. **Action**: Check for possible storage overlay.
8	xxxx0824	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: The caller invoked the service but was in 24-bit addressing mode. **Action**: Request this function only when you are in 31-bit addressing mode.
8	xxxx0825	**Equate Symbol**: IwmRsnCodeAscModeNotPrimary **Meaning**: The caller invoked the service but was not DAT on primary ASC mode. **Action**: Avoid requesting this function in this environment.
8	xxxx0828	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: Version number in parameter list is not valid. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx0888	**Equate Symbol**: IwmRsnCodeBadRequestList **Meaning**: The request list does not pass verification. No request in the request list was processed. **Action**: Check the return and reason codes in the request list.
8	xxxx0889	**Equate Symbol**: IwmRsnCodeBadResourceIdLen **Meaning**: The length of the resource id must not exceed 264 bytes. **Action**: Specify a correct resource id length.

Table 25. Return and Reason Codes for the IWMCNTN Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx088C	**Equate Symbol**: IwmRsnCodeBadRequestListVersion **Meaning**: The version specified in the request list is not supported **Action**: Specify a correct request list version.
8	xxxx088D	**Equate Symbol**: IwmRsnCodeBadRequestListLength **Meaning**: The specified request list length is too small to carry the specified number of request entries. **Action**: Specify a correct request list length or correct the entry count.
8	xxxx08A6	**Equate Symbol**: IwmRsnCodeBadRequestListEntry **Meaning**: The processing of at least one of the request list entries failed. Refer to the return code and reason code stored for this request list entry. SRM continues with the request list processing. **Action**: Check the return and reason codes in the request list.
8	xxxx08A7	**Equate Symbol**: IwmRsnCodeBadResource **Meaning**: The resource ID was not found in the topology for INVOCATIONTYPE=ENDOFCONTENTION. **Action**: Specify a correct resource ID.
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error. **Action**: 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
```

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:

Minimum authorization	Supervisor state or program key mask (PKM) allowing keys 0-7.
Dispatchable unit mode	Task
Cross memory mode	Any PASN, any HASN, any SASN
AMODE	31-bit

ASC mode	Primary or access register (AR) If in Access Register ASC mode, specify SYSSTATE ASCENV=AR before invoking this macro.
Interrupt status	Enabled for I/O and external interrupts
Locks	No locks may be held.
Control parameters	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.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

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:

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

The syntax of the IWMCQRY macro is as follows:

```
►►─┬──────┬─IWMCQRY──ANSAREA=ansarea──,ANSLEN=anslen──,QUERYLEN=querylen──────────►
   └─name─┘

►─┬─────────────────────┬─┬──────────────────┬─┬──────────────────┬───────────────►
  └─,POLICY_ID=policy_id─┘ └─,RETCODE=retcode─┘ └─,RSNCODE=rsncode─┘

  ┌─,PLISTVER=IMPLIED_VERSION─┐ ┌─,MF=S────────────────────────────────┐
►─┼───────────────────────────┼─┼──────────────────────────────────────┼──────────►◄
  ├─,PLISTVER=MAX─────────────┤ │                      ┌─,0D───┐       │
  └─,PLISTVER=0───────────────┘ ├─,MF=(L─,list addr─┴───────┴─)────┤
                                │                      └─,attr─┘       │
                                │                      ┌─,COMPLETE─┐   │
                                └─,MF=(E─,list addr─┴───────────┴─)─┘
```

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***,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_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
Explanation

X'0Axx0005'
An attempt to reference caller's parameters caused an OC4 abend.

Return codes 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 26. Return and Reason Codes for the IWMCQRY Macro

Return Code	Reason Code	Equate Symbol, Meaning, and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.

Table 26. Return and Reason Codes for the IWMCQRY Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
4	xxxx040A	**Equate Symbol**: IwmRsnCodeOutputAreaTooSmall **Meaning**: 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. **Action**: None required. If necessary, invoke the service again with an output area of sufficient size to receive all information.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0801	**Equate Symbol**: IwmRsnCodeSrbMode **Meaning**: The caller is in SRB mode. **Action**: Avoid requesting this function while in SRB mode.
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: The caller is locked. **Action**: Avoid requesting this function while locked.
8	xxxx0810	**Equate Symbol**: IwmRsnCodeEutFrr **Meaning**: The caller has EUT FRR established. **Action**: Avoid requesting this function with an EUT FRR set.
8	xxxx0823	**Equate Symbol**: IwmRsnCodeDatoff **Meaning**: The caller invoked the service while DATOFF **Action**: Avoid requesting this function in this environment.
8	xxxx0824	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: The caller invoked the service but was in 24-bit addressing mode. **Action**: Request this function only when you are in 31-bit addressing mode.
8	xxxx0827	**Equate Symbol**: IwmRsnCodeRsvdNot0 **Meaning**: Reserved field in parameter list was non-zero. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx0828	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: Version number in parameter list is not valid or the length specified is incorrect. **Action**: Check for possible storage overlay of the parameter list.

Table 26. Return and Reason Codes for the IWMCQRY Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx0830	**Equate Symbol**: IwmRsnCodeBadAlet **Meaning**: The caller has passed an invalid ALET. **Action**: Check for possible storage overlay of the parameter list or variable.
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error. **Action**: No action required. The function may be successful if invoked again.

IWMDEXTR — Extract WLM 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 data set 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 data set.

The service definition can be extracted either in XML format, or as a data area mapped by the IWMSERVD mapping macro. The XML structure is defined by the DTD described in Appendix C, "Structure of the XML service definition (DTD)," on page 817. The IWMSERVD mapping 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 define 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.

Because 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 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 extract macro is provided in PL/AS and assembler formats.

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:

Minimum authorization	Problem state, any PSW key. The caller must have read authority to the resource MVSADMIN.WLM.POLICY in the FACILITY class.
Dispatchable unit mode	Task
Cross memory mode	PASN=HASN=SASN
AMODE	31-bit
ASC mode	Primary or access register (AR) If in Access Register ASC mode, specify SYSSTATE ASCENV=AR before invoking this macro.
Interrupt status	Enabled for I/O and external interrupts
Locks	No locks may be held.
Control parameters	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 caller is in key 0.

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. The caller cannot have an EUT FRR established.
2. This macro supports multiple versions. Some keywords are unique to certain versions. For further information, see the PLISTVER parameter description.

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

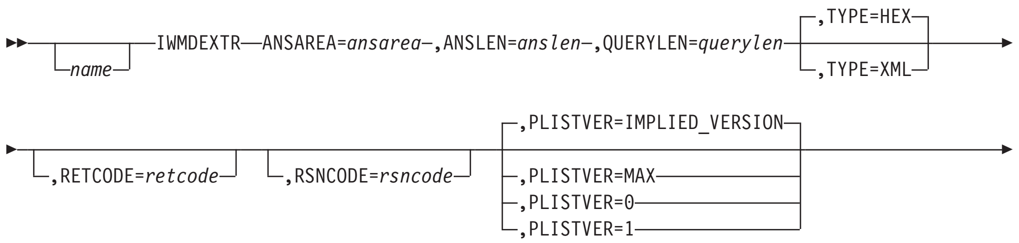
Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

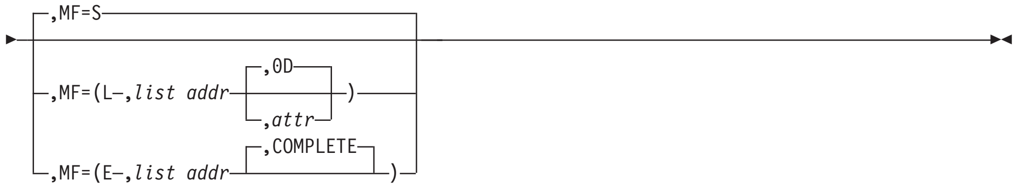
Performance implications

None.

Syntax

The syntax of the IWMDEXTR macro is as follows:





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.

ANSAREA=*ansarea*

A required output parameter, variable specifying an area to contain the service definition data returned by the extract service. When TYPE=HEX is specified, this area is defined by the IWMSERVD macro. When TYPE=XML is specified, it is the area where the service definition XML stream is to be stored.

To code: Specify the RS-type address, or address in register (2)-(12), of a character field.

Note: The specified address must not be 0.

,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, or specify a literal decimal value.

,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**, which supports all parameters except those specifically referenced in higher versions.
- **1**, which supports the following parameter and those from version 0:

  TYPE

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 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. If you specify 15, GPR15, REG15, or R15 (with or without parentheses), the value will be left in GPR 15.

To code: Specify the RS-type address of a fullword field, or register (2)-(12), or (15), (GPR15), (REG15), or (R15).

,RSNCODE=*rsncode*
An optional output parameter into which the reason code is to be copied from GPR 0. If you specify 0, 00, GPR0, GPR00, REG0, REG00, or R0 (with or without parentheses), the value will be left in GPR 0.

To code: Specify the RS-type address of a fullword field, or register (0) or (2)-(12), (00), (GPR0), (GPR00), REG0), (REG00), or (R0).

,TYPE=HEX
,TYPE=XML
An optional parameter indicating in which format the service definition will be extracted. The default is TYPE=HEX.

,TYPE=HEX
The service definition will be returned in HEX format mapped by IWMSERVD (default).

,TYPE=XML
The service definition will be returned in XML format.

ABEND codes

Reason Code (Hex)
Explanation

X'0Axx0005'
An attempt to reference caller's parameters caused an OC4 abend.

Return codes 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.

Table 27. Return and Reason Codes for the IWMDEXTR Macro

Return Code	Reason Code	Equate Symbol, Meaning and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.

Table 27. Return and Reason Codes for the IWMDEXTR Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning and Action
4	xxxx040A	**Equate Symbol**: IwmRsnCodeOutputAreaTooSmall **Meaning**: 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. **Action**: None required. If necessary, reinvoke the service with an output area of sufficient size to receive all information.
4	xxxx0414	**Equate Symbol**: IwmRsnCodeNullCDS **Meaning**: No service definition is currently installed. As a result, no service definition data is returned. **Action**: None required.
4	xxxx0417	**Equate Symbol**: IwmRsnCodeBadServDE **Meaning**: Service definition retrieved from WLM CDS has failed validation but the structure is still returned to the caller. **Action**: None required. Caution is advised in using the structure.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0801	**Equate Symbol**: IwmRsnCodeSrbMode **Meaning**: Caller is in SRB mode. **Action**: Avoid requesting this function while in SRB mode.
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: The caller is disabled. **Action**: Avoid requesting this function while disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: The caller is locked. **Action**: Avoid requesting this function while locked.
8	xxxx0810	**Equate Symbol**: IwmRsnCodeEutFrr **Meaning**: The caller has EUT FRR established. **Action**: Avoid requesting this function with an EUT FRR set.
8	xxxx0823	**Equate Symbol**: IwmRsnCodeDatoff **Meaning**: The caller invoked the service while DATOFF **Action**: Avoid requesting this function in this environment.

Table 27. Return and Reason Codes for the IWMDEXTR Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning and Action
8	xxxx0824	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: The caller invoked the service but was in 24-bit addressing mode. **Action**: Request this function only when you are in 31-bit addressing mode.
8	xxxx0827	**Equate Symbol**: IwmRsnCodeRsvdNot0 **Meaning**: Reserved field in parameter list was non-zero. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx0828	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: Version number in parameter list is not valid or the length specified is incorrect. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx0830	**Equate Symbol**: IwmRsnCodeBadAlet **Meaning**: The caller has passed an invalid ALET. **Action**: Check for possible storage overlay of the parameter list or variable.
8	xxxx0841	**Equate Symbol**: IwmRsnCodeXMemMode **Meaning**: The caller is in cross-memory mode. **Action**: Invoke the function in non-cross memory mode.
8	xxxx083E	**Equate Symbol**: IwmRsnCodeLevelMismatch **Meaning**: The service definition retrieved from the WLM couple data set is at a higher level than the WLM code running on this system. A system with a lower level version cannot extract this service policy because it is not capable of handling all the function in the service definition. **Action**: None required. If necessary, invoke the service again on a higher level system.
8	xxxx085B	**Equate Symbol**: IwmRsnCodeZeroAnsArea **Meaning**: The caller invoked the service with an address of zero for parameter ANSAREA. **Action**: Specify a non-zero address.
C	—	**Equate Symbol**: IwmRetCodeEnvError **Meaning**: Environmental error.
C	xxxx0C01	**Equate Symbol**: IwmRsnCodeNoStg **Meaning**: Storage is not available for the request. **Action**: There is a storage shortage. The function may work successfully at a later time.

Table 27. Return and Reason Codes for the IWMDEXTR Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning and Action
C	xxxx0C0E	**Equate Symbol**: IwmRsnCodeInsufAccess **Meaning**: The caller does not have read authority to the RACF® resource MVSADMIN.WLM.POLICY in the FACILITY class. **Action**: Invoke the function when the condition is fulfilled.
C	xxxx0C0F	**Equate Symbol**: IwmRsnCodeCDSNotAvail **Meaning**: 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. **Action**: No action required.
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error. **Action**: No action required. The function may be successful if invoked again.
10	xxxx0F09	**Equate Symbol**: IwmRsnCodeUnknownLvl **Meaning**: Internal error. **Action**: Contact IBM.

IWMDINST — Install a 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 data set for the current sysplex. After this service definition is installed in the couple data set all policies contained in it are eligible to be activated in the sysplex.

The service definition can be installed either in XML format, or as a data area mapped by the IWMSERVD mapping macro. The XML structure is defined by the DTD described in Appendix C, “Structure of the XML service definition (DTD),” on page 817. The IWMSERVD mapping 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:

General service definition 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.

Service definition classification rules 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.

Notepad data area
Contains any comments (or change history) that an installation chooses to associate with the service definition. This area is mapped by IWMSVNPA.

Service definition application environment data area
Contains the definitions of the application environments. This area is mapped by IWMSVAEA.

Service definition scheduling environment data area
Contains the definitions of the scheduling environments. This area is mapped by IWMSVSEA.

The service definition descriptor element and all five data areas of the service definition must be passed as input 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 service definition in the WLM couple data set. This allows the caller the ability to prevent inadvertent overwrites 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 description of the **COND** keyword.

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:

Minimum authorization:	Problem state. Any PSW key. The caller must have update authority to the resource MVSADMIN.WLM.POLICY in the FACILITY class.
Dispatchable unit mode:	Task
Cross memory mode:	PASN=HASN=SASN
AMODE:	31-bit
ASC mode:	Primary or access register (AR) If in Access Register ASC mode, specify SYSSTATE ASCENV=AR before invoking this macro.
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks may be held.
Control parameters:	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. 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. The caller cannot have an EUT FRR established.
2. This macro supports multiple versions. Some keywords are unique to certain versions. See the PLISTVER parameter description.

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.

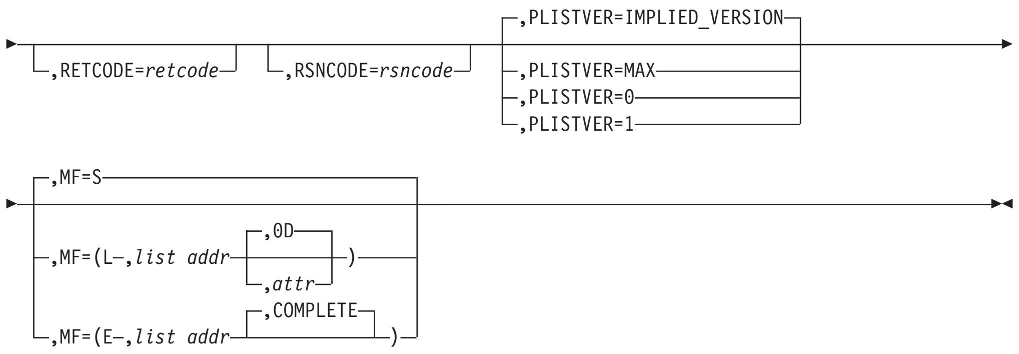
Performance implications

None.

Syntax

The syntax of the IWMDINST macro is as follows:

```
                                                 ┌─,TYPE=HEX──────────────────────┐
►►──┬──────┬──IWMDINST──SERVD_AREA=servd_area──┤                                ├──────────►
    └─name─┘                                     └─,TYPE=XML─,XML_LEN=xml_len─────┘

►──┬───────────────────────────┬─,PRODUCT_ID=product_id──,VALCHECK_RSN=valcheck_rsn────────►
   └─,QRY_BASEID=qry_baseid────┘

►─,VALCHECK_OFFSET=valcheck_offset──┬─,COND=YES─,IN_BASEID=in_baseid─┬─────────────────────►
                                    └─,COND=NO───────────────────────┘
```



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 was 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***,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**, which supports all parameters except those specifically referenced in higher versions.
- **1**, which supports the following parameters and those from version 0:

  TYPE
  XML_LEN

To code: Specify one of the following:
- IMPLIED_VERSION
- MAX
- A decimal value of 0, or 1

,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 data set. 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).

,TYPE=HEX
,TYPE=XML
An optional parameter indicating in which format the service definition will be extracted. The default is TYPE=HEX.

,TYPE=HEX
The service definition will be returned in HEX format mapped by IWMSERVD (default).

,TYPE=XML
The service definition will be returned in XML format.

SERVD_AREA=*servd_area*
A required input parameter, variable specifying an area that contains the service definition data to be installed. When TYPE=HEX is specified, this area is defined by the IWMSERVD macro. When TYPE=XML is specified, it is the area where the service definition XML stream is to be stored.

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. The variable will contain the offset identifying the specific error in the input service definition found during validity checking. If the primary reason code is xxxx083D and the service definition type is HEX, the variable contains 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. If the primary reason code is xxxx083D and the service definition type is XML, the VALCHECK_OFFSET has no meaning. If the primary reason code is xxxx08B2 or xxx08B5, the variable contains the offset from the beginning of the service definition in XML format. 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. The variable 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, xxxx08B2, or xxxx08B5 is returned. If reason codes xxxx083D and xxxx08B5 are issued, refer to Appendix B, “Application validation reason codes,” on page 801 for an explanation. If the primary reason code is xxxx08B2, refer to "*XML System Services Users Guide and Reference*" for an explanation.

To code: Specify the RS-type address, or address in register (2)-(12), of a fullword field.

,XML_LEN=*xml_len*

When TYPE=XML is specified, a required input parameter. The variable contains the length of the area specified on the SERVD_AREA keyword to contain the service definition data in XML format.

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 codes 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 28. Return and Reason Codes for the IWMDINST Macro

Return Code	Reason Code	Equate Symbol, Meaning, and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.

Table 28. Return and Reason Codes for the IWMDINST Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.
4	xxxx0413	**Equate Symbol**: IwmRsnCodeIdsDontMatch **Meaning**: 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. **Action**: None required.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0801	**Equate Symbol**: IwmRsnCodeSrbMode **Meaning**: The caller is in SRB mode. **Action**: Avoid requesting this function while in SRB mode.
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: The caller is locked. **Action**: Avoid requesting this function while locked.
8	xxxx0810	**Equate Symbol**: IwmRsnCodeEutFrr **Meaning**: The caller has EUT FRR established. **Action**: Avoid requesting this function with an EUT FRR set.
8	xxxx0823	**Equate Symbol**: IwmRsnCodeDatoff **Meaning**: The caller invoked the service while DATOFF **Action**: Avoid requesting this function in this environment.
8	xxxx0824	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: The caller invoked the service but was in 24-bit addressing mode. **Action**: Request this function only when you are in 31-bit addressing mode.
8	xxxx0827	**Equate Symbol**: IwmRsnCodeRsvdNot0 **Meaning**: Reserved field in parameter list was non-zero. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx0828	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: Version number in parameter list is not valid or the length specified is incorrect. **Action**: Check for possible storage overlay of the parameter list.

Table 28. Return and Reason Codes for the IWMDINST Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx0829	**Equate Symbol**: IwmRsnCodeBadOptions **Meaning**: Parameter list omits required parameters or supplies mutually exclusive parameters or provides data associated with options not selected. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx0830	**Equate Symbol**: IwmRsnCodeBadAlet **Meaning**: The caller has passed an invalid ALET. **Action**: Check for possible storage overlay of the parameter list or variable.
8	xxxx083D	**Equate Symbol**: IwmRsnCodeBadServDI **Meaning**: The caller has passed a Service Definition area that failed validation. **Action**: See values in VALCHECK_RSN and VALCHECK_OFFSET parameters for more information concerning the specific failure.
8	xxxx0841	**Equate Symbol**: IwmRsnCodeXMemMode **Meaning**: The caller is in cross-memory mode. **Action**: Invoke the function in non-cross memory mode.
8	xxxx085B	**Equate Symbol**: IwmRsnCodeNoSERVDArea **Meaning**: The caller invoked the service without a required SERVD area or the SERVD area address is 0. **Action**: Check for possible storage overlay of the parameter list or variable.
8	xxxx08B2	**Equate Symbol**: IwmRsnCodeParserError **Meaning**: The XML parser was not able to parse the service definition XML document. **Action**: See values in VALCHECK_RSN and VALCHECK_OFFSET parameters for more information concerning the specific failure. The XML parser reason codes can be found in the book: "XML System Services Users Guide and Reference".
8	xxxx08B3	**Equate Symbol**: IwmRsnCodeXmlZeroLen **Meaning**: The length of the provided XML service definition is zero. **Action**: Provide the correct length of the service definition.
8	xxxx08B5	**Equate Symbol**: IwmRsnCodeXmlInvalid **Meaning**: The XML parser was not able to parse the service definition XML document. **Action**: See values in VALCHECK_RSN and VALCHECK_OFFSET parameters for more information concerning the specific failure. The reason codes can be found in *z/OS MVS Programming: Workload Management Services* .
C	—	**Equate Symbol**: IwmRetCodeEnvError **Meaning**: Environmental error.

Table 28. Return and Reason Codes for the IWMDINST Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
C	xxxx0C01	**Equate Symbol**: IwmRsnCodeNoStg **Meaning**: Storage is not available for the request. **Action**: There is a storage shortage. The function may work successfully at a later time.
C	xxxx0C0E	**Equate Symbol**: IwmRsnCodeInsufAccess **Meaning**: The caller does not have update authority to the RACF resource MVSADMIN.WLM.POLICY in the FACILITY class. **Action**: Invoke the function when the condition is fulfilled.
C	xxxx0C0F	**Equate Symbol**: IwmRsnCodeCDSNotAvail **Meaning**: 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. **Action**: No action required.
C	xxxx0C10	**Equate Symbol**: IwmRsnCodeCDSTooSmall **Meaning**: WLM CDS is too small to process the request. **Action**: No action required.
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error. **Action**: No action required. The function may be successful if invoked again.
10	xxxx0F0A	**Equate Symbol**: IwmRsnCodeEndOfBuffer **Meaning**: Internal error. **Action**: Contact IBM.

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:

Minimum authorization:	Supervisor state or program key mask (PKM) allowing keys 0-7
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31- or 64-bit. If in 64-bit addressing mode, code SYSSTATE AMODE64=YES before invoking this macro.
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks may be held.
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, 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 EWLMMODE=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

This macro supports multiple versions. Some keywords are unique to certain versions. See the PLISTVER parameter description.

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

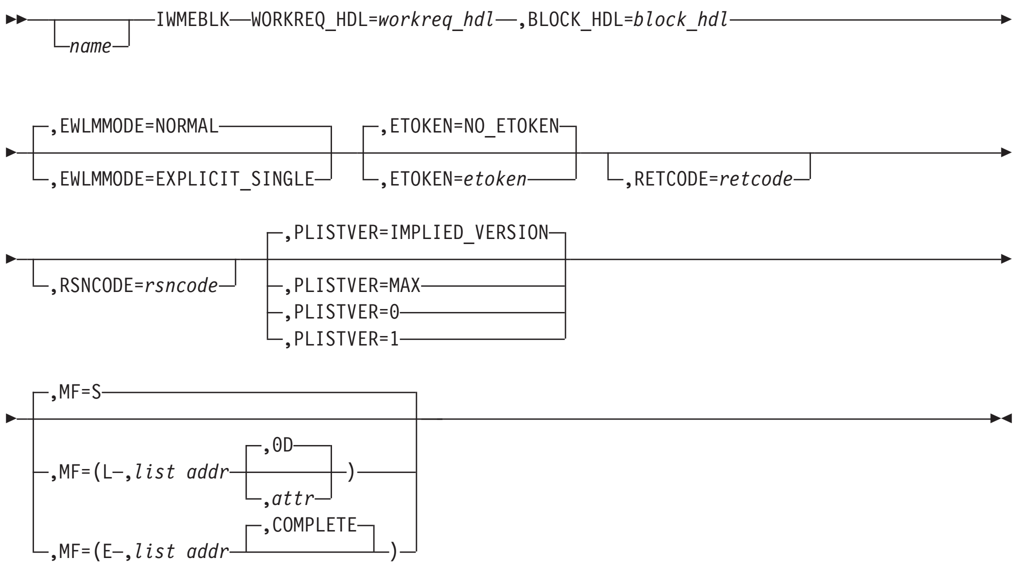
Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

Performance implications

None.

Syntax

The syntax of the IWMEBLK macro is as follows:



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=etoken** is required, if the option EWLMMODE=EXPLICIT_SINGLE is specified.
  - **ETOKEN=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, 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**, which supports all parameters except those specifically referenced in higher versions.
- 1, which supports 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 codes 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 29. Return and Reason Codes for the IWMEBLK Macro

Return Code	Reason Code	Equate Symbol, Meaning, and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: Caller is locked. **Action**: Avoid requesting this function while locked.
8	xxxx080B	**Equate Symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list. **Action**: Check for possible storage overlay.
8	xxxx0824	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: The caller invoked the service but was in 24-bit addressing mode. **Action**: Request this function only when you are in 31-bit addressing mode.
8	xxxx0828	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: The Version number in the parameter list is not valid. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx083A	**Equate Symbol**: IwmRsnCodeBadEnclave **Meaning**: EWLMMODE=EXPLICIT_SINGLE was specified and the required enclave token does not pass verification. **Action**: Check for possible storage overlay of the enclave token, or for asynchronous events which may have deleted the enclave.
8	xxxx0895	**Equate Symbol**: IwmRsnCodeEWLMServNotEnabled **Meaning**: The service is not enabled because the caller invoked the IWM4CON (IWMCONN) service with EWLM=NO. **Action**: Ensure that EWLM=YES is specified on the IWM4CON (IWMCONN) request to enable this service.

Table 29. Return and Reason Codes for the IWMEBLK Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx0896	**Equate Symbol**: IwmRsnCodeBadWorkReqHandle **Meaning**: The work request handle is invalid. **Action**: Check the specification of the WORKREQ_HDL parameter.
8	xxxx08A0	**Equate Symbol**: IwmRsnCodeNotExplicitSingle **Meaning**: 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. **Action**: 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).
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error. **Action**: No action required. The function may be successful if invoked again.

Examples

```
         IWMEBLK WORKREQ_HDL=WRHANDLE,
                 BLOCK_HDL=BKHANDLE
*
* Storage areas
*
WRHANDLE DS    CL8             Work Request Handle
BKHANDLE DS    CL8             Work Request Block Handle
```

IWMEDREG — Deregister a WLM enclave

The IWMEDREG service allows the caller to deregister an enclave which it previously registered using the IWMEREG 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:

Minimum authorization:	Supervisor state or program key mask (PKM) allowing keys 0-7.
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31-bit
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks may be held.
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, 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

None.

Input register information

Before issuing the IWMEDREG 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

The syntax of the IWMEDREG macro is as follows:

```
►►─┬──────┬─IWMEDREG──REGTOKEN=regtoken──,ETOKEN=etoken──┬──────────────────┬──────────►
   └─name─┘                                              └─,RETCODE=retcode─┘

                            ┌─,PLISTVER=IMPLIED_VERSION─┐
►─┬───────────────────┬─────┼───────────────────────────┼─────────────────────────────►
  └─,RSNCODE=rsncode──┘     ├─,PLISTVER=MAX─────────────┤
                            └─,PLISTVER=0───────────────┘
```

```
    ┌─,MF=S──────────────────────────────────┐
►───┼────────────────────────────────────────┼──────────────────────────────────────►◄
    │                       ┌─,0D───┐         │
    ├─,MF=(L─,list addr─────┼───────┼─)──────┤
    │                       └─,attr─┘         │
    │                       ┌─,COMPLETE─┐     │
    └─,MF=(E─,list addr─────┴───────────┴─)──┘
```

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***,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 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 codes 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	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: The caller is locked. **Action**: Avoid requesting this function while locked.
8	xxxx080B	**Equate Symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list. **Action**: Check for possible storage overlay.
8	xxxx0824	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: The caller invoked the service but was in 24-it addressing mode. **Action**: Request this function only when you are in 31-bit addressing mode.
8	xxxx0828	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: Version number in parameter list is not valid. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx083A	**Equate Symbol**: IwmRsnCodeBadEnclave **Meaning**: Enclave token does not pass verification. **Action**: Check for possible storage overlay of the enclave token, or asynchronous events which may have deleted the enclave.
8	xxxx0880	**Equate Symbol**: IwmRsnCodeBadRegToken **Meaning**: The register token does not pass verification. **Action**: Check for possible storage overlay of the register token, or asynchronous events which may have deregistered the enclave already.
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error. **Action**: 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
```

IWMEGCOR — Retrieve 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:

Minimum authorization:	Supervisor state or program key mask (PKM) allowing keys 0-7
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31- or 64-bit. If in 64-bit addressing mode, code SYSSTATE AMODE64=YES before invoking this macro.
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks may be held.
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, 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 EWLMMODE=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

This macro supports multiple versions. Some keywords are unique to certain versions. See the PLISTVER parameter description.

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.

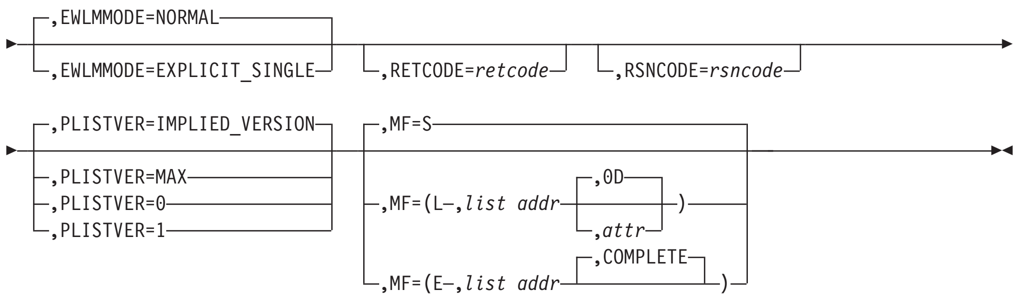
Performance implications

None.

Syntax

The syntax of the IWMEGCOR macro is as follows:

```
►►──┬──────┬──IWMEGCOR──┬─GET=CORRELATOR─,WORKREQ_HDL=workreq_hdl─,EWLM_CORR=ewlm_corr─┬──────►
    └─name─┘            ├─GET=MAXLENGTH─,EWLM_CORRML=ewlm_corrml─────────────────────┤
                        │ ┌─,ETOKEN=NO_ETOKEN─┐                                       │
                        └─┴───────────────────┴───────────────────────────────────────┘
                          └─,ETOKEN=etoken────┘
```



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=etoken** is required, if the option EWLMMODE=EXPLICIT_SINGLE is specified.
- **ETOKEN=NO_TOKEN** 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.

,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.

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***,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**, which supports all parameters except those specifically referenced in higher versions.
- 1, which supports 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 codes 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.

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	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: The caller is locked. **Action**: Avoid requesting this function while locked.
8	xxxx080B	**Equate Symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list. **Action**: Check for possible storage overlay.
8	xxxx0824	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: The caller invoked the service but was in 24-it addressing mode. **Action**: Request this function only when you are in 31-it addressing mode.
8	xxxx0828	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: Version number in parameter list is not valid. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx083A	**Equate Symbol**: IwmRsnCodeBadEnclave **Meaning**: EWLMMODE=EXPLICIT_SINGLE was specified and the required enclave token does not pass verification. **Action**: Check for possible storage overlay of the enclave token, or for asynchronous events which may have deleted the enclave.
8	xxxx0895	**Equate Symbol**: IwmRsnCodeEWLMServNotEnabled **Meaning**: The service is not enabled because the caller invoked the IWM4CON (IWMCONN) service with EWLM=NO. **Action**: Ensure that EWLM=YES is specified on the IWM4CON request to enable this service.

Table 31. Return and Reason Codes for the IWMEGCOR Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx0896	**Equate Symbol**: IwmRsnCodeBadWorkReqHandle **Meaning**: Work request handle in parameter list is not valid. **Action**: Check the specification of the WORKREQ_HDL parameter.
8	xxxx08A0	**Equate Symbol**: IwmRsnCodeNotExplicitSingle **Meaning**: 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. **Action**: 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).
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error. **Action**: No action required. The function may be successful if invoked again.

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
```

IWMEJOIN — Join WLM enclave

The purpose of this service is to allow the task (TCB) invoking this service to join an enclave for the purpose of performance management. The scope of this service affects only a single TCB at the time the service is invoked unless `SUBTASKS=YES` is in effect. Any TCBs which are attached by the current TCB 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. Unless `SUBTASKS=YES` is in effect, 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 TCBs 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 TCB (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 TCB. Management and reporting for the enclave will include activity for the TCB until the TCB either leaves the enclave or the enclave is deleted.

Environment

The requirements for the caller are:

Minimum authorization:	Supervisor state or program key mask (PKM) allowing keys 0-7.
Dispatchable unit mode:	Task
Cross memory mode:	Otherwise: Any PASN, any HASN, any SASN
AMODE:	31-bit
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks may be held.
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, 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.
2. The caller cannot have an EUT FRR established.
3. This macro supports multiple versions. Some keywords are unique to certain versions. See the PLISTVER parameter description.

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

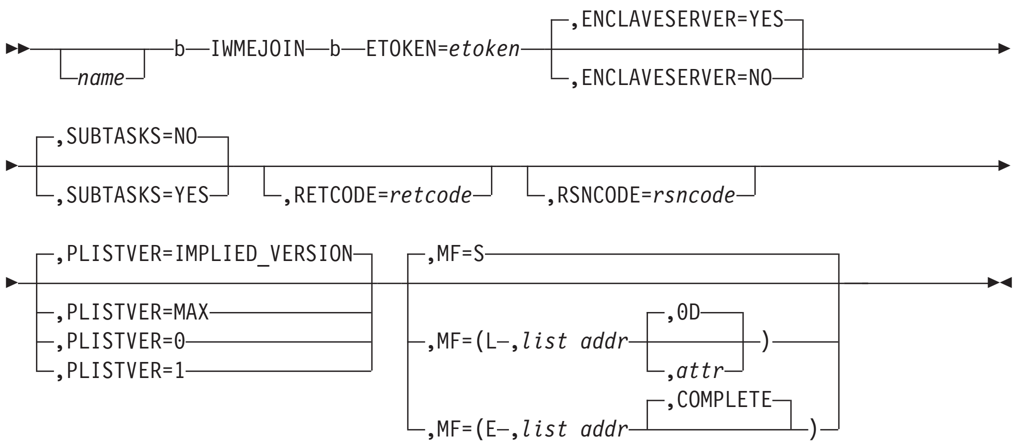
Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

Performance implications

None.

Syntax

main diagram



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.

,ENCLAVESERVER=YES
,ENCLAVESERVER=NO
An optional parameter, for internal use only The default is ENCLAVESERVER=YES.

,ENCLAVESERVER=YES

for internal use only

,ENCLAVESERVER=NO

for internal use only

ETOKEN=*etoken*
A required input parameter, which contains the enclave token to be associated with the TCB 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
,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:

ENCLAVESERVER

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. If you specify 15, GPR15, REG15, or R15 (within or without parentheses), the value will be left in GPR 15.

To code: Specify the RS-type address of a fullword field, or register (2)-(12) or (15), (GPR15), (REG15), or (R15).

,RSNCODE=*rsncode*

An optional output parameter into which the reason code is to be copied from GPR 0. If you specify 0, 00, GPR0, GPR00, REG0, REG00, or R0 (within or without parentheses), the value will be left in GPR 0.

To code: Specify the RS-type address of a fullword field, or register (0) or (2)-(12), (00), (GPR0), (GPR00), REG0), (REG00), or (R0).

,SUBTASKS=NO
,SUBTASKS=YES

An optional parameter, which specifies if subtasks of the joining task are also to be processed. The default is SUBTASKS=NO.

,SUBTASKS=NO

specifies that subtasks of the joining task are not to be processed.

,SUBTASKS=YES

specifies that subtasks of the joining task that are not already joined to an enclave are to be joined to the enclave identified by this invocation's ETOKEN parameter. When a currently-dispatched subtask is joined to the enclave, its CPU time for that dispatch is associated with the enclave rather than the address space. When the subtask is removed from the enclave, if it is currently dispatched, its CPU time for that dispatch is associated with the address space rather than the enclave.

If the caller does not have PASN=HASN, this is treated as SUBTASKS=NO. The caller is notified with a return and reason code combination.

If SYSEVENT REQSRMST does not indicate, via bit SRMSTSTS being on, that this function is available, this is treated as SUBTASKS=NO.

When SUBTASKS=YES is in effect, this task's corresponding IWMELEAV will also perform leave processing upon any subtasks that are implicitly associated with the enclave. This includes subtasks that were joined to the enclave due to this task's IWMEJOIN processing as well as subtasks that were joined to the enclave by ATTACH processing.

ABEND codes

None.

Return codes and reason codes

When the IWMEJOIN 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 32. Return and Reason Codes for the IWMEJOIN Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.
4	xxxx041F	**Equate Symbol**: IwmRsnCodeExecEnvChanged **Meaning**: The execution environment has changed while the requested function is in progress. **Action**: None required.
4	xxxx044D	**Equate Symbol**: IwmRsnCodeXmSoNoSubtasks **Meaning**: IWMEJOIN requesting SUBTASKS=YES was issued with the primary address space not equal to the home address space. No processing of subtasks was done. The rest of Join processing completed successfully. **Action**: Issue IWMEJOIN with PASN=HASN if you need the subtasks processed.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0801	**Equate Symbol**: IwmRsnCodeSrbMode **Meaning**: Caller is in SRB mode. **Action**: Avoid requesting this function while in SRB mode.
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: Caller is locked. **Action**: Avoid requesting this function while locked.
8	xxxx080B	**Equate Symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list. **Action**: Check for possible storage overlay.
8	xxxx0810	**Equate Symbol**: IwmRsnCodeEutFrr **Meaning**: Caller has EUT FRR established. **Action**: Avoid requesting this function with an EUT FRR set.

Table 32. Return and Reason Codes for the IWMEJOIN Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0823	**Equate Symbol**: IwmRsnCodeDatoff **Meaning**: Caller invoked service while DATOFF **Action**: Avoid requesting this function in this environment.
8	xxxx0824	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: Caller invoked service but was in 24 bit addressing mode. **Action**: Request this function only when you are in 31 bit addressing mode.
8	xxxx0825	**Equate Symbol**: IwmRsnCodeAscModeNotPrimary **Meaning**: Caller invoked service but was not DAT on Primary ASC mode. **Action**: Avoid requesting this function in this environment.
8	xxxx0826	**Equate Symbol**: IwmRsnCodeTaskTerm **Meaning**: Caller invoked service while task termination is in progress for the current TCB. **Action**: Avoid requesting this function while task termination is in progress.
8	xxxx0828	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: Version number in parameter list is not valid or version length field is not valid. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx083A	**Equate Symbol**: IwmRsnCodeBadEnclave **Meaning**: Enclave token does not pass verification. **Action**: Check for possible storage overlay of the enclave token, or asynchronous events which may have deleted the enclave.
8	xxxx0850	**Equate Symbol**: IwmRsnCodeBeginEnvOutstanding **Meaning**: Caller is already operating under an outstanding Begin environment which has implicitly joined an enclave. **Action**: Avoid requesting this function in this environment.
8	xxxx0857	**Equate Symbol**: IwmRsnCodeAlreadyInEnclave **Meaning**: Current dispatchable workunit is already in an enclave. **Action**: Avoid requesting this function while the caller is already in an enclave.
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error. **Action**: Contact your system programmer.

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
```

IWMELEAV — Leave WLM enclave

The purpose of this service is to allow the task (TCB) invoking this service to leave an enclave. In addition, if the join of this task to the enclave specified SUBTASKS=YES, and the primary address space matches the home address space (PASN=HASN) on this leave request, then any subtasks that are implicitly joined to the enclave (by that join, or by ATTACH processing) leave the 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 TCB 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 leave the enclave upon its own (task) termination, when the enclave is deleted (IWM4EDEL) or when leave processing for SUBTASKS=YES is being performed. 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 TCB (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:

Minimum authorization:	Supervisor state or program key mask (PKM) allowing keys 0-7.
Dispatchable unit mode:	Task
Cross memory mode:	When the joining task had subtasks processed due to SUBTASKS=YES and subtask processing is needed: PASN=HASN Otherwise: Any PASN, any HASN, any SASN
AMODE:	31-bit
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks may be held.
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, 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.
2. 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.

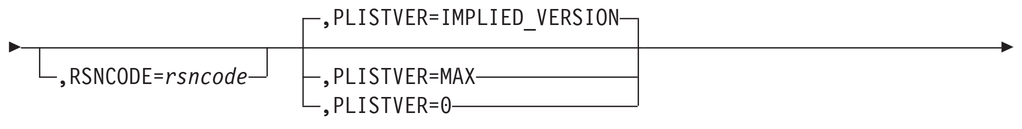
Performance implications

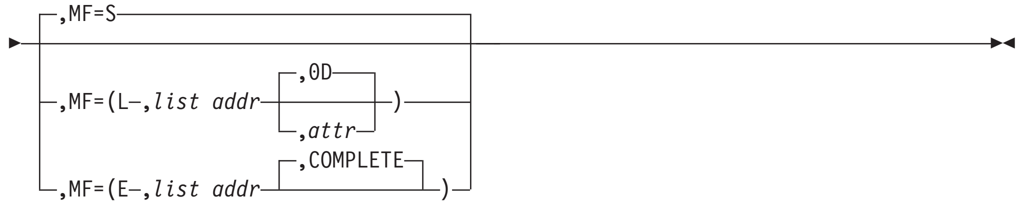
None.

Syntax

main diagram

```
►►─┬──────┬─b─IWMELEAV─b─ETOKEN=etoken─┬──────────────────┬──────────►
   └─name─┘                             └─,RETCODE=retcode─┘
```





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.
- **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. If you specify 15, GPR15, REG15, or R15 (within or without parentheses), the value will be left in GPR 15.

To code: Specify the RS-type address of a fullword field, or register (2)-(12) or (15), (GPR15), (REG15), or (R15).

,RSNCODE=*rsncode*
An optional output parameter into which the reason code is to be copied from GPR 0. If you specify 0, 00, GPR0, GPR00, REG0, REG00, or R0 (within or without parentheses), the value will be left in GPR 0.

To code: Specify the RS-type address of a fullword field, or register (0) or (2)-(12), (00), (GPR0), (GPR00), REG0), (REG00), or (R0).

ABEND codes

None.

Return codes and reason codes

When the IWMELEAV 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 33. Return and Reason Codes for the IWMELEAV Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.
4	xxxx041C	**Equate Symbol**: IwmRsnCodeNotEnclave **Meaning**: The current dispatchable workunit is not associated with an enclave. **Action**: Check for possible asynchronous events which may have deleted the enclave.
4	xxxx044D	**Equate Symbol**: IwmRsnCodeXmSoNoSubtasks **Meaning**: The corresponding IWMEJOIN requested SUBTASKS=YES but IWMELEAV was issued with the primary address space not equal to the home address space. No processing of subtasks was done. The Leave processing completed successfully because the current dispatchable work unit does not have residual subtasks propagated to the enclave which are still associated with the enclave. **Action**: Issue IWMELEAV with PASN=HASN if you need the subtasks processed.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0801	**Equate Symbol**: IwmRsnCodeSrbMode **Meaning**: Caller is in SRB mode. **Action**: Avoid requesting this function while in SRB mode.
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: Caller is locked. **Action**: Avoid requesting this function while locked.

Table 33. Return and Reason Codes for the IWMELEAV Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx080B	**Equate Symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list. **Action**: Check for possible storage overlay.
8	xxxx0810	**Equate Symbol**: IwmRsnCodeEutFrr **Meaning**: Caller has EUT FRR established. **Action**: Avoid requesting this function with an EUT FRR set.
8	xxxx0823	**Equate Symbol**: IwmRsnCodeDatoff **Meaning**: Caller invoked service while DATOFF **Action**: Avoid requesting this function in this environment.
8	xxxx0824	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: Caller invoked service but was in 24 bit addressing mode. **Action**: Request this function only when you are in 31 bit addressing mode.
8	xxxx0825	**Equate Symbol**: IwmRsnCodeAscModeNotPrimary **Meaning**: Caller invoked service but was not DAT on Primary ASC mode. **Action**: Avoid requesting this function in this environment.
8	xxxx0826	**Equate Symbol**: IwmRsnCodeTaskTerm **Meaning**: Caller invoked service while task termination is in progress for the current TCB. **Action**: Avoid requesting this function while task termination is in progress.
8	xxxx0828	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: Version number in parameter list is not valid or version length field is not valid. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx083A	**Equate Symbol**: IwmRsnCodeBadEnclave **Meaning**: Enclave token does not pass verification. **Action**: Check for possible storage overlay of the enclave token.
8	xxxx0845	**Equate Symbol**: IwmRsnCodeWrongEnclave **Meaning**: The current dispatchable workunit is not associated with the input enclave. **Action**: Check for possible storage overlay of the enclave token.

Table 33. Return and Reason Codes for the IWMELEAV Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0850	**Equate Symbol**: IwmRsnCodeBeginEnvOutstanding **Meaning**: Current dispatchable workunit is operating under an outstanding Begin environment, enclave leave is not allowed. IWM4STEN is the required operation. **Action**: Avoid requesting this function in this environment.
8	xxxx0858	**Equate Symbol**: IwmRsnCodeNotEjoinedTcb **Meaning**: The current dispatchable workunit did not issue enclave Join, but only inherited enclave attribute from mother TCB. **Action**: Avoid requesting this function in this environment.
8	xxxx0859	**Equate Symbol**: IwmRsnCodeEnclaveSubTaskExists **Meaning**: The current dispatchable workunit has residual subtasks propagated to the enclave which are still associated with the enclave. Either the join (IWMEJOIN) of this work unit to the enclave did not specify SUBTASKS=YES or the join (IWMEJOIN) of this work unit to the enclave did specify SUBTASKS=YES, but the IWMELEAV invocation was not made with PASN=HASN. **Action**: Avoid requesting this function in this environment.
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error. **Action**: Contact your system programmer.

Example

To cause the current task to leave its Enclave environment:

```
        IWMELEAV 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
```

IWMEQTME — Query enclave CPU time

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:

Minimum authorization:	Supervisor state. Any PSW key.
Dispatchable unit mode:	Task, if `CURRENT_DISP=YES`; otherwise, task or SRB
Cross memory mode:	Non-XMEM or XMEM. Any PASN, HASN, SASN.
AMODE:	31-bit
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts if `CURRENT_DISP=YES`, otherwise enabled or disabled for I/O and external interrupts
Locks:	No locks are required.
Control parameters:	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 unique to certain versions. See the PLISTVER parameter description.

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
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

```
►►──┬──────┬──IWMEQTME──┬────────────────────┬──┬──────────────────────────────┬──────────►
    └─name─┘            └─CPUTIME=cputime────┘  └─,ZIIPQUALTIME=ziipqualtime───┘

►──┬──────────────────────┬──┬─────────────────────────────┬─────────────────────────────►
   └─,ZIIPTIME=ziiptime───┘  └─,ZIIPONCPTIME=ziiponcptime──┘

►──┬──────────────────────┬──┬─────────────────────────────┬─────────────────────────────►
   └─,ZAAPTIME=zaaptime───┘  └─,ZAAPONCPTIME=zaaponcptime──┘

                                  ┌─,CURRENT_DISP=NO──┐
►──┬──────────────────────────────┼───────────────────┼──┬──────────────────────┬─────────►
   └─,ZAAPNFACTOR=zaapnfactor─────┘ └─,CURRENT_DISP=YES─┘  └─,RETCODE=retcode─────┘

                              ┌─,PLISTVER=IMPLIED_VERSION─┐
►──┬──────────────────────┬───┼───────────────────────────┼──────────────────────────────►
   └─,RSNCODE=rsncode─────┘   ├─,PLISTVER=MAX─────────────┤
                              ├─,PLISTVER=0───────────────┤
                              ├─,PLISTVER=1───────────────┤
                              └─,PLISTVER=2───────────────┘

   ┌─,MF=S──────────────────────────────────────┐
►──┼────────────────────────────────────────────┼────────────────────────────────────────►◄
   │                           ┌─,0D───┐        │
   ├─,MF=(L─,list addr─────────┼───────┼─)──────┤
   │                           └─,attr─┘        │
   │                          ┌─,COMPLETE─┐     │
   └─,MF=(E─,list addr────────┴───────────┴─)───┘
```

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 TCB and SRB time for the enclave that is associated with the current dispatchable workunit. 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.

,CURRENT_DISP=NO
,CURRENT_DISP=YES
An optional parameter indicating whether to call the dispatcher before querying the enclave processor times. Enclave processor times are only accumulated when an enclave loses the processor. So the processor times are the total accumulated times for the enclave up to the point when it was dispatched for the last time. Because the enclave being queried by the service is currently active on a processor, the processor times are not accurate; all processor times since the last dispatch of the enclave are missing. To make the enclave processor times returned by the service more accurate, the dispatcher can optionally be called to update the accumulated processor times before they are returned by the service.

This option provides benefit mostly for single-work-unit enclaves. For enclaves with multiple work units (tasks or SRBs) running in parallel, the benefit will be marginal because the call to the dispatcher will only update the times for the current (calling) workunit.

The default is CURRENT_DISP=NO.

,CURRENT_DISP=NO
The enclave processor times are queried without calling the dispatcher before. Which means that processor times since the last dispatch of the enclave are missing, leading to slightly too small results. In return invocation of the service is less costly, and has less prerequisites in the environment.

,CURRENT_DISP=YES
The enclave processor times are queried after calling the dispatcher. Which means that all processor times for the work unit up to invoking the service are returned. The price for getting more exact values is more cycles for invoking the service, and more restricted environments in which it can be invoked.

,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:
  - ZIIPONCPTIME
  - ZIIPQUALTIME
  - ZIIPTIME

- **2**, which supports both the following parameters and those from version 0 and 1:
  - CURRENT_DISP
  - ZAAPONCPTIME
  - ZAAPNFACTOR
  - ZAAPTIME

To code: Specify one of the following:
- IMPLIED_VERSION
- MAX
- A decimal value of 0, 1, or 2

,RETCODE=*retcode*
An optional output parameter into which the return code is to be copied from GPR 15. If you specify 15, GPR15, REG15, or R15 (within or without parentheses), the value will be left in GPR 15.

To code: Specify the RS-type address of a fullword field, or register (2)-(12) or (15), (GPR15), (REG15), or (R15).

,RSNCODE=*rsncode*
An optional output parameter into which the reason code is to be copied from GPR 0. If you specify 0, 00, GPR0, GPR00, REG0, REG00, or R0 (within or without parentheses), the value will be left in GPR 0.

To code: Specify the RS-type address of a fullword field, or register (0) or (2)-(12), (00), (GPR0), (GPR00), REG0), (REG00), or (R0).

,ZAAPNFACTOR=*zaapnfactor*
An optional output parameter, which will contain the normalization factor for application assist processors (zAAPs). If zAAPs are running at a different speed, multiply zAAP 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.

,ZAAPONCPTIME=*zaaponcptime*
An optional output parameter, which will contain the total accumulated time spent on a regular CP for application assist processor (zAAP) eligible work for the enclave that is associated with the current dispatchable work unit. The time will be in TOD clock format, normalized to the regular processor speed.

To code: Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

,ZAAPTIME=*zaaptime*
An optional output parameter, which will contain the total accumulated application assist processor (zAAP) time for the enclave that is associated with the current work unit. 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. The 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.

,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 workunit. 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 workunit was qualified to run an an integrated information processor (zIIP). The 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.

,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 workunit. 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

Invoking IWMEQTME with parameter CURRENT_DISP=YES while being disabled for I/O and external interrupts results in ABEND 05D-08:

Explanation: A program issued a CALLDISP macro that is not valid. A hexadecimal reason code in register 15 explains the error: 08 The macro specified FRRSTK=SAVE while the program holds a lock, or the macro specified FRRSTK=NOSAVE while the program holds a lock other than the LOCAL lock or the cross memory local (CML) lock.

Invoking IWMEQTME with parameter CURRENT_DISP=YES while being in SRB mode results in ABEND 05D-10:

Explanation: A program issued a CALLDISP macro that is not valid. A hexadecimal reason code in register 15 explains the error: 10 The program was not in task control block (TCB) mode.

Return codes 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 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 34. Return and Reason Codes for the IWMEQTME Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.
4	xxxx041C	**Equate Symbol**: IwmRsnCodeNotEnclave **Meaning**: The current dispatchable workunit is not associated with an Enclave. **Action**: None required.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0828	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: Version number in parameter list is not valid. **Action**: Check for possible storage overlay of the parameter list.

Examples

None.

IWMEREG — Register a WLM enclave

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:

Minimum authorization:	Supervisor state or program key mask (PKM) allowing keys 0-7.
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31-bit
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks may be held.
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, 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 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

The syntax of the IWMEREG macro is as follows:

```
►►─┬──────┬─IWMEREG──REGTOKEN=regtoken──,SUBSYS=subsys──,SUBSYSNM=subsysnm──────────────►
   └─name─┘

                                                        ┌─,OWNER=HOME────┐
►─┬─────────────────────────────────┬─,ETOKEN=etoken──┼────────────────┼─────────────────►
  └─,SUBSYSREQUEST=subsysrequest─┘                    └─,OWNER=PRIMARY─┘
```

```
                                                      ┌─,PLISTVER=IMPLIED_VERSION─┐
►─┬──────────────────────┬─┬──────────────────────┬───┼───────────────────────────┼────────────────────►
  └─,RETCODE=retcode─────┘ └─,RSNCODE=rsncode─────┘   ├─,PLISTVER=MAX─────────────┤
                                                      ├─,PLISTVER=0───────────────┤
                                                      └─,PLISTVER=1───────────────┘

  ┌─,MF=S────────────────────────────────────┐
►─┼──────────────────────────────────────────┼─────────────────────────────────────────────────────►◄
  │                            ┌─,0D───┐     │
  ├─,MF=(L─,list addr──────────┼───────┼─)───┤
  │                            └─,attr─┘     │
  │                            ┌─,COMPLETE─┐ │
  └─,MF=(E─,list addr──────────┴───────────┴─)┘
```

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.

,OWNER=HOME
,OWNER=PRIMARY
An optional parameter, for internal use only. The default is OWNER=HOME.

,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 the following parameter and those from version 0:

  OWNER

To code: Specify one of the following:
- IMPLIED_VERSION
- MAX
- A decimal value of 0, or 1

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 codes 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	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: The caller is locked. **Action**: Avoid requesting this function while locked.
8	xxxx080B	**Equate Symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list. **Action**: Check for possible storage overlay.

Table 35. Return and Reason Codes for the IWMEREG Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx0824	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: The caller invoked the service but was in 24-bit addressing mode. **Action**: Request this function only when you are in 31-bit addressing mode.
8	xxxx0828	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: Version number in parameter list is not valid. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx083A	**Equate Symbol**: IwmRsnCodeBadEnclave **Meaning**: Enclave token does not pass verification. **Action**: Check for possible storage overlay of the enclave token, or asynchronous events which may have deleted the enclave.
8	xxxx0882	**Equate Symbol**: IwmRsnCodeTooManyRegistrations **Meaning**: There are too many concurrent registrations requested. **Action**: There is a resource shortage. The function may work successfully at a later time.
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error. **Action**: 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
```

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. See "Restrictions" for details.

Environment

The requirements for the caller are:

Minimum authorization:	Supervisor state or program key mask (PKM) allowing keys 0-7.
Dispatchable unit mode:	Task or SRB
Cross memory mode:	PASN=HASN=SASN
AMODE:	31-bit
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks may be held.
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, IWMRSNCODE_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. The caller cannot reset a dependent enclave. A dependent enclave can only be reset by resetting the address space that owns the enclave.
3. The caller cannot reset a work-dependent enclave. Such an enclave can only be reset by resetting the owning independent 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 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

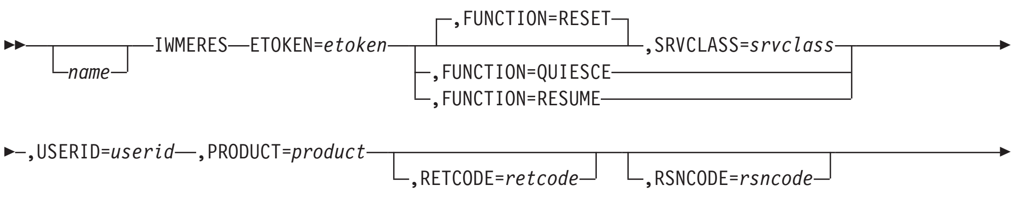
Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

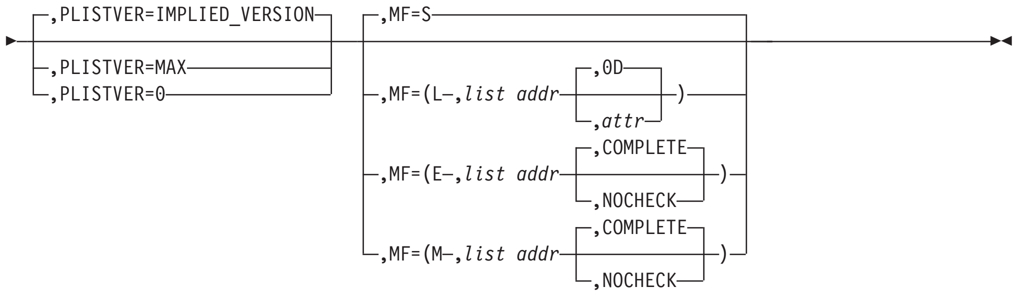
Performance implications

None.

Syntax

The syntax of the IWMERES macro is as follows:





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*,**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 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=*srvclass*
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 user ID is included in the SMF 90 subtype 30 record created by IWMERES. If there is no user ID 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 codes 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.

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	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.
4	xxxx043B	**Equate Symbol**: IwmRsnCodeIsQuiesced **Meaning**: 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. **Action**: Retry the request when no address spaces serving the enclave is quiesced.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: The caller is locked. **Action**: Avoid requesting this function while locked.
8	xxxx080B	**Equate Symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list. **Action**: Check for possible storage overlay.
8	xxxx0824	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: The caller invoked the service but was in 24-bit addressing mode. **Action**: Request this function only when you are in 31-bit addressing mode.
8	xxxx0825	**Equate Symbol**: IwmRsnCodeAscModeNotPrimary **Meaning**: The caller invoked the service but was not in primary ASC mode. **Action**: Avoid requesting this function in this environment.
8	xxxx0827	**Equate Symbol**: IwmRsnCodeRsvdNot0 **Meaning**: Reserved field in parameter list was non-zero. **Action**: 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	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: Version number in parameter list is not valid. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx0829	**Equate Symbol**: IwmRsnCodeBadOptions **Meaning**: Parameter list omits required parameters. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx083A	**Equate Symbol**: IwmRsnCodeBadEnclave **Meaning**: Enclave token does not pass verification. **Action**: Check for possible storage overlay of the enclave token, or asynchronous events which may have deleted the enclave.
8	xxxx0841	**Equate Symbol**: IwmRsnCodeXMemMode **Meaning**: Caller is in cross memory mode. **Action**: Invoke the function in non-cross memory mode.
8	xxxx0872	**Equate Symbol**: IwmRsnCodeForeignEnclave **Meaning**: The requested service is not supported for a foreign enclave. This reason code is returned for independent foreign enclaves only. **Action**: All participants of a multisystem enclave can only be reset together by resetting the original enclave on the originating system.
8	xxxx0885	**Equate Symbol**: IwmRsnCodeDependentEnclave **Meaning**: The requested service is not supported for a dependent enclave. This reason code is returned for both, an original and a foreign dependent enclave. **Action**: A dependent enclave cannot be reset directly. It can only be reset by resetting its owning address space.
8	xxxx08B7	**Equate Symbol**: IwmRsnCodeWorkDepEnclave **Meaning**: The requested service is not supported for a 'work-dependent' enclave. **Action**: A 'work-dependent' enclave cannot be reset directly. It can only be reset by resetting its owning independent enclave.
C	—	**Equate Symbol**: IwmRetCodeEnvError **Meaning**: Environmental error.
C	xxxx0C28	**Equate Symbol**: IwmRsnCodeBadServiceClass **Meaning**: The input service class name is not defined in the active workload manager policy. **Action**: Record or report the error if appropriate.
C	xxxx0C2E	**Equate Symbol**: IwmRsnCodeWrongMode **Meaning**: Reserved.

Table 36. Return and Reason Codes for the IWMERES Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
C	xxxx0C32	**Equate Symbol**: IwmRsnCodeNotEligibleForSrvClass **Meaning**: 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. **Action**: Record or report the error if appropriate.
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error. **Action**: 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
```

IWMESQRY — Query enclave state

The purpose of this service is to query whether or not the current dispatchable work unit is associated with an enclave. The output is either the enclave token or the STOKEN of the address space associated with the caller's work unit.

Environment

The requirements for the caller are:

Minimum authorization:	Supervisor state. Any PSW key.
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Non-XMEM or XMEM. Any PASN, HASN, SASN.
AMODE:	31-bit
ASC mode:	Primary
Interrupt status:	Enabled or disabled for I/O and external interrupts
Locks:	No locks are required.
Control parameters:	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. The caller is responsible for the error recovery.

Restrictions

This macro supports multiple versions. Some keywords are unique to certain versions. See the PLISTVER parameter description.

Input register information

Before issuing the IWMESQRY 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

The syntax of the IWMESQRY macro is as follows:

```
►►──┬──────┬──IWMESQRY──┬─ETOKEN=etoken─┬──┬──────────────────────────┬──────────────►
    └─name─┘            └─TOKEN=token───┘  └─,IMPORTANCE=importance───┘

                                            ┌─,PLISTVER=IMPLIED_VERSION─┐
►──┬────────────────────┬──┬────────────────────┬──┼───────────────────────────┼──────►
   └─,RETCODE=retcode───┘  └─,RSNCODE=rsncode───┘  ├─,PLISTVER=MAX─────────────┤
                                                   ├─,PLISTVER=0───────────────┤
                                                   └─,PLISTVER=1───────────────┘

   ┌─,MF=S──────────────────────────────┐
►──┼────────────────────────────────────┼─────────────────────────────────────────►◄
   │                       ┌─,0D───┐    │
   ├─,MF=(L─,list addr─────┼───────┼─)──┤
   │                       └─,attr─┘    │
   │                       ┌─,COMPLETE─┐│
   └─,MF=(E─,list addr─────┴───────────┴─)┘
```

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 work unit 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.

,IMPORTANCE=*importance*

An optional output parameter that will receive the importance value of the service class to which the unit of work is classified.

To code: Specify the RS-type address, or address in register (2)-(12), of a halfword 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**, which supports all parameters except those specifically referenced in higher versions.
- **1**, which supports the following parameter and those from version 0:

  IMPORTANCE

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. If you specify 15, GPR15, REG15, or R15 (with or without parentheses), the value will be left in GPR 15.

To code: Specify the RS-type address of a fullword field, or register (2)-(12) or (15), (GPR15), (REG15), or (R15).

,RSNCODE=*rsncode*
An optional output parameter into which the reason code is to be copied from GPR 0. If you specify 0, 00, GPR0, GPR00, REG0, REG00, or R0 (with or without parentheses), the value will be left in GPR 0.

To code: Specify the RS-type address of a fullword field, or register (2)-(12), (00), (GPR0), (GPR00), REG0), (REG00), or (R0).

,TOKEN=*token*
A required output parameter which will receive either the enclave token if the current dispatchable work unit is associated with an enclave (as indicated by return code 0), or the STOKEN of the address space the work unit 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 codes 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 37. Return and Reason Codes for the IWMESQRY Macro

Return Code	Reason Code	Equate Symbol, Meaning, and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.
4	xxxx041C	**Equate Symbol**: IwmRsnCodeNotEnclave **Meaning**: The current dispatchable work unit is not associated with an enclave. The returned token is the STOKEN of the associated address space. **Action**: None required.

Example

To query whether the current dispatchable work unit is associated with an enclave or an address space, specify:

```
         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
```

IWMESTOP — Stop 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:

Minimum authorization:	Supervisor state or program key mask (PKM) allowing keys 0-7
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31- and 64-bit. If in 64-bit addressing mode, code SYSSTATE AMODE64=YES before invoking this macro.
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks may be held.
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 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 EWLMMODE=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 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

The syntax of the IWMESTOP macro is as follows:

```
►►──┬──────┬──IWMESTOP──ETOKEN=etoken──WORKREQ_HDL=workreq_hdl──────────────►
    └─name─┘

   ┌─,WORKREQ_STA=0───────────┐  ┌─,EWLMMODE=NORMAL──────────┐
►──┴─,WORKREQ_STA=workreq_sta─┴──┴─,EWLMMODE=EXPLICIT_SINGLE─┴──┬──────────────────┬──►
                                                                └─,RETCODE=retcode─┘

                          ┌─,PLISTVER=IMPLIED_VERSION─┐
►──┬──────────────────┬───┼───────────────────────────┼─────────────────────►
   └─,RSNCODE=rsncode─┘   ├─,PLISTVER=MAX─────────────┤
                          └─,PLISTVER=0───────────────┘

   ┌─,MF=S────────────────────────────────────┐
►──┼──────────────────────────────────────────┼─────────────────────────────►◄
   │                       ┌─,0D────┐         │
   ├─,MF=(L─,list addr─────┼────────┼──)──────┤
   │                       └─,attr──┘         │
   │                       ┌─,COMPLETE─┐      │
   └─,MF=(E─,list addr─────┴───────────┴──)───┘
```

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 adressable 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

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 codes 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 38. Return and Reason Codes for the IWMESTOP Macro

Return Code	Reason Code	Equate Symbol, Meaning, and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.

Table 38. Return and Reason Codes for the IWMESTOP Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: The caller is locked. **Action**: Avoid requesting this function while locked.
8	xxxx080B	**Equate Symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list. **Action**: Check for possible storage overlay.
8	xxxx0824	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: The caller invoked the service but was in 24-bit addressing mode. **Action**: Request this function only when you are in 31-bit addressing mode.
8	xxxx0828	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: Version number in parameter list is not valid. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx083A	**Equate Symbol**: IwmRsnCodeBadEnclave **Meaning**: Enclave token does not pass verification. **Action**: Check for possible storage overlay of the enclave token, or for asynchronous events which may have deleted the enclave.
8	xxxx0885	**Equate Symbol**: IwmRsnCodeDependentEnclave **Meaning**: This service is not available for dependent enclaves. **Action**: Avoid requesting this function for dependent enclaves.
8	xxxx0895	**Equate Symbol**: IwmRsnCodeEWLMServNotEnabled **Meaning**: This service is not enabled because the caller invoked the IWM4CON (IWMCONN) service with EWLM=NO. **Action**: Ensure that EWLM=YES is specified on the IWM4CON request to enable this service.
8	xxxx0896	**Equate Symbol**: IwmRsnCodeBadWorkReqHandle **Meaning**: Work request handle is not associated with passed enclave. **Action**: Check the specification of the WORKREQ_HDL parameter.

Table 38. Return and Reason Codes for the IWMESTOP Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx0897	**Equate Symbol**: IwmRsnCodeTranStatusInvalid **Meaning**: Passed work request completion status is not valid. **Action**: Check the EWLM ARM interface specification for valid completion status values.
8	xxxx08A0	**Equate Symbol**: IwmRsnCodeNotExplicitSingle **Meaning**: 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. **Action**: 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).
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error. **Action**: No action required. The function may be successful if invoked again.

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
```

IWMESTRT — Start 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 the WLM 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:

Minimum authorization:	Supervisor state or program key mask (PKM) allowing keys 0-7.
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31- or 64-bit. If in 64-bit addressing mode, code `SYSSTATE AMODE64=YES` before invoking this macro.
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks may be held.
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 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 `EWLMMODE=EXPLICIT_SINGLE` is specified, some 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 save area 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.

- All data in the optional classify parameter list (`CLSFY=`*`xxx`*) are 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.

Restrictions

None

Input register information

If `EWLMMODE=EXPLICIT_SINGLE` is specified, the caller must provide a standard 72-Byte savearea pointed to by R13.

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

```
                                          ┌─,CLSFY=NO_CLSFY─┐
►►──┬──────┬──b──IWMESTRT──b──ETOKEN=etoken──┴─,CLSFY=clsfy────┴──────────────────────────────►
    └─name─┘

                          ┌─,ARRIVALTIME=NO_ARRIVALTIME─┐
►─,WORKREQ_HDL=workreq_hdl──┴─,ARRIVALTIME=arrivaltime────┴──────────────────────────────────►

  ┌─,EWLMMODE=NORMAL──────────┐
►─┴─,EWLMMODE=EXPLICIT_SINGLE─┴──┬───────────────────┬──┬───────────────────┬─────────────────►
                                 └─,RETCODE=retcode──┘  └─,RSNCODE=rsncode──┘

  ┌─,PLISTVER=IMPLIED_VERSION─┐  ┌─,MF=S───────────────────────────────────┐
►─┼─,PLISTVER=MAX─────────────┼──┼─────────────────────────────────────────┼──────────────────►◄
  └─,PLISTVER=0───────────────┘  │                      ┌─,0D───┐          │
                                 ├─,MF=(L─,list addr──┴─,attr─┴─)────────┤
                                 │                      ┌─,COMPLETE─┐      │
                                 └─,MF=(E─,list addr──┴───────────┴─)────┘
```

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. 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 or IWM4CLSY NOTE that this name is the data area name, not its pointer. IWM4CLSY or 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.

  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 or IWM4CLSY:

  - SUBSYSPM is limited to 255 bytes
  - COLLECTION is limited to 18 bytes
  - CORRELATION is limited to 12 bytes

Note: If EWLMMODE=EXPLICIT_SINGLE is specified, all data in the optional classify parameter list (CLSFY=xxx) are 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. The default is NO_CLSFY. indicates that no classify parameter list is passed.

To code: Specify the RS-type address, or address in register (2)-(12), of a character field.

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 parameter that indicates, if the parameter EXPLICIT_SINGLE has been specified on the IWM4ECRE(or IWMECREA) call. The EWLMMODE parameter has to be omitted (or the default value NORMAL must be specified), if the enclave was not created with the ESTRT=EXPLICIT_SINGLE option and vice versa. Otherwise the results may be unpredictable. The default is EWLMMODE=NORMAL.

,EWLMMODE=NORMAL

indicates that the enclave was created with the ESTRT=EXPLICIT or ESTRT=IMPLIED option.

,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 also provide a standard 72-Byte savearea (addressable in AMODE 31) pointed to by GPR13. Moreover the parameter list and the classification parameters must also be adressable 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
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. If you specify 15, GPR15, REG15, or R15 (within or without parentheses), the value will be left in GPR 15.

To code: Specify the RS-type address of a fullword field, or register (2)-(12) or (15), (GPR15), (REG15), or (R15).

,RSNCODE=*rsncode*
An optional output parameter into which the reason code is to be copied from GPR 0. If you specify 0, 00, GPR0, GPR00, REG0, REG00, or R0 (within or without parentheses), the value will be left in GPR 0.

To code: Specify the RS-type address of a fullword field, or register (0) or (2)-(12), (00), (GPR0), (GPR00), REG0), (REG00), or (R0).

,WORKREQ_HDL=*workreq_hdl*
A required 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 codes and reason codes

When the IWMESTRT 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 39. Return and Reason Codes for the IWMESTRT Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: Caller is locked. **Action**: Avoid requesting this function while locked.
8	xxxx080B	**Equate Symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list. **Action**: Check for possible storage overlay.
8	xxxx0824	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: Caller invoked service but was in 24 bit addressing mode. **Action**: Request this function only when you are in 31 bit addressing mode.
8	xxxx0828	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: Version number in parameter list is not valid. **Action**: Check for possible storage overlay of the parameter list.

Table 39. Return and Reason Codes for the IWMESTRT Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0838	**Equate Symbol**: IwmRsnCodeClsfyAreaTooBig **Meaning**: Input area associated with classification information is larger than supported. **Action**: Invoke the function with an area of the proper size. Check for possible storage overlay.
8	xxxx0839	**Equate Symbol**: IwmRsnCodeClsfyPlTooSmall **Meaning**: Input Classify parameter list is too small. **Action**: Invoke the function with an area of the proper size. Check for possible storage overlay.
8	xxxx083A	**Equate Symbol**: IwmRsnCodeBadEnclave **Meaning**: Enclave token does not pass verification. **Action**: Check for possible storage overlay of the enclave token, or asynchronous events which may have deleted the enclave.
8	xxxx0885	**Equate Symbol**: IwmRsnCodeDependentEnclave **Meaning**: Service is not available for dependent enclaves. **Action**: Avoid requesting this function for dependent enclaves.
8	xxxx0893	**Equate Symbol**: IwmRsnCodeMissingEWLMCorr **Meaning**: Passed classification information must contain an EWLM correlator (EWLM_CORR). **Action**: Provide an EWLM correlator (EWLM_CORR) within the passed classification information.
8	xxxx0894	**Equate Symbol**: IwmRsnCodeInvalidEWLMCorr **Meaning**: Passed classification information contains an EWLM correlator (EWLM_CORR) that does not pass validity checking or that is not associated with the enclave. **Action**: Check the specification of the EWLM correlator in the classification information.
8	xxxx0895	**Equate Symbol**: IwmRsnCodeEWLMServNotEnabled **Meaning**: Service is not enabled because caller invoked the IWMCONN service with EWLM=NO. **Action**: Ensure that EWLM=YES is specified on the IWMCONN request to enable this service.
8	xxxx089F	**Equate Symbol**: IwmRsnCodeMoreThanOneStart **Meaning**: The service has been invoked with the EWLMMODE=EXPLICIT_SINGLE option, but a workrequest is already active for the enclave. For EWLMMODE=EXPLICIT_SINGLE only one workrequest may be active at a given point in time. **Action**: Ensure that the IWMESTRT and IWMESTOP invocations are correctly paired and that only one workrequest is active.

Table 39. Return and Reason Codes for the IWMESTRT Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx08A0	**Equate Symbol**: IwmRsnCodeNotExplicitSingle **Meaning**: 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. **Action**: If ESTRT=EXPLICIT_SINGLE was coded on the IWM4ECRE or IWMECREA call, EWLMMODE=EXPLICIT_SINGLE must also specified on the IWMESTRT call. If ESTRT=EXPLICIT OR ESTRT=IMPLIED was coded on the IWM4ECRE or IWMECREA call, EWLMMODE=NORMAL must be specified on the IWMESTRT call (or the EWLMMODE parameter can be omitted).
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error. **Action**: No action required. The function may be successful if invoked again.

Example

```
* Before calling IWMESTRT initialize classification
* attributes in CLFSY
*
        IWMESTRT ETOKEN=ENCTKN,CLSFY=CLFSY,
                 WORKREQ_HDL=WRHANDLE,ARRIVALTIME=ATIME
*
* Storage areas
*
        IWM4CLSY PLISTVER=MAX,MF=(L,CLFSY,0D)
ENCTKN   DS    CL8             Enclave token
ATIME    DS    CL8             Arrival time
WRHANDLE DS    CL8             Work Request Handle
```

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:

Minimum authorization:	Supervisor state or program key mask (PKM) allowing keys 0-7
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31- or 64-bit. If in 64-bit addressing mode, code `SYSSTATE AMODE64=YES` before invoking this macro.
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks may be held.
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 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 EWLMMODE=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

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

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

The syntax of the IWMEUBLK macro is as follows:

```
►►─┬──────┬─IWMEUBLK──WORKREQ_HDL=workreq_hdl──,BLOCK_HDL=block_hdl──────────────►
   └─name─┘

   ┌─,EWLMMODE=NORMAL──────────┐  ┌─,ETOKEN=NO_ETOKEN─┐
►──┼───────────────────────────┼──┼───────────────────┼──┬──────────────────┬─────►
   └─,EWLMMODE=EXPLICIT_SINGLE─┘  └─,ETOKEN=etoken────┘  └─,RETCODE=retcode─┘

                          ┌─,PLISTVER=IMPLIED_VERSION─┐
►──┬──────────────────┬───┼───────────────────────────┼───────────────────────────►
   └─,RSNCODE=rsncode─┘   ├─,PLISTVER=MAX─────────────┤
                          ├─,PLISTVER=0───────────────┤
                          └─,PLISTVER=1───────────────┘

   ┌─,MF=S────────────────────────────────────┐
►──┼──────────────────────────────────────────┼───────────────────────────────────►◄
   │                       ┌─,0D────┐         │
   ├─,MF=(L─,list addr─────┼────────┼─)───────┤
   │                       └─,attr──┘         │
   │                       ┌─,COMPLETE─┐      │
   └─,MF=(E─,list addr─────┴───────────┴─)────┘
```

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=etoken
  : required, if the option EWLMMODE=EXPLICIT_SINGLE is specified.

  ETOKEN=NOTOKEN
  : 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 adressable 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**, which supports all parameters except those specifically referenced in higher versions.
- 1, which supports 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 codes 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	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: The caller is locked. **Action**: Avoid requesting this function while locked.
8	xxxx080B	**Equate Symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list. **Action**: Check for possible storage overlay.
8	xxxx0824	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: The caller invoked the service but was in 24-bit addressing mode. **Action**: Request this function only when you are in 31-bit addressing mode.
8	xxxx0828	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: Version number in parameter list is not valid. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx083A	**Equate Symbol**: IwmRsnCodeBadEnclave **Meaning**: EWLMMODE=EXPLICIT_SINGLE was specified and the required enclave token does not pass verification. **Action**: Check for possible storage overlay of the enclave token, or for asynchronous events which may have deleted the enclave.
8	xxxx0895	**Equate Symbol**: IwmRsnCodeEWLMServNotEnabled **Meaning**: The service is not enabled because the caller invoked the IWM4CON (IWMCONN) service with EWLM=NO. **Action**: Ensure that EWLM=YES is specified on the IWM4CON request to enable this service.
8	xxxx0896	**Equate Symbol**: IwmRsnCodeBadWorkReqHandle **Meaning**: Work request handle is invalid. **Action**: Check the specification of the WORKREQ_HDL parameter.

Table 40. Return and Reason Codes for the IWMEUBLK Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx0898	**Equate Symbol**: IwmRsnCodeBadBlockHandle **Meaning**: Block handle is invalid. **Action**: Check the specification of the BLOCK_HDL parameter.
8	xxxx08A0	**Equate Symbol**: IwmRsnCodeNotExplicitSingle **Meaning**: 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. **Action**: 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).
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error. **Action**: No action required. The function may be successful if invoked again.

Examples

```
         IWMEUBLK WORKREQ_HDL=WRHANDLE,
                 BLOCK_HDL=BKHANDLE
*
* Storage areas
*
WRHANDLE DS    CL8             Work Request Handle
BKHANDLE DS    CL8             Work Request Block Handle
```

IWMEXPT — Export a WLM enclave

The IWMEXPT 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 IWMUEXPT 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:

Minimum authorization:	Supervisor state or program key mask (PKM) allowing keys 0-7.
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any HASN, any SASN. PASN must be the address space which connected to WLM.
AMODE:	31-bit
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks may be held.
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, IWMRSNCODE_MASK_CONST defined in IWMYCON, may be used for this purpose.

Restrictions

It is not supported to export work-dependent enclaves.

Input register information

Before issuing the IWMEXPT 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

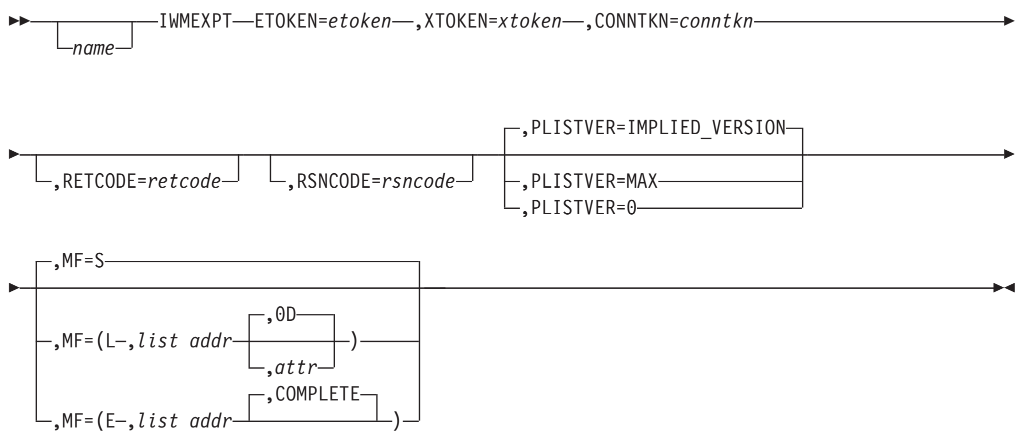
Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

Performance implications

None.

Syntax

The syntax of the IWMEXPT macro is as follows:



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:

- **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 codes 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	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.
4	xxxx0433	**Equate Symbol**: IwmRsnCodeEncAlreadyExported **Meaning**: The input enclave was imported from another system. It cannot be exported by this system. **Action**: 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.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: Caller is locked. **Action**: Avoid requesting this function while locked.
8	xxxx080B	**Equate Symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list. **Action**: Check for possible storage overlay.
8	xxxx0821	**Equate Symbol**: IwmRsnCodeBadConn **Meaning**: Input connect token does not pass validity checking. **Action**: 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	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: The caller invoked the service but was in 24-bit addressing mode. **Action**: Request this function only when you are in 31-bit addressing mode.
8	xxxx0825	**Equate Symbol**: IwmRsnCodeAscModeNotPrimary **Meaning**: The caller invoked the service but was not in primary ASC mode. **Action**: Avoid requesting this function in this environment.

Table 41. Return and Reason Codes for the IWMEXPT Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx0827	**Equate Symbol**: IwmRsnCodeRsvdNot0 **Meaning**: Reserved field in parameter list was non-zero. **Action**: Check for use of keywords that are not supported by the z/OS release on which the program is running.
8	xxxx0828	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: Version number in parameter list is not valid. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx083A	**Equate Symbol**: IwmRsnCodeBadEnclave **Meaning**: Enclave token does not pass verification. **Action**: Check for possible storage overlay of the enclave token, or asynchronous events which may have deleted the enclave.
8	xxxx0840	**Equate Symbol**: IwmRsnCodeServiceNotEnabled **Meaning**: Caller's connection is not enabled for this service. **Action**: Make sure that EXPTIMPT=YES is specified on the IWM4CON macro invocation.
8	xxxx0842	**Equate Symbol**: IwmRsnCodeNoWLMConnect **Meaning**: Caller's primary address space is not connected to WLM. **Action**: Invoke the IWM4CON macro before invoking this macro.
8	xxxx0884	**Equate Symbol**IwmRsnCodeEnclaveDefEx **Meaning**: Enclave is marked Execution Start Time to be deferred. **Action**: Such an enclave cannot be exported.
8	xxxx08B7	**Equate Symbol**: IwmRsnCodeWorkDepEnclave **Meaning**: The requested service is not supported for a work-dependent enclave. **Action**: Do not try to export work-dependent enclaves.
C	—	**Equate Symbol**: IwmRetCodeEnvError **Meaning**: Environmental error.
C	xxxx0C01	**Equate Symbol**: IwmRsnCodeNoStg **Meaning**: Virtual storage is not available for the request. **Action**: Process the unit-of-work on the local system or fail the unit-of-work, whichever is appropriate.
C	xxxx0C2F	**Equate Symbol**: IwmRsnCodeSystemSpace **Meaning**: The enclave is owned by a system (i.e. limited-function) address space. Exporting such an enclave is not supported. **Action**: 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	xxxx0C37	**Equate Symbol**: IwmRsnCodeStructureFull **Meaning**: The coupling facility structure is full. **Action**: Process the unit-of-work on the local system or fail the unit-of-work, whichever is appropriate.
C	xxxx0C36	**Equate Symbol**: IwmRsnCodeStructureUnavailable **Meaning**: WLM does not have access to its coupling facility structure. **Action**: 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.
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error. **Action**: Contact your system programmer.

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
```

IWMGCORF — Get correlator flags

The IWMGCORF service allows to check whether or not certain Application Response Measurement (ARM) flags in a provided EWLM correlator are set. These flags are contained in byte 3 of the ARM correlator format.

Environment

The requirements for the caller are:

Minimum authorization:	Problem state or supervisor state.
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31- or 64-bit. If in 64-bit addressing mode, code `SYSSTATE AMODE64=YES` before invoking this macro.
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No restriction.
Control parameters:	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 bits 0-31 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 IWMGCORF 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**
0-1	Used as work registers by the system
2-13	Unchanged

14-15 Used as work registers by the system

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

The syntax of the IWMGCORF macro is as follows:

```
►►──┬──────┬──IWMGCORF──EWLM_CORR=ewlm_corr──┬───────────────────────┬──────────────►
    └─name─┘                                  └─,ASYNC_FLAG=async_flag─┘

►──┬─────────────────────────┬──,MF=(M─,addr──┬─,COMPLETE─┬─)──────────────────────►◄
   └─,INDEP_FLAG=indep_flag──┘                 └─,NOCHECK──┘
```

Parameters

The parameters are explained as follows:

name
An optional symbol, starting in column 1, that is the name on the IWMGCORF macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

,ASYNC_FLAG=*async_flag*
An optional output parameter, which receives the indication whether or not the asynchronous flow flag is set. A value of 1 is returned, if the asynchronous flow flag is set. Otherwise 0 is returned. The caller must initialize the ASYNC_FLAG field before issuing the macro because IWMGCORF will not modify it if the EWLM_CORR field contains an invalid ARM correlator.

To code: Specify the RS-type address, or address in register (2)-(12), of a fullword field.

EWLM_CORR=*ewlm_corr*
A required input parameter, which contains an EWLM correlator. If the correlator is invalid (the architected length in the first two bytes is less than 4 or greater than 512), a call to this macro acts as a no-operation and no change to the ASYNC_FLAG or INDEP_FLAG is made.

To code: Specify the RS-type address, or address in register (2)-(12), of a character field.

,INDEP_FLAG=*indep_flag*
An optional output parameter, which receives the indication whether or not the independent flag is set. A value of 1 is returned, if the independent flag is set. Otherwise 0 is returned. The caller must initialize the INDEP_FLAG field before issuing the macro because IWMGCORF will not modify it if the EWLM_CORR field contains an invalid ARM correlator.

To code: Specify the RS-type address, or address in register (2)-(12), of a fullword field.

ABEND codes

None.

Return codes and reason codes

None.

Example

To extract the setting of the asynchronous and the independent flag from a storage area holding an ARM correlator, specify the following:

```
LHI   2,-1          INIT TO CATCH INVALID CORRELATOR
          ST    2,ASYNCHRO
          ST    2,INDEPEND
          IWMGCORF                                     *
                EWLM_CORR=MYCORR,                      *
                ASYNC_FLAG=ASYNCHRO,                   *
                INDEP_FLAG=INDEPEND
          L     2,ASYNCHRO    LOAD ASNYCHRONOUS INDIDICATOR
          LTR   2,2           TEST ASYNC RESULT
          JP    ASYN_ON       ASYNC FLAG IS ON
          JZ    ASYN_OFF      ASYNC FLAG IS OFF
CORR_INVA DS    0H            NOT A VALID ARM CORRELATOR
          :
          :
ASYN_ON   L     3,INDEPEND    LOAD INDEPENDENT INDICATOR
          LTR   3,3           TEST INDEP (RELEVANT, IF ASYNC)
          JP    INDE_ON
INDE_OFF  DS    0H            ASYNC ON, BUT NOT INDEPENDANT
          :
          :
ASYNCHRO  DS    F             STORAGE AREA FOR ASYNC FLAG
INDEPEND  DS    F             STORAGE AREA FOR INDEP FLAG
MYCORR    DS    CL512         STORAGE AREA FOR ARM CORRELATOR
```

IWMIMPT — Import an enclave

The IWMIMPT macro imports an enclave that has been previously exported using the IWMEXPT 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:

Minimum authorization:	Supervisor state or program key mask (PKM) allowing keys 0-7.
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any HASN, any SASN. PASN must be the address space which connected to WLM.
AMODE:	31-bit
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks may be held.
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, IWMRSNCODE_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:

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

The syntax of the IWMIMPT macro is as follows:

```
►►─┬──────┬─IWMIMPT──XTOKEN=xtoken──,ETOKEN=etoken──,CONNTKN=conntkn───────────────────►
   └─name─┘

                                                     ┌─,PLISTVER=IMPLIED_VERSION─┐
►─┬──────────────────┬─┬──────────────────┬─┼───────────────────────────┼──────────────►
  └─,RETCODE=retcode─┘ └─,RSNCODE=rsncode─┘ ├─,PLISTVER=MAX─────────────┤
                                            └─,PLISTVER=0───────────────┘

  ┌─,MF=S──────────────────────────────────┐
►─┼────────────────────────────────────────┼───────────────────────────────────────────►◄
  │                       ┌─,0D────┐       │
  ├─,MF=(L─,list addr─────┼────────┼─)─────┤
  │                       └─,attr──┘       │
  │                       ┌─,COMPLETE─┐    │
  └─,MF=(E─,list addr─────┴───────────┴─)──┘
```

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***,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).

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 codes 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	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.
4	xxxx0432	**Equate Symbol**: IwmRsnCodeUnknownExportToken **Meaning**: 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. **Action**: Discontinue processing the unit of work.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: The caller is locked. **Action**: Avoid requesting this function while locked.
8	xxxx080B	**Equate Symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list. **Action**: Check for possible storage overlay.
8	xxxx0821	**Equate Symbol**: IwmRsnCodeBadConn **Meaning**: Input connect token does not pass validity checking. **Action**: 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	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: The caller invoked the service but was in 24-bit addressing mode. **Action**: Request this function only when you are in 31-bit addressing mode.
8	xxxx0825	**Equate Symbol**: IwmRsnCodeAscModeNotPrimary **Meaning**: The caller invoked the service but was not in primary ASC mode. **Action**: Avoid requesting this function in this environment.
8	xxxx0827	**Equate Symbol**: IwmRsnCodeRsvdNot0 **Meaning**: Reserved field in parameter list was non-zero. **Action**: Check for use of keywords that are not supported by the OS/390® release on which the program is running.

Table 42. Return and Reason Codes for the IWMIMPT Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx0828	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: Version number in parameter list is not valid. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx0836	**Equate Symbol**: IwmRsnCodeMaxEnclave **Meaning**: Enclave could not be created because the enclave limit ha been reached. **Action**: 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	**Equate Symbol**: IwmRsnCodeServiceNotEnabled **Meaning**: The caller's connection is not enabled for this service. **Action**: Make sure that EXPTIMPT=YES is specified on the IWM4CON macro invocation.
8	xxxx0842	**Equate Symbol**: IwmRsnCodeNoWLMConnect **Meaning**: Caller's primary address space is not connected to WLM. **Action**: Invoke the IWM4CON macro before invoking this macro.
8	xxxx0870	**Equate Symbol**: IwmRsnCodeBadExportToken **Meaning**: The export token is not validly formatted. **Action**: Check for possible storage overlay of the export token.
C	—	**Equate Symbol**: IwmRetCodeEnvError **Meaning**: Environmental error.
C	xxxx0C01	**Equate Symbol**: IwmRsnCodeNoStg **Meaning**: Virtual storage is not available for the request. **Action**: Contact your system programmer.
C	xxxx0C36	**Equate Symbol**: IwmRsnCodeStructureUnavailable **Meaning**: WLM does not have access to its coupling facility structure. **Action**: Check for WLM or XES messages which describe the problem.
C	xxxx0C38	**Equate Symbol**: IwmRsnCodeUplevelObject **Meaning**: The multisystem enclave requires functions that are not available on this level of the operating system. **Action**: Do not process the work request on this system.

Table 42. Return and Reason Codes for the IWMIMPT Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
C	xxxx0C39	**Equate Symbol**: IwmRsnCodeTooManySystems **Meaning**: The sysplex has exceeded 32 systems with unique names. This can occur when a system is reIPLed into the sysplex with a different SYSNAME or CPU Adjustment factor. **Action**: Do not process the work request on this system.
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error. **Action**: Contact your system programmer.

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
```

IWMMXDC — Exit for resource data collection

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:

Minimum authorization:	Supervisor state with the same PSW key as at the time of registration (IWM4MREG).
Dispatchable unit mode:	Task or SRB
Cross-memory mode:	PASN=HASN=SASN. The current home address space must be the same as at the time of registration (IWM4MREG).
AMODE:	31-bit
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks may be held. FRRs may be established.
Control parameters:	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
- 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

The syntax of the IWMMXDC macro is as follows:

main diagram

```
►►──┬──────┬──IWMMXDC──RESOURCE_TKN=resource_tkn──,RES_DATA_EXIT@=res_data_exit@──────────►
    └─name─┘

►──┬─,RESOURCE_TYPE=BUFFER_POOL──┤ parameters-1 ├──┬──┬──────────────────┬──────────────────►
   └─,RESOURCE_TYPE=NULL───────────────────────────┘  └─,RETCODE=retcode─┘

                             ┌─,PLISTVER=IMPLIED_VERSION─┐
►──┬──────────────────┬──────┼───────────────────────────┼───────────────────────────────────►
   └─,RSNCODE=rsncode─┘      ├─,PLISTVER=MAX─────────────┤
                             └─,PLISTVER=0───────────────┘

   ┌─,MF=S─────────────────────────────────────────┐
►──┼───────────────────────────────────────────────┼──────────────────────────────────────►◄
   │                          ┌─,0D───┐            │
   ├─,MF=(L─,list addr────────┼───────┼──)─────────┤
   │                          └─,attr─┘            │
   │                          ┌─,COMPLETE─┐        │
   └─,MF=(E─,list addr────────┴───────────┴──)─────┘
```

parameters-1

```
►►──,OWNER_TKN=owner_tkn──┬──────────────────────────────┬──────────────────────────────────►
                          └─,RES_CUR_SIZE=res_cur_size───┘

►──┬──────────────────────────────────┬──┬────────────────────────────────────────┬─────────►
   └─,RES_INUSE_SIZE=res_inuse_size───┘  └─,RES_#REFERENCES=res_#references───────┘

►──┬──────────────────────────┬───────────────────────────────────────────────────────────►◄
   └─,RES_#HITS=res_#hits─────┘
```

Parameters

The parameters are explained as follows:

name

An optional symbol, starting in column 1, that is the name on the IWMMXDC 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 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_#HITS=*res_#hits*

When RESOURCE_TYPE=BUFFER_POOL is specified, an optional output parameter, which contains the number of hits among the references to the bufferpool since the last invocation of the data collection exit. The corresponding field in the parameter list (IWMMXDC 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_#REFERENCES=*res_#references*
When RESOURCE_TYPE=BUFFER_POOL is specified, an optional output parameter, which contains the number of references to the bufferpool since the last invocation of the data collection exit. The corresponding field in the parameter list (IWMMXDC 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 (IWMMXDC 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 (IWMMXDC 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 codes 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 43. Return and Reason Codes for the IWMMXDC Macro

Return Code	Reason Code	Equate Symbol, Meaning, and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.

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
```

IWMMXRA — Exit for resource adjustment

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:

Minimum authorization:	Supervisor state with the same PSW key as at the time of registration (IWM4MREG).
Dispatchable unit mode:	Task or SRB
Cross-memory mode:	PASN=HASN=SASN. The current home address space must be the same as at the time of registration (IWM4MREG).
AMODE:	31-bit
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks may be held. FRRs may be established.
Control parameters:	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
- 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

The syntax of the IWMMXRA macro is as follows:

main diagram

```
►►──┬──────┬──IWMMXRA──RESOURCE_TKN=resource_tkn──,RES_ADJ_EXIT@=res_adj_exit@──────────►
    └─name─┘
```

```
►──┬─,RESOURCE_TYPE=BUFFER_POOL─┤ parameters-1 ├─┬──┬────────────────┬──────────────────►
   └─,RESOURCE_TYPE=NULL─────────────────────────┘  └─,RETCODE=retcode─┘

                               ┌─,PLISTVER=IMPLIED_VERSION─┐
►──┬─────────────────────┬─────┼───────────────────────────┼──────────────────────────────►
   └─,RSNCODE=rsncode────┘     ├─,PLISTVER=MAX─────────────┤
                               └─,PLISTVER=0───────────────┘

   ┌─,MF=S─────────────────────────────────┐
►──┼───────────────────────────────────────┼──────────────────────────────────────────────►◄
   │                     ┌─,0D──────┐      │
   ├─,MF=(L─,list addr───┼──────────┼─)────┤
   │                     └─,attr────┘      │
   │                     ┌─,COMPLETE─┐     │
   └─,MF=(E─,list addr───┴───────────┴─)───┘
```

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 IWMMXRA 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 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, an 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 codes and reason codes

When the IWMMXRA 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 44. Return and Reason Codes for the IWMMXRA Macro

Return Code	Reason Code	Equate Symbol, Meaning, and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.

Example

To call a resource adjustment exit, specify the following:

```
         IWMMXRA                                              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
               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
```

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 the **SYSTEM_NAME** keyword.

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:

Minimum authorization:	Problem state. Any PSW key. The caller must have update authority to the resource MVSADMIN.WLM.POLICY in the FACILITY class.
Dispatchable unit mode:	Task
Cross memory mode:	PASN=HASN=SASN
AMODE:	31-bit
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks may be held.
Control parameters:	All parameter areas must reside in current primary.

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 that are 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 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.

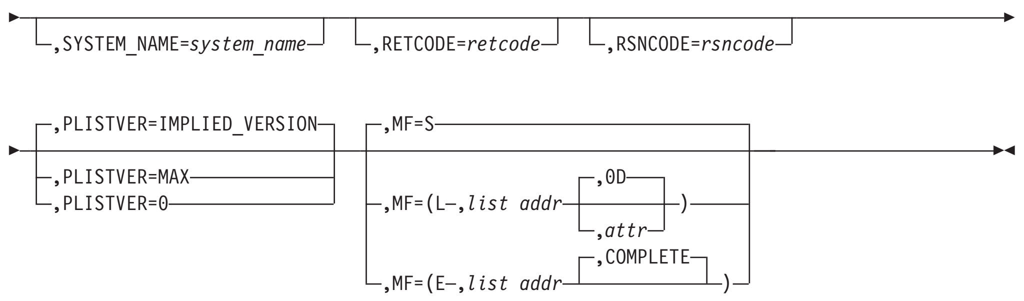
Performance implications

None.

Syntax

The syntax of the IWMPACT macro is as follows:

```
                                                     ┌─,CHECKHISTORY=CHECKALL──┐
►►──┬──────┬──IWMPACT──POLICY_NAME=policy_name──┴─────────────────────────┴──────────►
    └─name─┘                                         └─,CHECKHISTORY=IGNORE_CR─┘
```



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.

,CHECKHISTORY=CHECKALL
,CHECKHISTORY=IGNORE_CR

An optional input parameter that applies only if the WLM policy to be activated with this request is the same policy that was active previously, that is, if this is a request to reactivate a WLM policy. As a result of the changes in effect with the reactivated policy, due to a changed WLM service definition on the CDS, WLM might discard historical data for service class periods. Among the criteria that cause WLM to discard historical data are the following:

- Classification groups are added, changed in content, or removed.
- Classification rules are added, changed, or removed.

CHECKHISTORY changes the criteria that WLM applies to make that decision.

CHECKALL

Leave WLM's behavior unchanged. Historical data is discarded when classification groups or classification rules are changed. This is the default.

IGNORE_CR

Requests that WLM not check for changes in classification groups or changes in classification rules when making the decision about whether or not to discard historical data. Use this option if the only changes to your WLM service definition are in the categories listed above and if keeping the historical data does not negatively impact your workload.

,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

X'0Axx0005'
An attempt to reference caller's parameters caused an OC4 abend.

Return codes 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 45. Return and Reason Codes for the IWMPACT Macro

Return Code	Reason Code	Equate Symbol, Meaning, and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.
4	xxxx0414	**Equate Symbol**: IwmRsnCodeNullCDS **Meaning**: This request could not be completed because no Service definition has been installed on the WLM CDS. **Action**: None required.

Table 45. Return and Reason Codes for the IWMPACT Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
4	xxxx0415	**Equate Symbol**: IwmRsnCodePolicyActInProgress **Meaning**: This request could not be completed because another policy activation request is currently being processed. If specified, the SYSTEM_NAME parameter will contain the name of the system on which policy activation is in progress. **Action**: None required. If this service is re-invoked at a later time it may be successful.
4	xxxx0416	**Equate Symbol**: IwmRsnCodePolicyUndefined **Meaning**: 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. **Action**: None required. Verify that the policy was specified correctly.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0801	**Equate Symbol**: IwmRsnCodeSrbMode **Meaning**: The caller is in SRB mode. **Action**: Avoid requesting this function while in SRB mode.
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: The caller is locked. **Action**: Avoid requesting this function while locked.
8	xxxx0810	**Equate Symbol**: IwmRsnCodeEutFrr **Meaning**: The caller has EUT FRR established. **Action**: Avoid requesting this function with an EUT FRR set.
8	xxxx0823	**Equate Symbol**: IwmRsnCodeDatoff **Meaning**: The caller invoked the service while DATOFF **Action**: Avoid requesting this function in this environment.
8	xxxx0824	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: The caller invoked the service but was in 24-bit addressing mode. **Action**: Request this function only when you are in 31-bit addressing mode.
8	xxxx0827	**Equate Symbol**: IwmRsnCodeRsvdNot0 **Meaning**: Reserved field in parameter list was non-zero. **Action**: Check for possible storage overlay of the parameter list.

Table 45. Return and Reason Codes for the IWMPACT Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx0828	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: Version number in parameter list is not valid or the length specified is incorrect. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx0841	**Equate Symbol**: IwmRsnCodeXMemMode **Meaning**: The caller is in cross memory mode. **Action**: Invoke the function in non-cross memory mode.
C	—	**Equate Symbol**: IwmRetCodeEnvError **Meaning**: Environmental error.
C	xxxx0C01	**Equate Symbol**: IwmRsnCodeNoStg **Meaning**: Storage is not available for the request. **Action**: There is a storage shortage. The function may work successfully at a later time.
C	xxxx0C0E	**Equate Symbol**: IwmRsnCodeInsufAccess **Meaning**: The caller does not have update authority to the RACF resource MVSADMIN.WLM.POLICY in the FACILITY class. **Action**: Invoke the function when the condition is fulfilled.
C	xxxx0C0F	**Equate Symbol**: IwmRsnCodeCDSNotAvail **Meaning**: 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. **Action**: No action required.
C	xxxx0C10	**Equate Symbol**: IwmRsnCodeCDSTooSmall **Meaning**: WLM CDS is too small to process the request. **Action**: No action required.
C	xxxx0C11	**Equate Symbol**: IwmRsnCodeOneSystemUnable **Meaning**: 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. **Action**: No action required.
C	xxxx0C13	**Equate Symbol**: IwmRsnCodePolicyNotAvail **Meaning**: 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. **Action**: No action required.

Table 45. Return and Reason Codes for the IWMPACT Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
C	xxxx0C1E	**Equate Symbol**: IwmRsncodeHigherVersionLevel **Meaning**: The policy cannot be activated on this system because the service definition in the WLM couple data set is at a higher level than the WLM code on this system. A system with a lower level version cannot activate this service policy because it is not capable of handling all the function in the service definition. **Action**: None required. If necessary, reinvoke the service on a higher level system.
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error. **Action**: No action required. The function may be successful if invoked again.

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 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 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:

Minimum authorization:	Supervisor state or program key mask (PKM) allowing keys 0-7.
Dispatchable unit mode:	Task
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31-bit
ASC mode:	Primary or access register (AR) If in Access Register ASC mode, specify SYSSTATE ASCENV=AR before invoking this macro.
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks may be held.
Control parameters:	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. 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 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

The syntax of the IWMPQRY macro is as follows:

```
►►──┬──────┬──IWMPQRY──ANSAREA=ansarea──,ANSLEN=anslen──,QUERYLEN=querylen──────────────►
    └─name─┘

                                       ┌─,PLISTVER=IMPLIED_VERSION─┐
►──┬────────────────┬──┬────────────────┬──┼───────────────────────────┼──────────────────►
   └─,RETCODE=retcode─┘  └─,RSNCODE=rsncode─┘  ├─,PLISTVER=MAX─────────────┤
                                       └─,PLISTVER=0───────────────┘
   ┌─,MF=S──────────────────────────────┐
►──┼────────────────────────────────────┼──────────────────────────────────────►◄
   │                      ┌─,0D───┐     │
   ├─,MF=(L─,list addr────┼───────┼─)───┤
   │                      └─,attr─┘     │
   │                      ┌─,COMPLETE─┐ │
   └─,MF=(E─,list addr────┴───────────┴)┘
```

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.

,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
X'0Axx0005'	An attempt to reference caller's parameters caused an OC4 abend.

Return codes 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 46. Return and Reason Codes for the IWMPQRY Macro

Return Code	Reason code	Equate Symbol, Meaning, and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.
4	xxxx040A	**Equate Symbol**: IwmRsnCodeOutputAreaTooSmall **Meaning**: The output area supplied is too small to receive all the available information. **Action**: None required. If necessary, reinvoke the service with an output area of sufficient size to receive all information.
4	xxxx0423	**Equate Symbol**: IwmRsnCodeDefaultPolicy **Meaning**: The default policy was returned. **Action**: None required.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0801	**Equate Symbol**: IwmRsnCodeSrbMode **Meaning**: The caller is in SRB mode. **Action**: Avoid requesting this function while in SRB mode.
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: The caller is locked. **Action**: Avoid requesting this function while locked.
8	xxxx0810	**Equate Symbol**: IwmRsnCodeEutFrr **Meaning**: The caller has EUT FRR established. **Action**: Avoid requesting this function with an EUT FRR set.

Table 46. Return and Reason Codes for the IWMPQRY Macro (continued)

Return Code	Reason code	Equate Symbol, Meaning, and Action
8	xxxx0830	**Equate Symbol**: IwmRsnCodeBadAlet **Meaning**: The caller has passed an invalid ALET. **Action**: Check for possible storage overlay of the parameter list or variable.

IWMQCXIT — Queue manager connect exit

IWMQCXIT will invoke the Queue Manager Connect exit specified. This exit is responsible for taking the action indicated for the associated Queue Manager. The return/reason codes for IWMQCXIT are those set by the exit invoked.

The list form of IWMQCXIT is intended for use by the exit routine to map the input parameters. The execute and standard form of IWMQCXIT are intended for use only by the operating system.

The exit environment is described in the environment description below. 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 as 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, `IWMQCXIT 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:

Minimum authorization:	Supervisor state 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 may be held.
Control parameters:	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 Queue Manager Connect exit.

The execute form of the macro is intended for use by MVS. The assembler execute form only initializes the parameter list and calls the exit routine, and has the following restrictions:

- the invoker must save registers required before invoking the macro

- the invoker must restore registers required immediately after invoking the macro, without depending on the exit to preserve any registers.

Restrictions

- The exit routine may not invoke functions or suspend execution which prevents return to the caller for a protracted timeframe. This includes the use of system services which either explicitly or implicitly give control back to the system. In this context, protracted would include durations of one second or longer. When the need for such activities is required, the exit should use asynchronous techniques.

Input register information

Before issuing the IWMQCXIT 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 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

```
►►──┬──────┬──b──IWMQCXIT──b──QMGR_EXIT@=qmgr_exit@──┬,ACTION=QDEL┬──────────────►
    └─name─┘                                          └,ACTION=NULL┘
```

```
►─,ETOKEN=etoken──,USERDATA=userdata──,APPLENV=applenv──────────────────────────►

  ┌─,SECUSER=NO──────────────────────┐
►─┼──────────────────────────────────┼──┬───────────────────┬──┬───────────────────┬──►
  └─,SECUSER=YES─,USERID=userid──────┘  └─,RETCODE=retcode──┘  └─,RSNCODE=rsncode──┘

  ┌─,PLISTVER=IMPLIED_VERSION─┐   ┌─,MF=S──────────────────────────────────────┐
►─┼───────────────────────────┼───┼────────────────────────────────────────────┼──►◄
  ├─,PLISTVER=MAX─────────────┤   │                         ┌─,0D────┐         │
  └─,PLISTVER=0───────────────┘   ├─,MF=(L─,list addr───────┼────────┼─)───────┤
                                  │                         └─,attr──┘         │
                                  │                         ┌─,COMPLETE─┐      │
                                  └─,MF=(E─,list addr───────┴───────────┴─)────┘
```

Parameters

The parameters are explained as follows:

name
An optional symbol, starting in column 1, that is the name on the IWMQCXIT macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

,ACTION=QDEL
,ACTION=NULL
A required parameter, which indicates the type of action requested. The exit should make an explicit check for the action indicated to anticipate the introduction of new values in later releases.

,ACTION=QDEL
indicates that exit is being called for the deletion of a previously queued work element. The exit will only be called once per queued work element. If the exit should end abnormally, the system will not invoke the exit again for the same queued work element. In the event of an abnormal termination of the exit routine, the system may chose to call the exit for any remaining queued work elements, or may chose to discontinue use of the exit upon some threshold number of errors.

,ACTION=NULL
indicates that no exit is to be called. The exit routine need not check for this action.

,APPLENV=*applenv*
A required input parameter, which contains the application environment name associated with the queued work request or blanks if not available.

To code: Specify the RS-type address, or address in register (2)-(12), of a 32-character field.

,ETOKEN=*etoken*
A required input parameter, which contains the Enclave token associated with the work request at the time the work request was queued or binary zeros when the system knows the Enclave no longer exists or the system no longer knows which Enclave was associated with the work request.

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

QMGR_EXIT@=*qmgr_exit@*

A required input parameter that contains the address of the Queue Manager Connect Exit to be invoked.

To code: Specify the RS-type address, or address in register (2)-(12), of a pointer field.

,RETCODE=*retcode*

An optional output parameter into which the return code is to be copied from GPR 15. If you specify 15, GPR15, REG15, or R15 (within or without parentheses), the value will be left in GPR 15.

To code: Specify the RS-type address of a fullword field, or register (2)-(12) or (15), (GPR15), (REG15), or (R15).

,RSNCODE=*rsncode*

An optional output parameter into which the reason code is to be copied from GPR 0. If you specify 0, 00, GPR0, GPR00, REG0, REG00, or R0 (within or without parentheses), the value will be left in GPR 0.

To code: Specify the RS-type address of a fullword field, or register (0) or (2)-(12), (00), (GPR0), (GPR00), REG0), (REG00), or (R0).

,SECUSER=NO
,SECUSER=YES

An optional parameter, which specifies whether the security environment of a user was associated with the queued work request. The default is SECUSER=NO.

,SECUSER=NO

No security environment was to be established.

,SECUSER=YES

The specified userid was to be used to establish a security environment.

,USERDATA=*userdata*

A required input parameter, which contains data passed to Queue Insert. 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.

ABEND codes

None.

Return codes and reason codes

When the IWMQCXIT 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.

Table 47. Return and Reason Codes for the IWMQCXIT Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.

Example

To call a Queue Manager connect exit:

```
        IWMQCXIT                                      X
              QMGR_EXIT@=CONEXIT@,                    X
              ACTION=QDEL,                            X
              ETOKEN=ENCTOKEN,                        X
              USERDATA=USERDATA,APPLENV=APPLENV,SECUSER=NO, X
              RETCODE=RC,RSNCODE=RSN
*
* Storage areas
*
CONEXIT@   DS  AL4             Contains the address of the
*                              Queue Manager Connect Exit
*                              to be invoked
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
*
RC       DS    F               Return code
RSN      DS    F               Reason code
```

IWMRCOLL — Collect 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 101.

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, for example, 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:

Minimum authorization:	Supervisor state or program key mask (PKM) allowing keys 0-7
Dispatchable unit mode:	Task
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31-bit

ASC mode:	Primary or access register. If in access register ASC mode, specify SYSSTATE ASCENV = AR before invoking IWMRCOLL.
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks held.
Control parameters:	Control parameters must be in the primary address space. The caller of IWMRCOLL must provide storage for an answer area mapped by IWMWRCAA. 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).

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:

Register	Contents
0	Reason code if the GP 15 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.

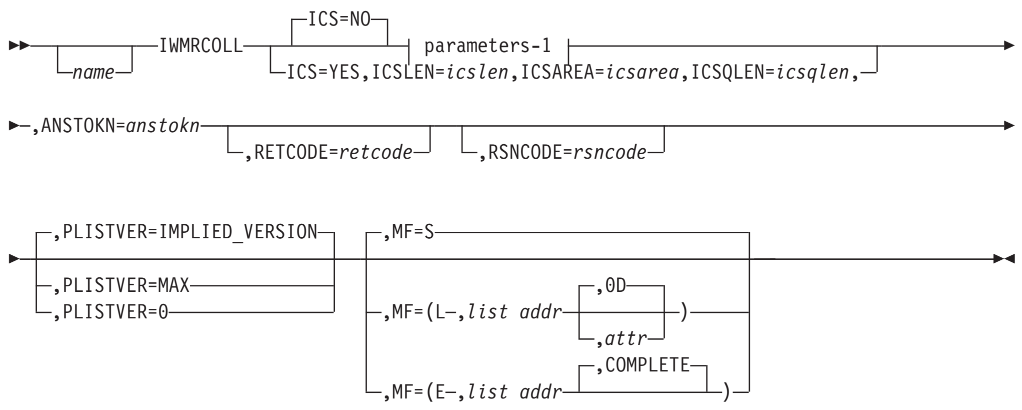
Performance implications

None.

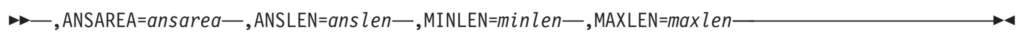
Syntax

The syntax of the IWMRCOLL macro is as follows:

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*
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*
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
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.

,ICSLEN=*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*
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***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 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*
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. If you specify 15, GPR15, REG15, or R15 (with or without parentheses), the value will be left in GPR 15.

To code: Specify the RS-type address of a fullword field, or register (2)-(12), or (15), (GPR15), (REG15), or (R15).

,RSNCODE=*rsncode*
An optional output parameter into which the reason code is to be copied from GPR 0. If you specify 0, 00, GPR0, GPR00, REG0, REG00, or R0 (with or without parentheses), the value will be left in GPR 0.

To code: Specify the RS-type address of a fullword field, or register (0) or (2)-(12), (00), (GPR0), (GPR00), REG0), (REG00), or (R0).

ABEND codes

None.

Return codes 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 48. Return and Reason Codes for the IWMRCOLL Macro

Return Code	Return Code	Equate Symbol, Meaning, and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. All requested data returned. **Action**: None required.
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.
4	xxxx040A	**Equate Symbol**: IwmRsnCodeOutputAreaTooSmall **Meaning**: 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. **Action**: None required. If necessary, reinvoke the service with an output area of sufficient size to receive all information.

Table 48. Return and Reason Codes for the IWMRCOLL Macro (continued)

Return Code	Return Code	Equate Symbol, Meaning, and Action
4	xxxx040F	**Equate Symbol**: IwmRsnCodeStateInvDataRet **Meaning**: 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. **Action**: Reinvoke the service with the token passed with the ANSTOKN keyword.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0801	**Equate Symbol**: IwmRsnCodeSrbMode **Meaning**: The caller is in SRB mode. **Action**: Avoid requesting this function while in SRB mode.
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: The caller is disabled. **Action**: Avoid requesting this function while disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: The caller is locked. **Action**: Avoid requesting this function while locked.
8	xxxx0810	**Equate Symbol**: IwmRsnCodeEutFrr **Meaning**: The caller has an EUT FRR set. **Action**: Avoid requesting this function with an EUT FRR set.
8	xxxx0830	**Equate Symbol**: IwmRsnCodeBadAlet **Meaning**: The caller specified an invalid alet for the storage pointed to by the ANSAREA keyword. **Action**: Check for possible storage overlay of the parameter list or variable.
8	xxxx0832	**Equate Symbol**: IwmRsnCodeStateInvNoDatRet **Meaning**: 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. **Action**: Reinvoke the service with an output area of sufficient size to receive all information and the token passed with the ANSTOKN keyword.

Table 48. Return and Reason Codes for the IWMRCOLL Macro (continued)

Return Code	Return Code	Equate Symbol, Meaning, and Action
8	xxxx0833	**Equate Symbol**: IwmRsnCodeNotInCompatMode **Meaning**: IWMRCOLL was invoked with a parameter list specifying ICS=YES. WLM no longer supports compatibility mode, and therefore no longer returns ICS information. **Action**: Reinvoke IWMRCOLL with ICS=NO specified.
C	—	**Equate Symbol**: IwmRetCodeEnvError **Meaning**: Environmental error
C	xxxx0C01	**Equate Symbol**: IwmRsnCodeNoStg **Meaning**: Storage is not available for the request. **Action**: There is a storage shortage. The function may work successfully at a later time.
C	xxxx0C0A	**Equate Symbol**: IwmRsnCodeSuspended **Meaning**: 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. **Action**: Reinvoke this service.
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error **Action**: No action required. The function may work successfully if invoked again.

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)
```

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, 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:

Minimum authorization:	Supervisor state or program key mask (PKM) allowing keys 0-7.
Dispatchable unit mode:	Task
Cross memory mode:	PASN=HASN=SASN
AMODE:	31-bit
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks may be held.
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, 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 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.

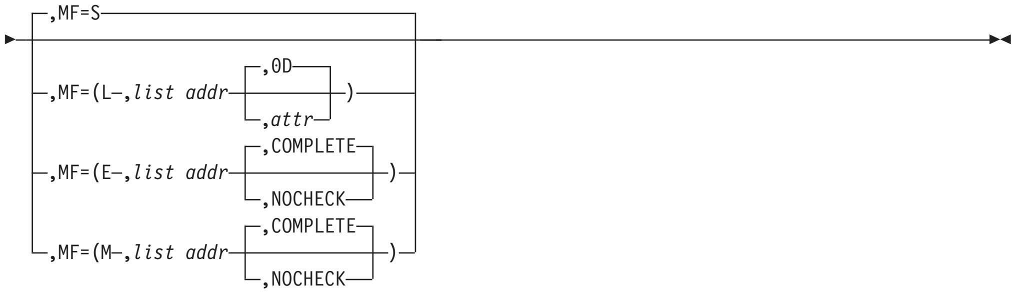
Performance implications

None.

Syntax

The syntax of the IWMRESET macro is as follows:

```
►►──┬──────┬──IWMRESET──┬─JOBNAME=jobname──┬──────────────────────────────────────►
    └─name─┘            │  ┌─.,──────────┐ │
                        │  ▼             │ │
                        └──ASID=asid─────┴─┘

   ┌─,FUNCTION=RESET─┐
►──┼─────────────────┴─,SRVCLASS=srvclass─┬─,USERID=userid─,PRODUCT=product───────►
   ├─,FUNCTION=QUIESCE────────────────────┤
   └─,FUNCTION=RESUME─────────────────────┘

                                          ┌─,PLISTVER=IMPLIED_VERSION─┐
►──┬──────────────────┬──┬──────────────────┬──┼───────────────────────────┼─────────►
   └─,RETCODE=retcode─┘  └─,RSNCODE=rsncode─┘  ├─,PLISTVER=MAX─────────────┤
                                          └─,PLISTVER=0───────────────┘
```



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***,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 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=*srvclass*

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 user ID is included in the SMF 90 subtype 30 record created by IWMRESET. If there is no user ID 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 codes and reason codes

When the IWMRESET 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 IWMRESET Macro

Return Code	Reason Code	Equate Symbol, Meaning, and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.
4	xxxx0408	**Equate Symbol**: IwmRsnCodeWorkNotFound **Meaning**: A job matching the input job name or ASID was not found. **Action**: The caller should report the error appropriately.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0801	**Equate Symbol**: IwmRsnCodeSRBMode **Meaning**: The caller is in SRB mode. **Action**: Avoid requesting this function while in SRB mode.
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: The caller is locked. **Action**: Avoid requesting this function while locked.
8	xxxx080B	**Equate Symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list. **Action**: Check for possible storage overlay.
8	xxxx0810	**Equate Symbol**: IwmRsnCodeEutFrr **Meaning**: The caller has EUT FRR established. **Action**: Avoid requesting this function with an EUT FRR set.

Table 49. Return and Reason Codes for the IWMRESET Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx0824	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: The caller invoked the service but was in 24-it addressing mode. **Action**: Request this function only when you are in 31-bit addressing mode.
8	xxxx0825	**Equate Symbol**: IwmRsnCodeAscModeNotPrimary **Meaning**: The caller invoked the service but was not in primary ASC mode. **Action**: Avoid requesting this function in this environment.
8	xxxx0827	**Equate Symbol**: IwmRsnCodeRsvdNot0 **Meaning**: Reserved field in parameter list was non-zero. **Action**: Check for use of keywords that are not supported by the MVS Release on which the program is running.
8	xxxx0828	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: Version number in parameter list is not valid. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx0829	**Equate Symbol**: IwmRsnCodeBadOptions **Meaning**: Parameter list omits required parameters. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx0841	**Equate Symbol**: IwmRsnCodeXMemMode **Meaning**: The caller is in cross-memory mode. **Action**: Invoke the function in non-cross memory mode.
C	—	**Equate Symbol**: IwmRetCodeEnvError **Meaning**: Environmental error.
C	xxxx0C28	**Equate Symbol**: IwmRsnCodeBadServiceClass **Meaning**: The input service class name is not defined in the active workload manager policy. **Action**: Record or report the error if appropriate.
C	xxxx0C2D	**Equate Symbol**: IwmRsnCodeBadPerformanceGroup **Meaning**: Reserved.
C	xxxx0C2E	**Equate Symbol**: IwmRsnCodeWrongMode **Meaning**: The caller tried to perform a goal-mode function in compatibility mode, or vice versa. **Action**: Record or report the error if appropriate.

Table 49. Return and Reason Codes for the IWMRESET Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
C	xxxx0C2F	**Equate Symbol**: IwmRsnCodeSystemSpace **Meaning**: 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. The system component address space still cannot be reset. **Action**: Record or report the error if appropriate.
C	xxxx0C30	**Equate Symbol**: IwmRsnCodeDuplicateJobs **Meaning**: There is more than one job active with the same jobname. **Action**: Specify the ASID parameter to identify the specific job.
C	xxxx0C31	**Equate Symbol**: IwmRsnCodeWrongASID **Meaning**: The active job in the specified address space has a different jobname than the one passed by the caller. **Action**: Record or report the error if appropriate.
C	xxxx0C32	**Equate Symbol**: IwmRsnCodeNotEligibleForSrvClass **Meaning**: 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. **Action**: Record or report the error if appropriate.
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error. **Action**: Contact your system programmer.

Example

To change the service class of the job executing in the ASID at location JOBASID, specify:

```
         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
```

IWMRQRY — Collect 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 (DUal).

Environment

The requirements for the caller are:

Minimum authorization:	Supervisor state or program key mask (PKM) allowing keys 0-7.
Dispatchable unit mode:	Task
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31-bit
ASC mode:	Primary or access register (AR) Any P,S,H. If in Access Register ASC mode, specify `SYSSTATE ASCENV=AR` before invoking this macro.
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks may be held.
Control parameters:	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 (DUal).

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 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:

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.

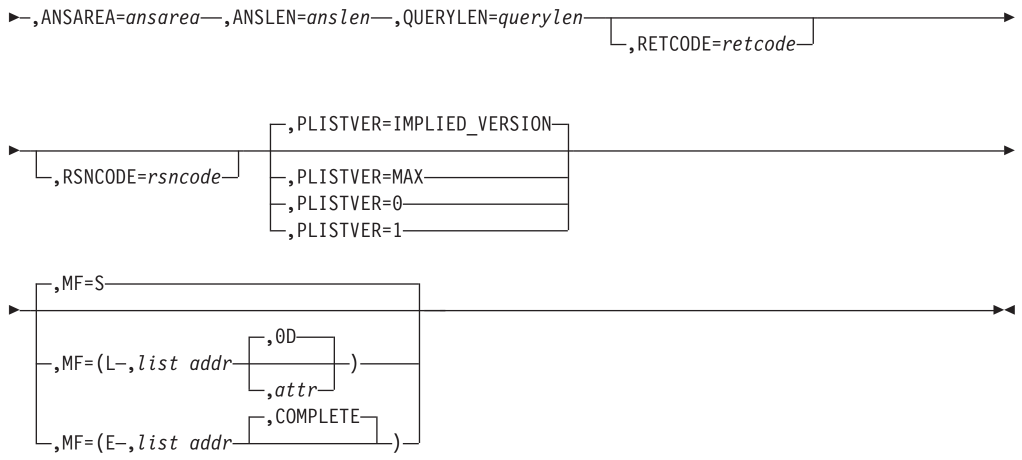
Performance implications

None.

Syntax

The syntax of the IWMRQRY macro is as follows:

```
                                                 ┌─,ENCLAVES=NONE────────────────┐
►►──┬──────┬──IWMRQRY──┬─INFO=ALL──────────────┬──┼───────────────────────────────┼──────►
    └─name─┘           └─INFO=ONE─,ASID=asid───┘  ├─,ENCLAVES=ALL─────────────────┤
                                                  └─,ENCLAVES=ONE─,ETOKEN=etoken──┘
```



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.

,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 the following parameter and those of version 0:

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 codes 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 50. Return and Reason Codes for the IWMRQRY Macro

Return Code	Reason Code	Equate Symbol, Meaning, and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.
4	xxxx040A	**Equate Symbol**: IwmRsnCodeOutputAreaTooSmall **Meaning**: The output area supplied is too small to receive all the available information. **Action**: None required. If necessary, reinvoke the service with an output area of sufficient size to receive all information.
4	xxxx042C	**Equate Symbol**: IwmRsnCodeEtokenNoMatch **Meaning**: No enclave information matching the input enclave token was found. Enclave related information is not returned. **Action**: None required.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0801	**Equate Symbol**: IwmRsnCodeSrbMode **Meaning**: Caller is in SRB mode. **Action**: Avoid requesting this function while in SRB mode.
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: Caller is locked. **Action**: Avoid requesting this function while locked.
8	xxxx080B	**Equate Symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list. **Action**: Check for possible storage overlay.

Table 50. Return and Reason Codes for the IWMRQRY Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx0810	**Equate Symbol**: IwmRsnCodeEutFrr **Meaning**: Caller has EUT FRR established. **Action**: Avoid requesting this function with an EUT FRR set.
8	xxxx0812	**Equate Symbol**: IwmRsnCodeBadAscb **Meaning**: The ASID value specified on the ASID keyword is invalid. **Action**: Check for possible storage overlay.
8	xxxx0828	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: Version number in parameter list is not valid or version length field is not valid. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx0830	**Equate Symbol**: IwmRsnCodeBadAlet **Meaning**: Caller has passed an invalid ALET. **Action**: Check for possible storage overlay of the parameter list or variable.
C	—	**Equate Symbol**: IwmRetCodeEnvError **Meaning**: Environmental error.
C	xxxx0C01	**Equate Symbol**: IwmRsnCodeNoStg **Meaning**: Storage is not available for the request. **Action**: There is a storage shortage. The function may work successfully at a later time.
C	xxxx0C0A	**Equate Symbol**: IwmRsnCodeSuspended **Meaning**: Data sampling is suspended as a result of a component error. No data can be returned for this IWMRQRY invocation. **Action**: Reinvoke the function as it may be sucessful.
C	xxxx0C0B	**Equate Symbol**: IwmRsnCodeStateChanged **Meaning**: 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. **Action**: The current sampling interval should be bypassed, future invocations of IWMRQRY for subsequent sampling intervals should begin returning data again.
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error. **Action**: No action required. The function may be successful if invoked again.

IWMSCORF — Set correlator flags

This service allows to set or clear certain correlator flags in a provided EWLM correlator. These flags are contained in byte 3 of the architected Application Response Measurement (ARM) correlator format.

Environment

The requirements for the caller are:

Minimum authorization:	Problem state or supervisor state.
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31- or 64-bit. If in 64-bit addressing mode, code `SYSSTATE AMODE64=YES` before invoking this macro.
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks may be held. No restriction.
Control parameters:	Control parameters must be in the primary address space.

Programming requirements

None.

Restrictions

1. The caller is responsible for error recovery.
2. The caller must serialize to prevent any correlator from being accessed concurrently.

Input register information

Before issuing the IWMSCORF 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
0-1	Used as work register by the system
2-13	Unchanged
14-15	Used as work register by the system

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

```
►►──┬──────┬──b──IWMSCORF──b──EWLM_CORR=ewlm_corr──────────────────────────►
    └─name─┘

   ┌─,SET_ASYNC_FLAG=NO_SET_ASYNC─┐  ┌─,SET_INDEP_FLAG=NO_SET_INDEP─┐
►──┼──────────────────────────────┼──┼──────────────────────────────┼──────►
   ├─,SET_ASYNC_FLAG=OFF──────────┤  ├─,SET_INDEP_FLAG=OFF──────────┤
   └─,SET_ASYNC_FLAG=ON───────────┘  └─,SET_INDEP_FLAG=ON───────────┘

►──┬──────────────────┬──┬──────────────────┬──────────────────────────────►
   └─,RETCODE=retcode─┘  └─,RSNCODE=rsncode─┘

                          ┌─,COMPLETE─┐
►─,MF=(M─,list addr──┼───────────┼─)──────────────────────────────────►◄
                          └─,NOCHECK──┘
```

Parameters

The parameters are explained as follows:

name
An optional symbol, starting in column 1, that is the name on the IWMSCORF macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

EWLM_CORR=*ewlm_corr*
A required input/output parameter, which contains an EWLM (ARM) correlator. If the correlator is invalid (architected length in the first two bytes is less than 4 or greater than 512) a call to this macro acts as a no-operation and no change to the EWLM_CORR field is made.

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. If you specify 15, GPR15, REG15, or R15 (within or without parentheses), the value will be left in GPR 15.

To code: Specify the RS-type address of a fullword field, or register (2)-(12) or (15), (GPR15), (REG15), or (R15).

,RSNCODE=*rsncode*
An optional output parameter into which the reason code is to be copied from

GPR 0. If you specify 0, 00, GPR0, GPR00, REG0, REG00, or R0 (within or without parentheses), the value will be left in GPR 0.

To code: Specify the RS-type address of a fullword field, or register (0) or (2)-(12), (00), (GPR0), (GPR00), REG0), (REG00), or (R0).

,SET_ASYNC_FLAG=NO_SET_ASYNC
,SET_ASYNC_FLAG=OFF
,SET_ASYNC_FLAG=ON
An optional parameter, which requests to update the asynchronous flow flag. The default is SET_ASYNC_FLAG=NO_SET_ASYNC.

,SET_ASYNC_FLAG=NO_SET_ASYNC

indicates that parameter SET_ASYNC_FLAG has not been specified and the asynchronous flow flag in the EWLM correlator will not be changed.

,SET_ASYNC_FLAG=OFF

requests to clear the asynchronous flow flag. The independent flag is also cleared, since the independent flag may only be ON, if the asynchronous flag is ON.

,SET_ASYNC_FLAG=ON

requests to set the asynchronous flow flag to ON.

,SET_INDEP_FLAG=NO_SET_INDEP
,SET_INDEP_FLAG=OFF
,SET_INDEP_FLAG=ON
An optional parameter, which requests to update the independent flag. The default is SET_INDEP_FLAG=NO_SET_INDEP.

,SET_INDEP_FLAG=NO_SET_INDEP

indicates that parameter SET_INDEP_FLAG has not been specified and the independent flag in the EWLM correlator will not be changed.

,SET_INDEP_FLAG=OFF

requests to clear the independent flag.

,SET_INDEP_FLAG=ON

requests to set the independent flag to ON. The independent flag should be set only if the asynchronous flag is already set or will be set by this request.

ABEND codes

None.

Return codes and reason codes

When the IWMSCORF 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.

Examples

IWMSCXIT — Server manager connect exit

IWMSCXIT will invoke the Server Manager Connect exit specified. This exit is responsible for taking the action indicated on behalf of the current connected server address space. The return and reason codes for IWMSCXIT are set by the exit that was invoked.

The list form of IWMSCXIT is intended for use by the exit routine to map the input parameters. The execute and standard form of IWMSCXIT are intended for use only by the operating system.

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 exit environment is described in the environment description below. 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 as 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 will contain the address of a parameter list as formatted by the list form of this macro, `IWMSCXIT 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:

Minimum authorization:	Supervisor state key 0.
Dispatchable unit mode:	SRB
Cross memory mode:	PASN=HASN=SASN The current home address space must be the same as at time of Connect (IWMCONN).
AMODE:	31-bit
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks may be held. FRRs may be established.
Control parameters:	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 Server Manager Connect exit.

The execute form of the macro is intended for use by MVS. The assembler execute form only initializes the parameter list and calls the exit routine, and has the following restrictions:

- The invoker must save registers required before invoking the 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 IWMSCXIT 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 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

```
►►──┬──────┬──b──IWMSCXIT──b──CONNTKN=conntkn──,SRV_MGR_EXIT@=srv_mgr_exit@──────►
    └─name─┘

►──┬─,ACTION=QUIESCE─┬──┬──────────────────┬──┬──────────────────┬──────────────────►
   ├─,ACTION=RESUME──┤  └─,RETCODE=retcode─┘  └─,RSNCODE=rsncode─┘
   └─,ACTION=NULL────┘

   ┌─,PLISTVER=IMPLIED_VERSION─┐  ┌─,MF=S──────────────────────────────────┐
►──┼───────────────────────────┼──┼────────────────────────────────────────┼──────►◄
   ├─,PLISTVER=MAX─────────────┤  │                      ┌─,0D───┐         │
   └─,PLISTVER=0───────────────┘  ├─,MF=(L─,list addr──┼───────┼─)───────┤
                                  │                      └─,attr─┘         │
                                  │                      ┌─,COMPLETE─┐     │
                                  └─,MF=(E─,list addr──┴───────────┴─)───┘
```

Parameters

The parameters are explained as follows:

name

An optional symbol, starting in column 1, that is the name on the IWMSCXIT macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

,ACTION=QUIESCE
,ACTION=RESUME
,ACTION=NULL

A required parameter, which indicates the type of action requested. The exit should explicitly check to see which action value was specified, as new values may be introduced in later releases.

,ACTION=QUIESCE

Indicates that the exit is to cause the server space to perform an orderly shutdown and terminate. Disconnect should be invoked during the shutdown phase.

,ACTION=RESUME

Indicates that the exit may ignore a previous request to perform an orderly shutdown and terminate. The server space is now eligible to be chosen via invocation of IWMSRFSV and may resume normal operation.

,ACTION=NULL

Indicates that no exit is to be called. The exit need not check for this action.

CONNTKN=*conntkn*

A required input parameter, which contains the WLM Connect token which was returned for the server by Connect (IWMCONN).

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. If you specify 15, GPR15, REG15, or R15 (within or without parentheses), the value will be left in GPR 15.

To code: Specify the RS-type address of a fullword field, or register (2)-(12) or (15), (GPR15), (REG15), or (R15).

,RSNCODE=*rsncode*
An optional output parameter into which the reason code is to be copied from GPR 0. If you specify 0, 00, GPR0, GPR00, REG0, REG00, or R0 (within or without parentheses), the value will be left in GPR 0.

To code: Specify the RS-type address of a fullword field, or register (0) or (2)-(12), (00), (GPR0), (GPR00), REG0), (REG00), or (R0).

,SRV_MGR_EXIT@=*srv_mgr_exit@*
A required input parameter that contains the address of the Server Manager Connect Exit to be invoked.

To code: Specify the RS-type address, or address in register (2)-(12), of a pointer field.

ABEND codes

None.

Return codes and reason codes

When the IWMSCXIT 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.

Table 51. Return and Reason Codes for the IWMSCXIT Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.

Example

To call a Server Manager Connect exit:

```
         IWMSCXIT                                         X
               SRV_MGR_EXIT@ =CONEXIT@,                   X
               CONNTKN=OWNERCTKN,                         X
               ACTION=QUIESCE,                            X
               RETCODE=RC,RSNCODE=RSN
*
* Storage areas
*
```

```
CONEXIT@   DS  AL4            Contains the address of the
*                             Connect Exit to be invoked
OWNERCTKN  DS  CL4            Contains the connect token for
*                             the server space
*
RC       DS    F              Return code
RSN      DS    F              Reason code
```

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:

Minimum authorization:	Supervisor state or program key mask (PKM) allowing keys 0-7.
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 may be held.
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, 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

The syntax of the IWMSEDES macro is as follows:

```
►►──┬──────┬──IWMSEDES──SCHENV=schenv──,SYSTEM_NAME=system_name──┬──────────────────┬──────►
    └─name─┘                                                    └─,RETCODE=retcode─┘

                          ┌─,PLISTVER=IMPLIED_VERSION─┐
►──┬───────────────────┬──┼───────────────────────────┼──────────────────────────────────────►
   └─,RSNCODE=rsncode──┘  ├─,PLISTVER=MAX─────────────┤
                          └─,PLISTVER=0───────────────┘

   ┌─,MF=S──────────────────────────────────┐
►──┼────────────────────────────────────────┼───────────────────────────────────────────────►◄
   │                        ┌─,0D────┐      │
   ├─,MF=(L─,list addr──────┼────────┼─)────┤
   │                        └─,attr──┘      │
   │                        ┌─,COMPLETE─┐   │
   └─,MF=(E─,list addr──────┴───────────┴─)─┘
```

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 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 codes 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 52. 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 52. Return and Reason Codes for the IWMSEDES Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
4	xxxx0425	**Equate Symbol**: IwmRsnCodeNoSCHENV: **Meaning**: The system does not support scheduling environments services. This return code is set for releases prior to OS/390 Release 4. **Action**: Avoid requesting this function on releases prior to OS/390 Release 4.
4	xxxx0426	**Equate Symbol**: IwmRsnCodeSCHENVNotFound: **Meaning**: The scheduling environment specified by SCHENV does not exist. **Action**: 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.
4	xxxx0427	**Equate Symbol**: IwmRsnCodeSCHENVNotAvailable: **Meaning**: The scheduling environment is not available on the specified system. **Action**: Do not process work that depends upon the scheduling environment being available on the specified system.
4	xxxx042A	**Equate Symbol**: IwmRsnCodeSCHENVNoSystem: **Meaning**: The scheduling environment exists. However, the specified system does not exist. **Action**: Do not process work that depends upon the scheduling environment being available on the specified system.
8	—	IwmRetCodeInvocError: Invalid invocation environment or parameters
8	xxxx0801	**Equate Symbol**: IwmRsnCodeSrbMode: **Meaning**: The caller is in SRB mode. **Action**: Avoid requesting this function while in SRB mode.
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: The caller is locked. **Action**: Avoid requesting this function while locked.
8	xxxx080B	**Equate Symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list. **Action**: Check for possible storage overlay.
8	xxxx0824	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: Caller in 24-bit addressing mode. **Action**: Request this function in 31-bit addressing mode.

Table 52. Return and Reason Codes for the IWMSEDES Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx0825	**Equate Symbol**: IwmRsnCodeAscModeNotPrimary **Meaning**: The caller invoked the service but was not in primary ASC mode. **Action**: Request this function in primary mode.
8	xxxx0827	**Equate Symbol**: IwmRsnCodeRsvdNot0 **Meaning**: Reserved field in parameter list was non-zero. **Action**: Request this function with reserved fields zero.
8	xxxx0828	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: Version number in parameter list or version length field is not valid. **Action**: Request this function with corrent version number.
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error. **Action**: 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
```

IWMSEQRY — Scheduling environments query service

IWMSEQRY 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. IWMSEQRY checks if the specified work area length (ANSLEN) is large enough to receive the output. If so, IWMSEQRY 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 IWMSEQRY returns the required amount of storage in the **QUERYLEN** parameter.

Environment

The requirements for the caller are:

Minimum authorization:	Supervisor state or program key mask (PKM) allowing keys 0-7.
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 may be held.
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, 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 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.

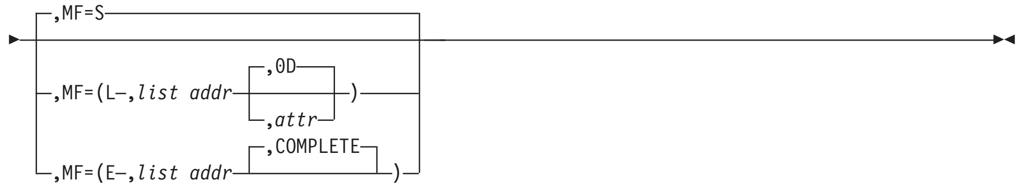
Performance implications

None.

Syntax

The syntax of the IWMSEQRY macro is as follows:

```
►►──┬──────┬──IWMSEQRY──ANSLEN=anslen──,ANSAREA=ansarea──,QUERYLEN=querylen──────────►
    └─name─┘


                                               ┌─,PLISTVER=IMPLIED_VERSION─┐
►──┬─────────────────┬──┬─────────────────┬──┼───────────────────────────┼──────────►
   └─,RETCODE=retcode┘  └─,RSNCODE=rsncode┘  ├─,PLISTVER=MAX─────────────┤
                                               └─,PLISTVER=0───────────────┘
```



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 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 codes 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 53. 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.
4	—	IwmRetCodeWarning: Successful completion, unusual conditions noted.
4	xxxx040A	**Equate Symbol**: IwmRsnCodeOutputAreaTooSmall: **Meaning**: The output area supplied is too small to receive all the available information. **Action**: Obtain a new output area using the length returned in the QUERYLEN parameter and invoke the service again.
4	xxxx0425	**Equate Symbol**: IwmRsnCodeNoSCHENV: **Meaning**: The system does not support scheduling environments services. This return code is set for releases prior to OS/390 Release 4. **Action**: Avoid requesting this function on releases prior to OS/390 Release 4.
4	xxxx0428	**Equate Symbol**: IwmRsnCodeNoSCHENVDefined **Meaning**: No scheduling environments or resources are defined in the active service policy. No data is returned in the output area. **Action**: Do not use the output area.
8	—	IwmRetCodeInvocError: Invalid invocation environment or parameters
8	xxxx0801	**Equate Symbol**: IwmRsnCodeSrbMode: **Meaning**: The caller is in SRB mode. **Action**: Avoid requesting this function while in SRB mode.
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: The caller is locked. **Action**: Avoid requesting this function while locked.
8	xxxx080B	**Equate Symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list. **Action**: Check for possible storage overlay.

Table 53. Return and Reason Codes for the IWMSEQRY Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx0810	**Equate Symbol**: IwmRsnCodeEutFrr **Meaning**: The caller has an EUT FRR established. **Action**: Avoid requesting this function with an EUT FRR set.
8	xxxx0824	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: The caller is in 24-bit addressing mode. **Action**: Request this function in 31-bit addressing mode.
8	xxxx0825	**Equate Symbol**: IwmRsnCodeAscModeNotPrimary **Meaning**: The caller invoked the service but was not in primary ASC mode. **Action**: Request this function in primary mode.
8	xxxx0827	**Equate Symbol**: IwmRsnCodeRsvdNot0 **Meaning**: Reserved field in parameter list was non-zero. **Action**: Request this function with reserved fields zero.
C	—	**Equate Symbol**: IwmRetCodeEnvError **Meaning**: Environmental error.
C	xxxx0C2C	**Equate Symbol**: IwmRsnCodeCannotAccessPolicy **Meaning**: The service cannot access the active policy possibly due to a policy activation in progress. **Action**: The caller can try the service again later, or return an error indication to its caller.
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error. **Action**: 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 ANSLEN, specify:

```
         IWMSEQRY ANSAREA=(R5),
                  ANSLEN=ANSLEN,
                  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
*
```

IWMSESET — Scheduling environments set resource service

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:

Minimum authorization:	Supervisor state or program key mask (PKM) allowing keys 0-7.
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 may be held.
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, 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

The syntax of the IWMSESET macro is as follows:

```
►►──┬──────┬──IWMSESET──RESOURCE=resource──┬─,STATE=ON────┬──┬───────────────────┬──────►
    └─name─┘                               ├─,STATE=OFF───┤  └─,RETCODE=retcode─┘
                                           └─,STATE=RESET─┘

                              ┌─,PLISTVER=IMPLIED_VERSION─┐
►──┬───────────────────┬──────┼───────────────────────────┼───────────────────────────►
   └─,RSNCODE=rsncode──┘      ├─,PLISTVER=MAX─────────────┤
                              └─,PLISTVER=0───────────────┘

   ┌─,MF=S─────────────────────────────────┐
►──┼───────────────────────────────────────┼──────────────────────────────────────────►◄
   │                      ┌─,0D───┐        │
   ├─,MF=(L─,list addr────┼───────┼─)──────┤
   │                      └─,attr─┘        │
   │                      ┌─,COMPLETE─┐    │
   └─,MF=(E─,list addr────┴───────────┴─)──┘
```

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:
- 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 codes 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 54. Return and Reason Codes for the IWMSESET Macro

Return Code	Reason Code	Equate Symbol, Meaning, and Action
0	—	**Equate Symbol**: IwmRetCodeOk: **Meaning**: Successful completion. All requested data returned.
4	—	**Equate Symbol**: IwmRetCodeWarning: **Meaning**: Successful completion, unusual conditions noted.
4	xxxx0425	**Equate Symbol**: IwmRsnCodeNoSCHENV: **Meaning**: The system does not support scheduling environments services. This return code is set for releases prior to OS/390 Release 4. **Action**: Avoid requesting this function on releases prior to OS/390 Release 4.
4	xxxx0429	**Equate Symbol**: IwmRsnCodeResourceNotFound: **Meaning**: The resource specified by RESOURCE does not exist. **Action**: Check the specification of the RESOURCE parameter. If the RESOURCE parameter is correct, check whether the resource is defined in the active service policy.
8	—	**Equate Symbol**: IwmRetCodeInvocError: **Meaning**: Invalid invocation environment or parameters.
8	xxxx0801	**Equate Symbol**: IwmRsnCodeSrbMode: **Meaning**: The caller is in SRB mode. **Action**: Avoid requesting this function while in SRB mode.
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: The caller is disabled. **Action**: Avoid requesting this function while disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: The caller is locked. **Action**: Avoid requesting this function while locked.
8	xxxx080B	**Equate Symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list. **Action**: Check for possible storage overlay.
8	xxxx0824	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: The caller is in 24-bit addressing mode. **Action**: Request this function in 31-bit addressing mode.
8	xxxx0825	**Equate Symbol**: IwmRsnCodeAscModeNotPrimary **Meaning**: The caller invoked the service but was not in primary ASC mode. **Action**: Request this function in primary mode.

Table 54. Return and Reason Codes for the IWMSESET Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx0827	**Equate Symbol**: IwmRsnCodeRsvdNot0 **Meaning**: Reserved field in parameter list was non-zero. **Action**: Request this function with reserved fields zero.
8	xxxx0828	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: Version number in parameter list or version length field is not valid. **Action**: Request this function with correct version number.
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error. **Action**: 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
*
```

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:

Minimum authorization:	Supervisor state or program key mask (PKM) allowing keys 0-7.
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 may be held.
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, 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

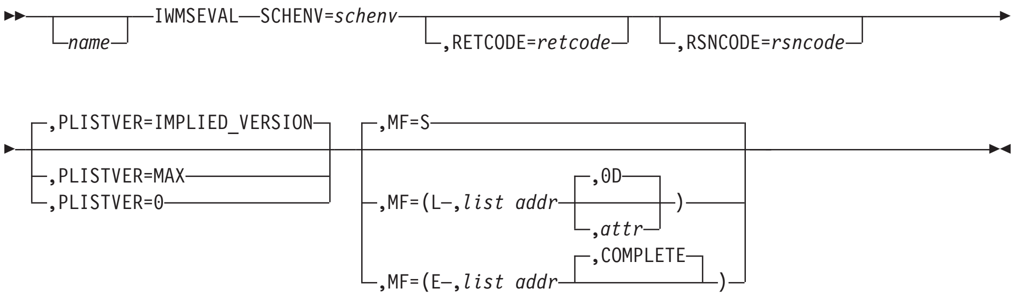
Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

Performance implications

None.

Syntax

The syntax of the IWMSEVAL macro is as follows:



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

,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 codes 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 55. Return and Reason Codes for the IWMSEVAL Macro

Return Code	Reason Code	Equate Symbol, Meaning, and Action
0	—	**Equate Symbol**: IwmRetCodeOk: **Meaning**: Successful completion.
4	—	**Equate Symbol**: IwmRetCodeWarning: **Meaning**: Successful completion, unusual conditions noted.
4	xxxx0425	**Equate Symbol**: IwmRsnCodeNoSCHENV: **Meaning**: The system does not support scheduling environments services. This return code is set for releases prior to OS/390 Release 4. **Action**: Avoid requesting this function on releases prior to OS/390 Release 4.
4	xxxx0426	**Equate Symbol**: IwmRsnCodeSCHENVNotFound: **Meaning**: The scheduling environment specified by SCHENV does not exist. **Action**: 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.

Table 55. Return and Reason Codes for the IWMSEVAL Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	—	**Equate Symbol**: IwmRetCodeInvocError: **Meaning**: Invalid invocation environment or parameters.
8	xxxx0801	**Equate Symbol**: IwmRsnCodeSrbMode: **Meaning**: Caller is in SRB mode. **Action**: Avoid requesting this function while in SRB mode.
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: Caller is locked. **Action**: Avoid requesting this function while locked.
8	xxxx080B	**Equate Symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list. **Action**: Check for possible storage overlay.
8	xxxx0824	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: The caller is in 24-bit addressing mode. **Action**: Request this function in 31 bit addressing mode.
8	xxxx0825	**Equate Symbol**: IwmRsnCodeAscModeNotPrimary **Meaning**: The caller invoked the service but was not in primary ASC mode. **Action**: Request this function in primary mode.
8	xxxx0827	**Equate Symbol**: IwmRsnCodeRsvdNot0 **Meaning**: Reserved field in parameter list was non-zero. **Action**: Request this function with reserved fields zero.
8	xxxx0828	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: Version number in parameter list or version length field is not valid. **Action**: Request this function with corrent version number.
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error. **Action**: 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
```

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:

`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.

`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.

`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.

`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:

Minimum authorization:	Problem state. Any PSW key
Dispatchable unit mode:	Task
Cross memory mode:	PASN=HASN=SASN
AMODE:	31-bit
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	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.

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

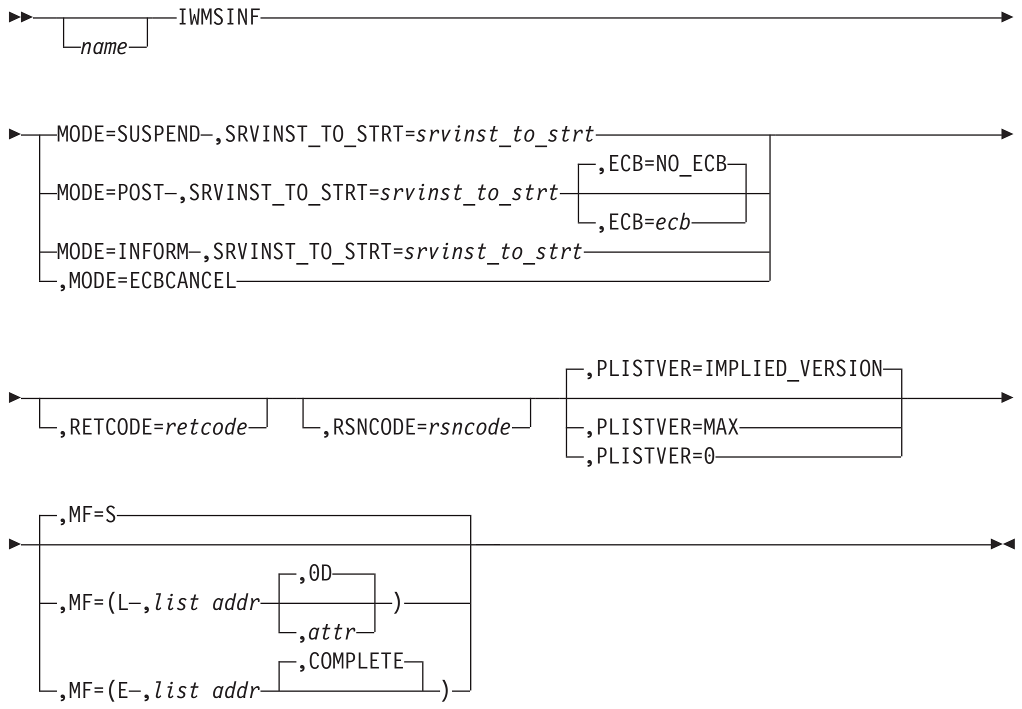
Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

Performance implications

None.

Syntax

The syntax of the IWMSINF macro is as follows:



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, which 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***,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.

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 usefull 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=*srvinst_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=*srvinst_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=*srvinst_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 codes 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 56. Return and Reason Codes for the IWMSINF Macro

Return Code	Reason Code	Equate Symbol, Meaning, and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0801	**Equate Symbol**: IwmRsnCodeSrbMode **Meaning**: The caller is in SRB mode. **Action**: Avoid requesting this function while in SRB mode.
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: Caller is locked. **Action**: Avoid requesting this function while locked.
8	xxxx080B	**Equate Symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list. **Action**: Check for possible storage overlay.
8	xxxx0824	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: The caller invoked the service but was in 24-bit addressing mode. **Action**: Request this function only when you are in 31-bit addressing mode.
8	xxxx0825	**Equate Symbol**: IwmRsnCodeAscModeNotPrimary **Meaning**: The caller invoked the service but was not in primary ASC mode. **Action**: Avoid requesting this function in this environment.

Table 56. Return and Reason Codes for the IWMSINF Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx0828	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: Version number in parameter list is not valid. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx0840	**Equate Symbol**: IwmRsnCodeServiceNotEnabled **Meaning**: Caller's space connection is not enabled for this service **Action**: Make sure that SERVER_MANAGER=YES, SERVER_TYPE=QUEUE and MANAGE_TASKS=YES is specified on the IWM4CON request to enable this service.
8	xxxx0841	**Equate Symbol**: IwmRsnCodeXmemMode **Meaning**: The caller is in cross-memory mode. **Action**: Request this function only when you are not in cross-memory mode.
8	xxxx0842	**Equate Symbol**: IwmRsnCodeNoWLMConnect **Meaning**: The caller's space is not connected to WLM. **Action**: Invoke the IWM4CON macro before invoking this macro.
8	xxxx084D	**Equate Symbol**: IwmRsnCodeNotAuthConnect **Meaning**: 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. **Action**: Avoid requesting this function in this environment.
8	xxxx087B	**Equate Symbol**: IwmRsnCodeUnexpectedCall **Meaning**: The system did not expect the caller to use this service. **Action**: Make sure that MANAGE_TASKS=YES is specified on the IWM4CON request.
C	—	**Equate Symbol**: IwmRetCodeEnvError **Meaning**: Environmental error.
C	xxxx0C14	**Equate Symbol**: IwmRsnCodeNoWorkShutDown **Meaning**: No work selected. The caller is to shutdown. **Action**: Caller must disconnect by invoking the IWM4DIS macro.
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error. **Action**: Contact your system programmer.

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,
          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
```

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:

Minimum authorization:	Problem state with any PSW key.
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 may be held. FRRs may be established.
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, 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

None.

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
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

The syntax of the IWMSRDNS macro is as follows:

```
►►──┬──────┬──IWMSRDNS──LOCATION_NAMES=location_names──,ANSLEN=anslen──────────────►
    └─name─┘

►──┬───────────────────────────┬─,QUERYLEN=querylen──┬──────────────────┬──────────►
   └─,ENTRY_COUNT=entry_count──┘                     └─,RETCODE=retcode─┘

                             ┌─,PLISTVER=IMPLIED_VERSION─┐
►──┬───────────────────┬─────┼───────────────────────────┼─────────────────────────►
   └─,RSNCODE=rsncode──┘     ├─,PLISTVER=MAX─────────────┤
                             └─,PLISTVER=0───────────────┘

   ┌─,MF=S─────────────────────────────────┐
►──┼───────────────────────────────────────┼───────────────────────────────────►◄
   │                     ┌─,0D────┐        │
   ├─,MF=(L─,list addr───┼────────┼─)──────┤
   │                     └─,attr──┘        │
   │                     ┌─,COMPLETE─┐     │
   └─,MF=(E─,list addr───┴───────────┴─)───┘
```

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 codes 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 57. Return and Reason Codes for the IWMSRDNS Macro

Return Code	Reason Code	Equate Symbol, Meaning, and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.
4	xxxx040A	**Equate Symbol**: IwmRsnCodeOutputAreaTooSmall **Meaning**: The output area supplied is too small to receive all the available information. **Action**: None required. If necessary, reinvoke the service with an output area of sufficient size to receive all information.
4	xxxx040B	**Equate Symbol**: IwmRsnCodeNoServersRegistered **Meaning**: No servers have registered in the sysplex. **Action**: None required.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: The caller is locked. **Action**: Avoid requesting this function while locked.
8	xxxx080B	**Equate Symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list. **Action**: Check for possible storage overlay.
8	xxxx0823	**Equate Symbol**: IwmRsnCodeDatoff **Meaning**: The caller invoked the service while DATOFF **Action**: Avoid requesting this function in this environment.

Table 57. Return and Reason Codes for the IWMSRDNS Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx0824	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: The caller invoked the service but was in 24-bit addressing mode. **Action**: Request this function only when you are in 31-bit addressing mode.
8	xxxx0825	**Equate Symbol**: IwmRsnCodeAscModeNotPrimary **Meaning**: The caller invoked the service but was not DAT on Primary ASC mode. **Action**: Avoid requesting this function in this environment.
C	—	**Equate Symbol**: IwmRetCodeEnvError **Meaning**: Environmental error.
C	xxxx0C01	**Equate Symbol**: IwmRsnCodeNoStg **Meaning**: Storage is not available for the request. **Action**: There is a storage shortage. The function may work successfully at a later time.
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error. **Action**: No action required. The function may be successful if invoked again.

Example

To list all locations registered with a host name, specify:

```
         IWMSRDNS LOCATION_NAMES=DATA,
                  ANSLEN=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
```

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.

Environment

The requirements for the caller are:

Minimum authorization:	Problem state with any PSW key if the server address space to be deregistered is the home address space. If resource BPX.WLMSERVER is defined in the FACILITY class, an unauthorized caller requires access authority to this resource or the IWM.SERVER.REGISTER resource in the FACILITY class. If the server to be deregistered is not the home address space, one of the following: • Supervisor state. • Program key mask (PKM) allowing at least one of the keys 0-7. • The caller has access authority to the resource IWM.SERVER.REGISTER in the FACILITY class. If this resource is not defined, access authority to the FACILITY class resource BPX.WLMSERVER is required.
Dispatchable unit mode:	Task or SRB
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. FRRs may be established.
Control parameters:	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

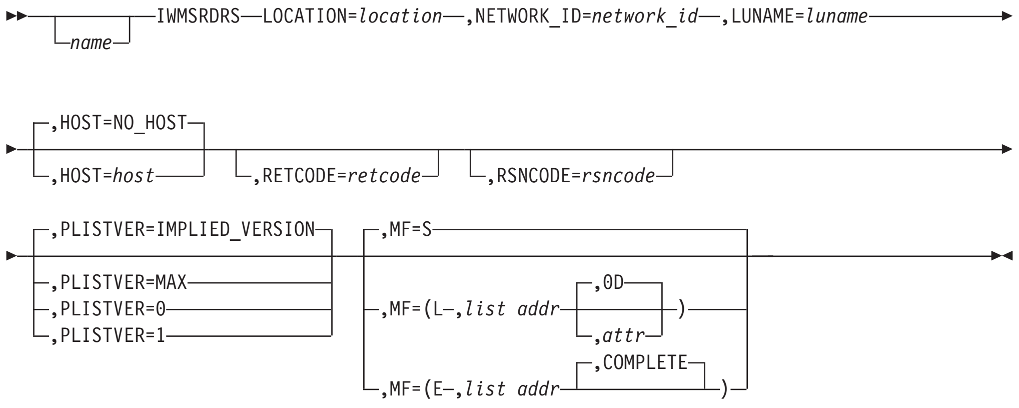
Some callers depend on register contents remaining the same before and after using a service. If the system changes the contents of registers on which the caller depends, the caller must save them before calling the service, and restore them after the system returns control.

Performance implications

None.

Syntax

The syntax of the IWMSRDRS macro is as follows:



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.

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 the following parameter and those from version 0:

  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 codes 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 58. Return and Reason Codes for the IWMSRDRS Macro

Return Code	Reason Code	Equate Symbol, Meaning, and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion.
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.
4	xxxx0418	**Equate Symbol**: IwmRsnCodeServerNotRegistered **Meaning**: Server not registered
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: Caller is locked.
8	xxxx080B	**Equate Symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list
8	xxxx0823	**Equate Symbol**: IwmRsnCodeDatoff **Meaning**: The caller invoked the service while DATOFF
8	xxxx0824	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: The caller invoked the service but was in 24-bit addressing mode.
8	xxxx0825	**Equate Symbol**: IwmRsnCodeAscModeNotPrimary **Meaning**: The caller invoked the service but was not DAT on Primary ASC mode.
8	xxxx0828	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: Version number in parameter list or version length field is not valid.
8	xxxx0829	**Equate Symbol**: IwmRsnCodeBadOptions **Meaning**: 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. **Action**: Check for possible storage overlay of the parameter list.

Table 58. Return and Reason Codes for the IWMSRDRS Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
C	—	**Equate Symbol**: IwmRetCodeEnvError **Meaning**: Environmental error.
C	xxxx0C35	**Equate Symbol**: IwmRsnCodeNotSecAuthServReg **Meaning**: The caller is not authorized by SAF to deregister a service. **Action**: Unauthorized callers (problem state or none of the authorized keys 0 to 7) require access authority to the RACF resource IWM.SERVER.REGISTER in the FACILITY class. If this resource is not defined, access authority to the FACILITY class resource BPX.WLMSERVER is required.
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error (no reason codes)

IWMSRFSV — Sysplex routing find server routine

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:

Minimum authorization:	Supervisor state or program key mask (PKM) allowing keys 0-7.
Dispatchable unit mode:	Task
Cross memory mode:	PASN=HASN=SASN
AMODE:	31-bit
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks may be held.
Control parameters:	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.
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

```
►►──┬──────┬──b──IWMSRFSV──b──CONNTKN=conntkn──,APPLENV=applenv──────────────►
    └─name─┘
```

```
►─,SERVER_DATA=server_data──┬──────────────────┬──┬──────────────────┬────────────────►
                            └─,RETCODE=retcode─┘  └─,RSNCODE=rsncode─┘

   ┌─,PLISTVER=IMPLIED_VERSION─┐  ┌─,MF=S──────────────────────────────────────┐
►──┼───────────────────────────┼──┼────────────────────────────────────────────┼────────►◄
   ├─,PLISTVER=MAX─────────────┤  │                          ┌─,0D───┐         │
   └─,PLISTVER=0───────────────┘  ├─,MF=(L─,list addr───────┼───────┼─)───────┤
                                  │                          └─,attr─┘         │
                                  │                          ┌─,COMPLETE─┐     │
                                  └─,MF=(E─,list addr────────┴───────────┴─)───┘
```

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***,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. If you specify 15, GPR15, REG15, or R15 (within or without parentheses), the value will be left in GPR 15.

To code: Specify the RS-type address of a fullword field, or register (2)-(12) or (15), (GPR15), (REG15), or (R15).

,RSNCODE=*rsncode*
An optional output parameter into which the reason code is to be copied from GPR 0. If you specify 0, 00, GPR0, GPR00, REG0, REG00, or R0 (within or without parentheses), the value will be left in GPR 0.

To code: Specify the RS-type address of a fullword field, or register (0) or (2)-(12), (00), (GPR0), (GPR00), REG0), (REG00), or (R0).

,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 codes and reason codes

When the IWMSRFSV 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 59. Return and Reason Codes for the IWMSRFSV Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0801	**Equate Symbol**: IwmRsnCodeSRBMode **Meaning**: Caller is in SRB mode. **Action**: Avoid requesting this function in SRB mode.
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: Caller is locked. **Action**: Avoid requesting this function while locked.
8	xxxx080B	**Equate Symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list. **Action**: Check for possible storage overlay.
8	xxxx0810	**Equate Symbol**: IwmRsnCodeEutFrr **Meaning**: Caller has EUT FRR established. **Action**: Avoid requesting this function with an EUT FRR.

Table 59. Return and Reason Codes for the IWMSRFSV Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0823	**Equate Symbol**: IwmRsnCodeDatoff **Meaning**: Caller invoked service while DATOFF **Action**: Avoid requesting this function in this environment.
8	xxxx0824	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: Caller invoked service but was in 24 bit addressing mode. **Action**: Request this function only when you are in 31 bit addressing mode.
8	xxxx0825	**Equate Symbol**: IwmRsnCodeAscModeNotPrimary **Meaning**: Caller invoked service but was not DAT on Primary ASC mode. **Action**: Avoid requesting this function in this environment.
8	xxxx0826	**Equate Symbol**: IwmRsnCodeTaskTerm **Meaning**: Caller invoked service while task termination is in progress for the current TCB. **Action**: Avoid requesting this function in this environment.
8	xxxx0828	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: Version number or version length field in parameter list is not valid. **Action**: Check for possible overlay of the parameter list.
8	xxxx083B	**Equate Symbol**: IwmRsnCodeHomeNotOwnConn **Meaning**: Home address space does not own the connect token from the input parameter list. **Action**: Invoke the function with the correct home address space.
8	xxxx0840	**Equate Symbol**: IwmRsnCodeServiceNotEnabled **Meaning**: Caller's space connection is not enabled for this service. **Action**: Avoid requesting this function under the input connection. IWM4CON options must be specified previously to enable this service.
8	xxxx0841	**Equate Symbol**: IwmRsnCodeXmemMode **Meaning**: Caller invoked service but was in cross-memory mode. **Action**: Avoid requesting this function in cross-memory mode.
8	xxxx0842	**Equate Symbol**: IwmRsnCodeNoWLMConnect **Meaning**: Caller's space is not connected to WLM. **Action**: Issue IWM4CON with the necessary options prior to invoking this service.
C	—	**Equate symbol:** IwmRetCodeEnvError **Meaning:** Environmental error.

Table 59. Return and Reason Codes for the IWMSRFSV Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
C	xxxx0C01	**Equate symbol:** IwmRsnCodeNoStg **Meaning:** Storage is not available for the request. **Action:** There is a storage shortage. The function may work successfully at a later time.
C	xxxx0C1A	**Equate symbol:** IwmRsnCodeApplNotDefined **Meaning:** The application environment name is not defined in the active WLM policy. **Action:** 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.
C	xxxx0C1B	**Equate symbol:** IwmRsnCodeApplNotSST **Meaning:** The application environment name is defined for use by a different subsystem type in the active WLM policy. **Action:** 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.
C	xxxx0C1C	**Equate symbol:** IwmRsnCodeServerNotStarted **Meaning:** No server exists for the specified application environment and no server could be started. **Action:** No action required. The function may be successful if invoked again.
C	xxxx0C22	**Equate symbol:** IwmRsnCodeApplEnvQuiesced **Meaning:** The specified application environment has been quiesced, server cannot be started for the request. **Action:** Restart the application environment and then retry the request.
C	xxxx0C23	**Equate symbol:** IwmRsnCodeIndLocalSystem **Meaning:** Local system is not running with the current WLM policy, new server cannot be started for the request. **Action:** Avoid requesting this function while the local system is not running with the current WLM policy.
C	xxxx0C24	**Equate symbol:** IwmRsncodeProcNameBlank **Meaning:** Server procname is blank, server cannot be started for the request. **Action:** Check the server procname, fix it, and then retry the request.

Table 59. Return and Reason Codes for the IWMSRFSV Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
C	xxxx0C25	**Equate symbol:** IwmRsnCodeApplEnvStopped **Meaning:** WLM has given up trying to start a server because of failures. The associated application environment has been internally stopped. **Action:** Restart the application environment and then retry the request.
C	xxxx0C26	**Equate symbol:** IwmRsnCodeRouterNotActive **Meaning:** 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. **Action:** Re-connect the router for the requested application environment to WLM and then retry the request.
C	xxxx0C27	**Equate symbol:** IwmRsnCodeFsvReqInCompat **Meaning:** Reserved.
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error. **Action**: No action required. The function may be successful if invoked again.

Example

To determine a best server to which to route work:

```
        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
```

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.

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 or to obtain the user data associated with each server. 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:

Minimum authorization:	Problem state with any PSW key if the address space token specified with STOKEN=stoken equals the address space token of the home address space. If resource BPX.WLMSERVER is defined in the FACILITY class, an unauthorized caller requires access authority to this resource or the IWM.SERVER.REGISTER resource in the FACILITY class. If the address space token specified with STOKEN=stoken is not the address space token of the home address space, one of the following: • Supervisor state. • Program key mask (PKM) allowing at least one of the keys 0-7. • The caller has access authority to the resource IWM.SERVER.REGISTER in the FACILITY class. If this resource is not defined, access authority to the FACILITY class resource BPX.WLMSERVER is required.
Dispatchable unit mode:	Task or SRB
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 may be held. FRRs may be established.
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, 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 not be used prior to the completion of WLM address space initialization.

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

The syntax of the IWMSRSRG macro is as follows:

```
►►─┬──────┬─IWMSRSRG──LOCATION=location──,NETWORK_ID=network_id──,LUNAME=luname──────────►
   └─name─┘

                   ┌─,USERDATA=NO_USERDATA─┐  ┌─,HOST=NO_HOST─┐  ┌─,HEALTH=NO_HEALTH─┐
►─,STOKEN=stoken───┼───────────────────────┼──┼───────────────┼──┼───────────────────┼──────►
                   └─,USERDATA=userdata────┘  └─,HOST=host────┘  └─,HEALTH=health────┘
```

```
                                                         ┌─,PLISTVER=IMPLIED_VERSION─┐
►──┬─────────────────────┬──┬─────────────────────┬──┼───────────────────────────┼──────────────►
   └─,RETCODE=retcode────┘  └─,RSNCODE=rsncode────┘  ├─,PLISTVER=MAX─────────────┤
                                                         ├─,PLISTVER=0───────────────┤
                                                         ├─,PLISTVER=1───────────────┤
                                                         ├─,PLISTVER=2───────────────┤
                                                         └─,PLISTVER=3───────────────┘

   ┌─,MF=S────────────────────────────────────┐
►──┼──────────────────────────────────────────┼──────────────────────────────────────────────►◄
   │                          ┌─,0D────┐      │
   ├─,MF=(L─,list addr────────┼────────┼─)────┤
   │                          └─,attr──┘      │
   │                          ┌─,COMPLETE─┐   │
   └─,MF=(E─,list addr────────┴───────────┴─)─┘
```

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

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).

,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.

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 codes 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 60. Return and Reason Codes for the IWMSRSRG Macro

Return Code	Reason Code	Equate Symbol, Meaning, and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.
4	xxxx0419	**Equate Symbol**: IwmRsnCodeServerAlreadyReg **Meaning**: Server already registered. **Action**: None required.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: Caller is locked. **Action**: Avoid requesting this function while locked.
8	xxxx0807	**Equate Symbol**: IwmRsnCodeBadSTOKEN **Meaning**: Bad STOKEN passed. **Action**: Check for possible storage overlay.
8	xxxx080B	**Equate Symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list. **Action**: Check for possible storage overlay.
8	xxxx0823	**Equate Symbol**: IwmRsnCodeDatoff **Meaning**: The caller invoked the service while DATOFF **Action**: Avoid requesting this function in this environment.
8	xxxx0824	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: The caller invoked the service but was in 24-bit addressing mode. **Action**: Request this function only when you are in 31-bit addressing mode.

Table 60. Return and Reason Codes for the IWMSRSRG Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx0825	**Equate Symbol**: IwmRsnCodeAscModeNotPrimary **Meaning**: The caller invoked the service but was not DAT on Primary ASC mode. **Action**: Avoid requesting this function in this environment.
8	xxxx0828	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: Version number in parameter list or version length field is not valid. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx08A2	**Equate Symbol**: IwmRsnCodeBadHealth **Meaning**: Health value out of range **Action**: Check for possible storage overlay.
C	—	**Equate Symbol**: IwmRetCodeEnvError **Meaning**: Environmental error.
C	xxxx0C01	**Equate Symbol**: IwmRsnCodeNoStg **Meaning**: Storage is not available for the request. **Action**: There is a storage shortage. The function may work successfully at a later time.
C	xxxx0C35	**Equate Symbol**: IwmRsnCodeNotSecAuthServReg **Meaning**: The caller is not authorized by SAF to register a server. **Action**: The caller is not authorized by SAF to register a server. Action: Invoke the function with the authorization requirements fulfilled. Unauthorized callers (problem state or none of the authorized keys 0 to 7) require access authority to the RACF resource IWM.SERVER.REGISTER in the FACILITY class. If this resource is not defined, access authority to the FACILITY class resource BPX.WLMSERVER is required.
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error (no reason codes).

Example

To register an instance of a given location, specify:

```
         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
```

IWMSRSRS — Sysplex routing information

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    CPU Weight    zAAP Weight    zIIP Weight
------            ------    ----------    -----------    -----------
NETIDA.LUNAME1       4          3              6              4
NETIDB.LUNAME2       7          8              3              9
NETIDC.LUNAME3       1          3              1              0
NETIDD.LUNAME43      4          5              2              3
NETID4.LUNAME2       2          1              2              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.

Starting with z/OS V1R9, three new output weights are available: the CPU weight, the zAAP weight and the zIIP weight. The CPU weight is computed the same way as the weight was prior to V1R9, taking only CPU data into account. The zAAP and the zIIP weights are computed when taking only zAAP, respectively zIIP, data into account. The weight is a combination of these three processor weights.

Starting with z/OS V1R11, an optional input keyword **METHOD** selects how the weight (also referred to as "mixed" weight) is computed. The default is `METHOD=PROPORTIONAL`, which calculates the weight as a combination of these 3 processor weights (CPU, zAAP and zIIP). It is the same method as prior to V1R11. With `METHOD=EQUICPU`, WLM computes the weight by trying to simulate a 100% usage of the system capacity, and determining the capacity of a CPU-only system having equivalent resource consumption.

Both methods can be specified with keyword **IL_WEIGHTING**, and EQUICPU by COST_ZAAP_ON_CP and COST_ZIIP_ON_CP, too.

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.

When using the QUERY function, the CPU, zAAP and zIIP weights are always set to 0.

When using the SELECT or the SPECIFIC function, the mixed weight is a combination of the CPU, zAAP and zIIP weights with the relative use of the CPU, zAAP and zIIP by the server. Moreover, the mixed weights are scaled up, so that their sum is 64. Due to rounding errors, the sum of the mixed weights is usually as low as 64-(number of servers).

If there are pre-V1R9 systems in the sysplex, the zAAP and zIIP weights are automatically set to 0, and the weight is equal to the CPU weight, because pre-V1R9 systems do not have such weights and could not be correctly compared to V1R9 systems.

The Sysplex Routing Service (IWMSRSRS) can be used to obtain the userdata associated with each server.

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. In contrast to the weights in the SELECT function, the weights in the SPECIFIC function do not only consider available capacity values and the number of servers on the same system.

With the SPECIFIC function, the following 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.

Environment

The requirements for the caller are:

Minimum authorization:	Problem state with any PSW key.
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 may be held. FRRs may be established.
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, 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 supports multiple versions. Some keywords are unique to certain versions. See the PLISTVER parameter description.

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

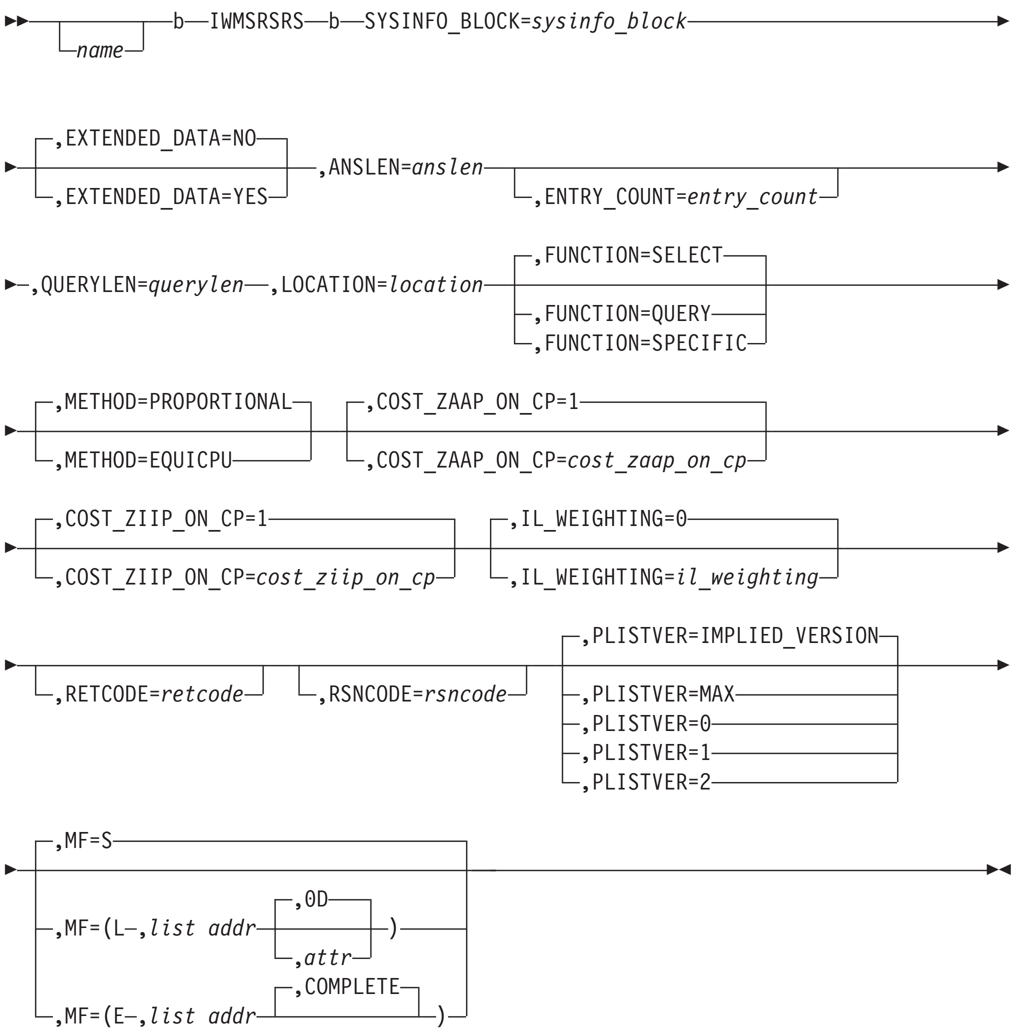
Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

Performance implications

None.

Syntax

main diagram



```
►►──┬──────┬──b──IWMSRSRS──b──SYSINFO_BLOCK=sysinfo_block──────────────►
    └─name─┘

   ┌─,EXTENDED_DATA=NO──┐
►──┼────────────────────┼──,ANSLEN=anslen──┬──────────────────────────┬──►
   └─,EXTENDED_DATA=YES─┘                  └─,ENTRY_COUNT=entry_count─┘

                                          ┌─,FUNCTION=SELECT───┐
►──,QUERYLEN=querylen──,LOCATION=location──┼────────────────────┼──────►
                                          ├─,FUNCTION=QUERY────┤
                                          └─,FUNCTION=SPECIFIC─┘

   ┌─,METHOD=PROPORTIONAL─┐  ┌─,COST_ZAAP_ON_CP=1───────────────┐
►──┼──────────────────────┼──┼──────────────────────────────────┼──────►
   └─,METHOD=EQUICPU──────┘  └─,COST_ZAAP_ON_CP=cost_zaap_on_cp─┘

   ┌─,COST_ZIIP_ON_CP=1───────────────┐  ┌─,IL_WEIGHTING=0────────────┐
►──┼──────────────────────────────────┼──┼────────────────────────────┼──►
   └─,COST_ZIIP_ON_CP=cost_ziip_on_cp─┘  └─,IL_WEIGHTING=il_weighting─┘

                                            ┌─,PLISTVER=IMPLIED_VERSION─┐
►──┬──────────────────┬──┬──────────────────┬──┼───────────────────────────┼──►
   └─,RETCODE=retcode─┘  └─,RSNCODE=rsncode─┘  ├─,PLISTVER=MAX─────────────┤
                                            ├─,PLISTVER=0───────────────┤
                                            ├─,PLISTVER=1───────────────┤
                                            └─,PLISTVER=2───────────────┘

   ┌─,MF=S────────────────────────────────────┐
►──┼──────────────────────────────────────────┼────────────────────────►◄
   │                         ┌─,0D───┐        │
   ├─,MF=(L─,list addr───────┼───────┼──)─────┤
   │                         └─,attr─┘        │
   │                         ┌─,COMPLETE─┐    │
   └─,MF=(E─,list addr───────┴───────────┴──)─┘
```

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, or specify a literal decimal value.

,COST_ZAAP_ON_CP=*cost_zaap_on_cp*
,COST_ZAAP_ON_CP=1

An optional input parameter, which is used in conjunction with METHOD=EQUICPU. It describes the additional cost of executing zAAP-eligible work on a CPU instead of on a zAAP processor.

If the caller wants to use the full system capacity, independently of the cost, then it should set COST_ZAAP_ON_CP=1. With high values of this cost parameter, WLM considers that a system having used up its free zAAP capacity should offload less work to the CPU, and gives this system a smaller output weight.

This cost parameter must be in the range from 1 to 100. If the specified value is outside of this range, WLM will instead use the nearest valid integer (1 or 100) as cost parameter. The default is 1.

To code: Specify the RS-type address, or address in register (2)-(12), of a fullword field, or specify a literal decimal value.

,COST_ZIIP_ON_CP=*cost_ziip_on_cp*
,COST_ZIIP_ON_CP=1

An optional input parameter, which is used in conjunction with METHOD=EQUICPU. It describes the additional cost of executing zIIP-eligible work on a CPU instead of on a zIIP processor.

If the caller wants to use the full system capacity, independently of the cost, then it should set COST_ZIIP_ON_CP=1. With high values of this cost parameter, WLM considers that a system having used up its free zIIP capacity should offload less work to the CPU, and gives this system a smaller output weight.

This cost parameter must be in the range from 1 to 100. If the specified value is outside of this range, WLM will instead use the nearest valid integer (1 or 100) as cost parameter. The default is 1.

To code: Specify the RS-type address, or address in register (2)-(12), of a fullword field, or specify a literal decimal value.

,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).

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.

,FUNCTION=SELECT
,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.

,FUNCTION=SPECIFIC

indicates that all servers in the sysplex which are associated with the input Location name along with a relative weighting for each server are returned.

,IL_WEIGHTING=*il_weighting*
,IL_WEIGHTING=0

An optional input parameter, which controls how WLM weights available capacity at importance levels (ILs) lower than the currently selected one. The value of IL_WEIGHTING should be in the range from 0 to 3. If the passed value is outside of this range, WLM will instead use the nearest valid integer (0 or 3) as IL_WEIGHTING.

When this parameter is set to 0 (the default value), all free capacities used by levels less important than the current one are weighted the same. This means that the free capacity below the current level is considered to be totally free, and this is equivalent to what WLM did prior to V1R11.

When this parameter is set to 1, free capacity at the lowest ILs is weighted more than the current IL, with a weighting growing proportionally to the square root of the IL difference + 1. For example , with a selected IL of 1, free capacity at IL 5 is weighted about 2.236 times more than free capacity at IL 1.

When this parameter is set to 2, free capacity at the lowest ILs is weighted more than the current IL, with a weighting growing proportionally to the IL difference + 1. For example , with a selected IL of 1, free capacity at IL 5 is weighted 5 times more than free capacity at IL 1.

When this parameter is set to 3, free capacity at the lowest ILs is weighted more than the current IL, with a weighting growing proportionally to the square of the IL difference + 1. For example, with a selected IL of 1, free capacity at IL 5 is weighted 25 times more than free capacity at IL 1. The default is 0.

To code: Specify the RS-type address, or address in register (2)-(12), of a fullword field, or specify a literal decimal value.

,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.

,METHOD=PROPORTIONAL

,METHOD=EQUICPU

An optional parameter, which selects the method for computing the output weights. The default is METHOD=PROPORTIONAL. The default is METHOD=PROPORTIONAL.

,METHOD=PROPORTIONAL

Calculates the output weights as a proportion of the 3 processor types weights.

,METHOD=EQUICPU

Calculates a CPU equivalent of the systems before computing the output weights.

In order for METHOD=EQUICPU to be active, all systems in sysplex must run at least z/OS V1R11. Otherwise WLM automatically switches back to METHOD=PROPORTIONAL.

,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:

ENTRY_COUNT	EXTENDED_DATA	FUNCTION

- **2**, which supports both the following parameters and those from version 0 and 1:

COST_ZAAP_ON_CP	IL_WEIGHTING
COST_ZIIP_ON_CP	METHOD

To code: Specify one of the following:

- IMPLIED_VERSION
- MAX
- A decimal value of 0, 1, or 2

,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. If you specify 15, GPR15, REG15, or R15 (within or without parentheses), the value will be left in GPR 15.

To code: Specify the RS-type address of a fullword field, or register (2)-(12) or (15), (GPR15), (REG15), or (R15).

,RSNCODE=*rsncode*

An optional output parameter into which the reason code is to be copied from

GPR 0. If you specify 0, 00, GPR0, GPR00, REG0, REG00, or R0 (within or without parentheses), the value will be left in GPR 0.

To code: Specify the RS-type address of a fullword field, or register (0) or (2)-(12), (00), (GPR0), (GPR00), REG0), (REG00), or (R0).

SYSINFO_BLOCK=*sysinfo_block*
A required input parameter, 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 codes and reason codes

When the IWMSRSRS 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 61. Return and Reason Codes for the IWMSRSRS Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.
4	xxxx040A	**Equate Symbol**: IwmRsnCodeOutputAreaTooSmall **Meaning**: The output area supplied is too small to receive all the available information. **Action**: None required. If necessary, reinvoke the service with an output area of sufficient size to receive all information.
4	xxxx040B	**Equate Symbol**: IwmRsnCodeNoServersRegistered **Meaning**: No Servers have registered in the sysplex. **Action**: None required.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.

Table 61. Return and Reason Codes for the IWMSRSRS Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: Caller is locked. **Action**: Avoid requesting this function while locked.
8	xxxx080B	**Equate Symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list. **Action**: Check for possible storage overlay.
8	xxxx0823	**Equate Symbol**: IwmRsnCodeDatoff **Meaning**: Caller invoked service while DATOFF **Action**: Avoid requesting this function in this environment.
8	xxxx0824	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: Caller invoked service but was in 24 bit addressing mode. **Action**: Request this function only when you are in 31 bit addressing mode.
8	xxxx0825	**Equate Symbol**: IwmRsnCodeAscModeNotPrimary **Meaning**: Caller invoked service but was not DAT on Primary ASC mode. **Action**: Avoid requesting this function in this environment.
8	xxxx0828	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: Version number in parameter list or version length field is not valid. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx089D	**Equate Symbol**: IwmRsnCodeWrongSysLevels **Meaning**: There are servers registered in the sysplex, associated with the input Location name, but with a too old z/OS level (prior to V1R7) for the SPECIFIC function in the routing service. **Action**: You may either deregister the servers with the old level, or decide to use IWMSRSRS with function SELECT instead of SPECIFIC.
C	—	**Equate Symbol**: IwmRetCodeEnvError **Meaning**: Environmental error.
C	xxxx0C01	**Equate Symbol**: IwmRsnCodeNoStg **Meaning**: Storage is not available for the request. **Action**: There is a storage shortage. The function may work successfully at a later time.

Table 61. Return and Reason Codes for the IWMSRSRS Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error. **Action**: No action required. The function may be successful if invoked again.

Example

To register an instance of a given location:

```
        IWMSRSRS SYSINFO_BLOCK=DATA,
                 EXTENDED_DATA=YES,
                 ANSLEN=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
```

IWMUEXPT — WLM undo export

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:

Minimum authorization:	Supervisor state or program key mask (PKM) allowing keys 0-7.
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any HASN, any SASN. PASN must be the address space which connected to WLM.
AMODE:	31-bit
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks may be held.
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, 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:

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

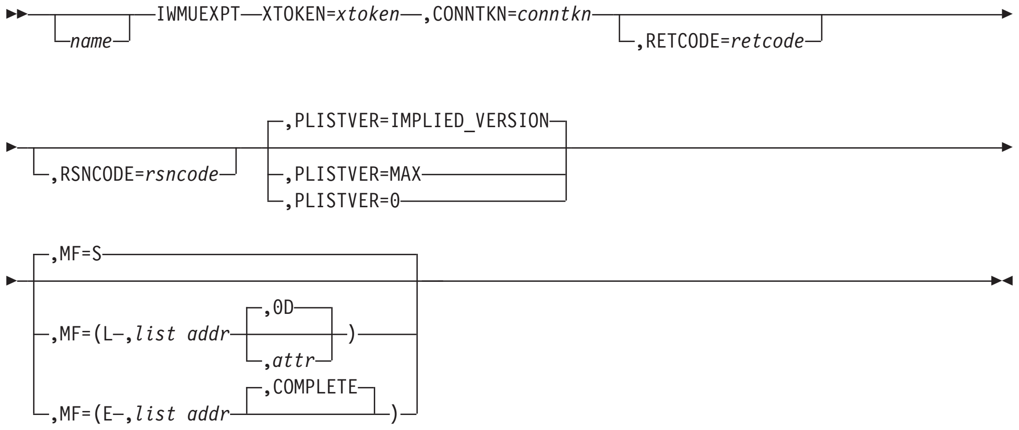
Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

Performance implications

None.

Syntax

The syntax of the IWMUEXPT macro is as follows:



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***,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).

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 codes 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 62. Return and Reason Codes for the IWMUEXPT Macro

Return Code	Reason Code	Equate Symbol, Meaning, and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.

Table 62. Return and Reason Codes for the IWMUEXPT Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.
4	xxxx0432	**Equate Symbol**: IwmRsnCodeUnknownExportToken **Meaning**: 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. **Action**: None.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: The caller is locked. **Action**: Avoid requesting this function while locked.
8	xxxx080B	**Equate Symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list. **Action**: Check for possible storage overlay.
8	xxxx0821	**Equate Symbol**: IwmRsnCodeBadConn **Meaning**: Input connect token does not pass validity checking. **Action**: 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	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: The caller invoked the service but was in 24-bit addressing mode. **Action**: Request this function only when you are in 31-bit addressing mode.
8	xxxx0825	**Equate Symbol**: IwmRsnCodeAscModeNotPrimary **Meaning**: The caller invoked the service but was not in primary ASC mode. **Action**: Avoid requesting this function in this environment.
8	xxxx0827	**Equate Symbol**: IwmRsnCodeRsvdNot0 **Meaning**: Reserved field in parameter list was non-zero. **Action**: Check for use of keywords that are not supported by the MVS release on which the program is running.
8	xxxx0828	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: Version number in parameter list is not valid. **Action**: Check for possible storage overlay of the parameter list.

Table 62. Return and Reason Codes for the IWMUEXPT Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx0840	**Equate Symbol**: IwmRsnCodeServiceNotEnabled **Meaning**: The caller's connection is not enabled for this service. **Action**: Make sure that EXPTIMPT=YES is specified on the IWM4CON macro invocation.
8	xxxx0842	**Equate Symbol**: IwmRsnCodeNoWLMConnect **Meaning**: The caller's primary address space is not connected to WLM. **Action**: Invoke the IWM4CON macro before invoking this macro.
8	xxxx0870	**Equate Symbol**: IwmRsnCodeBadExportToken **Meaning**: The export token is not validly formatted. **Action**: Check for possible storage overlay of the export token.
8	xxxx0871	**Equate Symbol**: IwmRsnCodeDidNotExportOrImport **Meaning**: The primary address space did not export the enclave. **Action**: Invoke IWMUEXPT from the correct address space.
C	—	**Equate Symbol**: IwmRetCodeEnvError **Meaning**: Environmental error.
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error. **Action**: Contact your system programmer.

Example

```
         IWMUEXPT XTOKEN=EXPORTT,CONNTKN=CONNECTT
*
* Storage areas
*
EXPORTT  DS    CL32            Export token
CONNECTT DS    CL4             Connect token
```

IWMUIMPT — WLM undo import

The IWMUIMPT macro undoes an earlier request to import an enclave via the IWMIMPT macro.

Environment

The requirements for the caller are:

Minimum authorization:	Supervisor state or program key mask (PKM) allowing keys 0-7.
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any HASN, any SASN. PASN must be the address space which connected to WLM.
AMODE:	31-bit
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks may be held.
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, 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

The syntax of the IWMUIMPT macro is as follows:

```
>>─┬──────┬─IWMUIMPT──XTOKEN=xtoken──,CONNTKN=conntkn─┬──────────────────┬──────>
   └─name─┘                                           └─,RETCODE=retcode─┘

                           ┌─,PLISTVER=IMPLIED_VERSION─┐
>─┬──────────────────┬─────┼───────────────────────────┼─────────────────────────>
  └─,RSNCODE=rsncode─┘     ├─,PLISTVER=MAX─────────────┤
                           └─,PLISTVER=0───────────────┘

   ┌─,MF=S────────────────────────────────┐
>──┼──────────────────────────────────────┼──────────────────────────────────><
   │                         ┌─,0D────┐   │
   ├─,MF=(L─,list addr───────┼────────┼─)─┤
   │                         └─,attr──┘   │
   │                      ┌─,COMPLETE─┐   │
   └─,MF=(E─,list addr────┴───────────┴─)─┘
```

Parameters

The parameters are explained as follows:

name

An optional symbol, starting in column 1, that is the name on the IWMUIMPT 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*,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).

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 codes and reason codes

When the IWMUIMPT 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 63. Return and Reason Codes for the IWMUIMPT Macro

Return Code	Reason Code	Equate Symbol, Meaning, and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.
4	xxxx0432	**Equate Symbol**: IwmRsnCodeUnknownExportToken **Meaning**: 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. **Action**: None.

Table 63. Return and Reason Codes for the IWMUIMPT Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: The caller is locked. **Action**: Avoid requesting this function while locked.
8	xxxx080B	**Equate Symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list. **Action**: Check for possible storage overlay.
8	xxxx0821	**Equate Symbol**: IwmRsnCodeBadConn **Meaning**: Input connect token does not pass validity checking. **Action**: 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	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: The caller invoked the service but was in 24-bit addressing mode. **Action**: Request this function only when you are in 31-bit addressing mode.
8	xxxx0825	**Equate Symbol**: IwmRsnCodeAscModeNotPrimary **Meaning**: The caller invoked the service but was not in primary ASC mode. **Action**: Avoid requesting this function in this environment.
8	xxxx0827	**Equate Symbol**: IwmRsnCodeRsvdNot0 **Meaning**: Reserved field in parameter list was non-zero. **Action**: Check for use of keywords that are not supported by the MVS release on which the program is running.
8	xxxx0828	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: Version number in parameter list is not valid. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx0840	**Equate Symbol**: IwmRsnCodeServiceNotEnabled **Meaning**: The caller's connection is not enabled for this service. **Action**: Make sure that EXPTIMPT=YES is specified, or used by default, on the IWM4CON macro invocation.

Table 63. Return and Reason Codes for the IWMUIMPT Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx0842	**Equate Symbol**: IwmRsnCodeNoWLMConnect **Meaning**: The caller's primary address space is not connected to WLM. **Action**: Invoke the IWM4CON macro before invoking this macro.
8	xxxx0870	**Equate Symbol**: IwmRsnCodeBadExportToken **Meaning**: The export token is not validly formatted. **Action**: Check for possible storage overlay of the export token.
8	xxxx0871	**Equate Symbol**: IwmRsnCodeDidNotExportOrImport **Meaning**: The primary address space did not import the enclave. **Action**: Invoke IWMUIMPT from the correct address space.
C	—	**Equate Symbol**: IwmRetCodeEnvError **Meaning**: Environmental error.
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error. **Action**: Contact your system programmer.

Example

```
         IWMUIMPT XTOKEN=EXPORTT,CONNTKN=CONNECTT
*
* Storage areas
*
EXPORTT  DS    CL32            Export token
CONNECTT DS    CL4             Connect token
```

IWMWMCON — WLM modify connect

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:

Minimum authorization:	Supervisor state or program key mask (PKM) allowing keys 0-7.
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any P,S. Current Home address space must be the same as Home when the corresponding Connect was invoked.
AMODE:	31-bit
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks may be held.
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, IWMRSNCODE_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 TCB mode (not SRB)
- current TCB must match the TCB 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 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 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:

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

```
►►──┬──────┬──b──IWMWMCON──b──CONNTKN=conntkn──,SUBSYS=subsys──────────────►
    └─name─┘

►─,SUBSYSNM=subsysnm──┬─────────────────┬──┬─────────────────┬─────────────►
                      └─,RETCODE=retcode┘  └─,RSNCODE=rsncode┘

   ┌─,PLISTVER=IMPLIED_VERSION─┐   ┌─,MF=S─────────────────────────────┐
►──┼───────────────────────────┼───┼───────────────────────────────────┼───►◄
   ├─,PLISTVER=MAX─────────────┤   │                    ┌─,0D───┐      │
   └─,PLISTVER=0───────────────┘   ├─,MF=(L─,list addr──┼───────┼─)────┤
                                   │                    └─,attr─┘      │
                                   │                    ┌─,COMPLETE─┐  │
                                   └─,MF=(E─,list addr──┴───────────┴─)┘
```

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***,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. If you specify 15, GPR15, REG15, or R15 (within or without parentheses), the value will be left in GPR 15.

To code: Specify the RS-type address of a fullword field, or register (2)-(12) or (15), (GPR15), (REG15), or (R15).

,RSNCODE=*rsncode*
An optional output parameter into which the reason code is to be copied from GPR 0. If you specify 0, 00, GPR0, GPR00, REG0, REG00, or R0 (within or without parentheses), the value will be left in GPR 0.

To code: Specify the RS-type address of a fullword field, or register (0) or (2)-(12), (00), (GPR0), (GPR00), REG0), (REG00), or (R0).

,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 codes and reason codes

When the IWMWMCON 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 64. Return and Reason Codes for the IWMWMCON Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.
4	xxxx0409	**Equate Symbol**: IwmRsnCodeNoConn **Meaning**: Input connection token does not reflect an active connection to WLM. **Action**: None required.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0802	**Equate Symbol**: IwmRsnCodeXmemUserKeyTkn **Meaning**: Caller is in cross-memory mode while the token was requested in user key. **Action**: Avoid requesting this function while in cross-memory mode.

Table 64. Return and Reason Codes for the IWMWMCON Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: Caller is locked. **Action**: Avoid requesting this function while locked.
8	xxxx0809	**Equate Symbol**: IwmRsnCodeSrbUserKeyTkn **Meaning**: Caller is in SRB mode, while the token was obtained in a user key (8-F). **Action**: Avoid requesting this function in SRB mode for tokens associated with user key.
8	xxxx080A	**Equate Symbol**: IwmRsnCodeTcbNotOwnerUserKeyTkn **Meaning**: Current TCB is not the owner, while the token was obtained in a user key (8-F). **Action**: Avoid requesting this function under a TCB other than the owner for a token associated with user key.
8	xxxx080B	**Equate Symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list. **Action**: Check for possible storage overlay.
8	xxxx0823	**Equate Symbol**: IwmRsnCodeDatoff **Meaning**: Caller invoked service while DATOFF **Action**: Avoid requesting this function in this environment.
8	xxxx0824	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: Caller invoked service but was in 24 bit addressing mode. **Action**: Request this function only when you are in 31 bit addressing mode.
8	xxxx0825	**Equate Symbol**: IwmRsnCodeAscModeNotPrimary **Meaning**: Caller invoked service but was not DAT on Primary ASC mode. **Action**: Avoid requesting this function in this environment.
8	xxxx0826	**Equate Symbol**: IwmRsnCodeTaskTerm **Meaning**: Caller invoked service while task termination is in progress for the TCB associated with the owner. **Action**: Avoid requesting this function in this environment.

Table 64. Return and Reason Codes for the IWMWMCON Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0827	**Equate Symbol**: IwmRsnCodeRsvdNot0 **Meaning**: Reserved field in parameter list was non-zero. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx0828	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: Version number in parameter list is not valid. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx082F	**Equate Symbol**: IwmRsnCodeWrongHome **Meaning**: Caller invoked the service from the wrong home address space. **Action**: Invoke the function with the correct home address space.
8	xxxx0840	**Equate Symbol**: IwmRsnCodeServiceNotEnabled **Meaning**: Caller's space connection is not enabled for this service. The caller is not connected to the WLM work management services. **Action**: 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.
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error. **Action**: No action required. The function may be successful if invoked again.

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
```

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*.

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:

Minimum authorization:	Supervisor state, or problem state. Any PSW key.
Dispatchable unit mode:	Task
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31-bit
ASC mode:	Primary or Access Register (AR) mode.
Interrupt status:	Enabled
Locks:	Unlocked
Control parameters:	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.

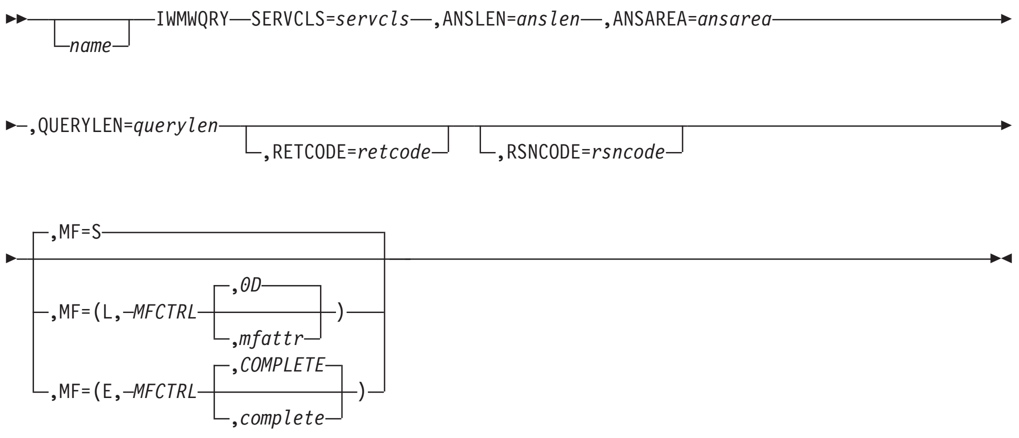
Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

Performance implications

None.

Syntax

The syntax of the IWMWQRY macro is as follows:



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.

,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 codes 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		**Meaning**: Warning.
04	040A	**Meaning**: Warning. The output area supplied is too small to receive all the available information.
04	0410	**Meaning**: Warning. The input service class token does not reflect a service class in the current service policy.
08	0801	**Meaning**: Program error. The caller is in SRB mode.
08	0803	**Meaning**: Program error. The caller is disabled.
08	0804	**Meaning**: Program error. The caller is locked.
08	080D	**Meaning**: Program error. Input service class is not valid.
08	0810	**Meaning**: Program error. The caller has EUT FRR established.
08	0830	**Meaning**: Program error. The 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,ANSLEN=ANSLEN,SERVCLS=SERVCLS
      QUERYLEN=QUERYLEN,RETCODE=RCODE,
      RSNCODE=RSN,MF=(E,MFWQRY)
```

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 transaction ABENDs are occurring.

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:

Minimum authorization:	Supervisor state, or problem state. Any PSW key.
Dispatchable unit mode:	Task
Cross memory mode:	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.
AMODE:	31-bit
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	Unlocked
Control parameters:	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

```
►►──┬──────┬──IWMWQWRK──SEARCH_BY=─┬─CONNTKN─┤ parameters-1 ├─┬──,RETCODE=retcode──────────►
    └─name─┘                       └─LU62TKN─┤ parameters-2 ├─┘
```

```
                    ┌─,MF=S──────────────────────────────┐
►─,RSNCODE=rsncode──┼────────────────────────────────────┼──────────────────►◄
                    │                 ┌─,0D──────┐        │
                    ├─,MF=(L,─MFCTRL──┼──────────┼─)──────┤
                    │                 └─,mfattr──┘        │
                    │                 ┌─,COMPLETE─┐       │
                    └─,MF=(E,─MFCTRL──┼───────────┼─)─────┘
                                      └─,complete─┘
```

parameters-1

```
                       ┌─,LU62_LIST=NO_LU62_LIST──────────────────────────────────┐
►►──,CONNTKN=conntkn───┴─,LU62_LIST=lu62_list───,LU62_LISTSIZE=lu62_listsize──────┴────►

►─,QUERYLEN=querylen─────────────────────────────────────────────────────────────────►◄
```

parameters-2

```
     ┌─,SUBSYS=NO_SUBSYS─┐   ┌─,SUBSYSNM=NO_SUBSYSNM─┐
►►───┴─,SUBSYS=subsys────┴───┴─,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 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 when you specify CONNTKN. When you specify LU62TKN, this parameter is required.. 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 <= ASID_LISTSIZE, AND that STKN_SIZENEED + 8*SKIP_REGIONS <= 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.

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 codes 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
X'00'		**Meaning**: Successful completion.
X'04'	X'0408'	**Meaning**: Warning. No work matching the input search arguments was found.
X'04'	X'0409'	**Meaning**: Warning. Input connection token does not reflect an active connection to WLM.

Hexadecimal Return Code	Hexadecimal Reason Code	Meaning
X'04'	X'040A'	**Meaning**: Warning. The output area supplied is too small to receive all the available information.
X'08'	X'0802'	**Meaning**: Program error. The caller is in cross-memory mode while some monitoring environments were in user key.
X'08'	X'0803'	**Meaning**: Program error. Caller is disabled.
X'08'	X'0804'	**Meaning**: Program error. Caller is locked.
X'08'	X'080B'	**Meaning**: Program error. Error accessing parameter list.
X'08'	X'0810'	**Meaning**: Program error. The caller has EUT FRR established.
X'08'	X'0823'	**Meaning**: Program error. The caller invoked the service while dynamic address translation was disabled.
X'08'	X'0824'	**Meaning**: Program error. The caller invoked the service but was in 24-bit addressing mode.
X'08'	X'0825'	**Meaning**: Program error. The caller invoked the service but was not in primary ASC mode.
X'08'	X'0827'	**Meaning**: Program error. Reserved field in parameter list was non-zero.
X'08'	X'0828'	**Meaning**: Program error. Version number in parameter list is not valid.
X'08'	X'0829'	**Meaning**: 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.

IWMWSYSQ — Query system information

This service queries 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 auxiliary storage, fixed storage, or SQA 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.

Notes:
- 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:

Minimum authorization:	Problem state. Any PSW key
Dispatchable unit mode:	Task
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31-bit
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks may be held.
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, IWMRSNCODE_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

The syntax of the IWMWSYSQ macro is as follows:

```
                                                  ┌─,EXTENDED_DATA=NO──┐
►►──┬──────┬──IWMWSYSQ──SYSINFO_BLOCK=sysinfo_block──┼────────────────────┼──────────────────►
    └─name─┘                                      └─,EXTENDED_DATA=YES─┘

►─,ANSLEN=anslen──,QUERYLEN=querylen──┬──────────────────┬──┬──────────────────┬──────────►
                                      └─,RETCODE=retcode─┘  └─,RSNCODE=rsncode─┘
```

```
   ┌─,PLISTVER=IMPLIED_VERSION─┐  ┌─,MF=S──────────────────────────────────┐
►──┼───────────────────────────┼──┼────────────────────────────────────────┼──────────►◄
   ├─,PLISTVER=MAX─────────────┤  │                    ┌─,0D───┐            │
   └─,PLISTVER=2───────────────┘  ├─,MF=(L─,list addr──┼───────┼─)──────────┤
                                  │                    └─,attr─┘            │
                                  │                    ┌─,COMPLETE─┐        │
                                  └─,MF=(E─,list addr──┴───────────┴─)──────┘
```

Parameters

The parameters are explained as follows:

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.

,EXTENDED_DATA=NO
,EXTENDED_DATA=YES
An optional input parameter, which describes whether the format of the output area named by SYSINFO_BLOCK includes the extended section or not (see IWMWSYSI). 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 SYSI, which consists of an array of entries described by SYSI_ENTRY.

,EXTENDED_DATA=YES
indicates that the format of the output area given by SYSINFO_BLOCK includes first the standard information mapped by the SYSI, which consists of an array of entries described by SYSI_ENTRY, followed immediately by the header for the extension section and another array of entries described by SYSI_EXT_Entry.

The EXTENDED_DATA parameter is only available to callers of the IWMWSYSQ service when invoked on a system running z/OS R9 or higher.

,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 codes 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 65. Return and Reason Codes for the IWMWSYSQ Macro

Return Code	Reason Code	Equate Symbol, Meaning, and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.
4	xxxx040A	**Equate Symbol**: IwmRsnCodeOutputAreaTooSmall **Meaning**: The output area supplied is too small to receive all the available information. **Action**: None required. If necessary, reinvoke the service with an output area of sufficient size (returned in QUERYLEN) to receive all information.

Table 65. Return and Reason Codes for the IWMWSYSQ Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
4	xxxx0420	**Equate Symbol**: IwmRsnCodeSysInfoIncomplete **Meaning**: System capacity data for one or more systems running in goal mode is unavailable when the IWMWSYSQ service is invoked. **Action**: None required. If necessary, wait a few minutes and reinvoke the service to receive all information.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0801	**Equate Symbol**: IwmRsnCodeSrbMode **Meaning**: Caller is in SRB mode. **Action**: Avoid requesting this function while in SRB mode.
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: Caller is locked. **Action**: Avoid requesting this function while locked.
8	xxxx080B	**Equate Symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list. **Action**: Check for possible storage overlay.
8	xxxx0823	**Equate Symbol**: IwmRsnCodeDatoff **Meaning**: The caller invoked the service while DATOFF **Action**: Avoid requesting this function in this environment.
8	xxxx0824	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: The caller invoked the service but was in 24-bit addressing mode. **Action**: Request this function only when you are in 31-bit addressing mode.
8	xxxx0828	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: Version number in parameter list is not valid or the length specified is incorrect. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx083C	**Equate Symbol**: IwmRsnCodeMissingAcro **Meaning**: Required parameter list acronym (eye catcher) not found or a zero SYSINFO_BLOCK pointer is found to be associated with a non-zero ANSLEN. **Action**: Check for possible storage overlay of the parameter list after ensuring that the acronym was correctly set.
C	—	**Equate Symbol**: IwmRetCodeEnvError **Meaning**: Environmental error.

Table 65. Return and Reason Codes for the IWMWSYSQ Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
C	xxxx0C12	**Equate Symbol**: IwmRsnNoGoalModeSystems **Meaning**: There are no goal mode systems in the sysplex **Action**: No action required.
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error. **Action**: No action required. The function may be successful if invoked again.

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
```

IWM4AEDF — WLM define dynamic application environments

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`.

Environment

The requirements for the caller are:

Minimum authorization:	Supervisor state or program key mask (PKM) allowing keys 0-7.
Dispatchable unit mode:	Task or SRB
Cross memory mode:	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).
AMODE:	31- or 64-bit
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks may be held.
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, 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

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 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

```
►►──┬──────┬──b──IWM4AEDF──b──CONNTKN=conntkn──────────────────────────────────────►
    └─name─┘

►──┬─,FUNC=ADD──┤ parameters-1 ├────────────────────────────────────────────┬──────►
   ├─,FUNC=DELETE─,APPLENV=applenv──────────────────────────────────────────┤
   │                               ┌─,DISTRIBUTE_WORK=FIRST_AVAILABLE─┐ ┌─,STATIC=NO──┐ │
   └─,FUNC=MODIFY─,APPLENV=applenv─┴─,DISTRIBUTE_WORK=ROUND_ROBIN─────┘ └─,STATIC=YES─┘─┘

                                              ┌─,PLISTVER=IMPLIED_VERSION─┐
►──┬────────────────────┬──┬────────────────────┬──┼─,PLISTVER=MAX─────────────┼──────────────►
   └─,RETCODE=retcode───┘  └─,RSNCODE=rsncode───┘  └─,PLISTVER=0───────────────┘
   ┌─,MF=S──────────────────────────────────┐
►──┤                                        ├──────────────────────────────────────────────►◄
   │                     ┌─,0D───┐          │
   ├─,MF=(L─,list addr───┤       ├─)────────┤
   │                     └─,attr─┘          │
   │                     ┌─,COMPLETE─┐      │
   └─,MF=(E─,list addr───┴───────────┴─)────┘
```

parameters-1

```
                                                ┌─,JCLPARMS=0──────────┐
►►──,APPLENV=applenv──,JCLPROC=jclproc──────────┤                      ├──────────────────►
                                                └─,JCLPARMS=jclparms───┘

   ┌─,JOBSPACE=NO──┐   ┌─,SINGLE_SERVER=NO──┐   ┌─,SELECT_POLICY=0──────────────┐
►──┤               ├───┤                    ├───┤                               ├─────────►
   └─,JOBSPACE=YES─┘   └─,SINGLE_SERVER=YES─┘   └─,SELECT_POLICY=select_policy──┘

   ┌─,DISTRIBUTE_WORK=FIRST_AVAILABLE─┐
►──┤                                  ├────────────────────────────────────────────────►◄
   └─,DISTRIBUTE_WORK=ROUND_ROBIN─────┘
```

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 dynamic application environment.

To code: Specify the RS-type address, or address in register (2)-(12), of a 32-character field.

,APPLENV=*applenv*
When FUNC=DELETE is specified, a required input parameter, which contains the name of the dynamic application environment.

To code: Specify the RS-type address, or address in register (2)-(12), of a 32-character field.

,APPLENV=*applenv*
When FUNC=MODIFY 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.

,DISTRIBUTE_WORK=FIRST_AVAILABLE
,DISTRIBUTE_WORK=ROUND_ROBIN
When FUNC=MODIFY 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.

,JOBSPACE=NO
,JOBSPACE=YES
When FUNC=ADD is specified, an optional parameter, which specifies whether the server address spaces for the application environment should be started as 'SYSTEM' or 'JOB' address spaces. System address spaces are defined in the RACF STARTED class as jobname.jobname. Job address spaces are defined in the RACF STARTED class as procname.jobname. The default is JOBSPACE=NO.

,JOBSPACE=NO
Server address spaces will be started as system address spaces. This is the default.

,JOBSPACE=YES
Server address spaces will be started as job address spaces.

,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. If you specify 15, GPR15, REG15, or R15 (within or without parentheses), the value will be left in GPR 15.

To code: Specify the RS-type address of a fullword field, or register (2)-(12) or (15), (GPR15), (REG15), or (R15).

,RSNCODE=*rsncode*

An optional output parameter into which the reason code is to be copied from GPR 0. If you specify 0, 00, GPR0, GPR00, REG0, REG00, or R0 (within or without parentheses), the value will be left in GPR 0.

To code: Specify the RS-type address of a fullword field, or register (0) or (2)-(12), (00), (GPR0), (GPR00), REG0), (REG00), or (R0).

,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.
- 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 a 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 codes 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 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 66. Return and Reason Codes for the IWM4AEDF Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.

Table 66. Return and Reason Codes for the IWM4AEDF Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: Caller is locked. **Action**: Avoid requesting this function while locked.
8	xxxx080B	**Equate Symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list. **Action**: Check for possible storage overlay.
8	xxxx0821	**Equate Symbol**: IwmRsnCodeBadConn **Meaning**: Input connect token does not pass validity checking. **Action**: 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	**Equate Symbol**: IwmRsnCodeDatoff **Meaning**: Caller invoked service while DATOFF **Action**: Avoid requesting this function in this environment.
8	xxxx0824	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: Caller invoked service but was in 24 bit addressing mode. **Action**: Request this function only when you are in 31 bit addressing mode.
8	xxxx0825	**Equate Symbol**: IwmRsnCodeAscModeNotPrimary **Meaning**: Caller invoked service but was not in primary ASC mode. **Action**: Avoid requesting this function in this environment.
8	xxxx0827	**Equate Symbol**: IwmRsnCodeRsvdNot0 **Meaning**: Reserved field in parameter list was non-zero. **Action**: Check for use of keywords that are not supported by the MVS release on which the program is running.
8	xxxx0828	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: Version number in parameter list is not valid. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx083F	**Equate Symbol**: IwmRsnCodePrimaryNotOwnConn **Meaning**: Primary address space does not own the passed connect token. **Action**: 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.

Table 66. Return and Reason Codes for the IWM4AEDF Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0840	**Equate Symbol**: IwmRsnCodeServiceNotEnabled **Meaning**: Caller's space connection is not enabled for this service **Action**: Make sure that Queue_Manager=Yes, or Server_Manager=Yes with Server_Type=Queue is specified on the IWM4CON request to enable this service.
8	xxxx0842	**Equate Symbol**: IwmRsnCodeNoWLMConnect **Meaning**: Caller's space disconnected from WLM during processing of the request. **Action**: None.
8	xxxx0890	**Equate Symbol**: IwmRsnCodeApplEnvExists **Meaning**: The caller tried to add an application environment that has already been defined. subsystem type. **Action**: Check whether the correct application environment name is being used. Make sure that a unique application environment name is used when adding application environments.
8	xxxx0891	**Equate Symbol**: IwmRsnCodeApplEnvNotFound **Meaning**: The caller tried to delete or modify an application environment that does not exist. **Action**: Check whether the correct application environment name is being used.
C	—	**Equate Symbol**: IwmRetCodeEnvError **Meaning**: Environmental error.
C	xxxx0C01	**Equate Symbol**: IwmRsnCodeNoStg **Meaning**: Storage is not available for the request. **Action**: Contact your system programmer. There is a common storage shortage.
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error. **Action**: Contact your system programmer.

Example

To add a dynamic application environment:

```
        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
```

IWM4CLSY — Classify work

The purpose of the IWM4CLSY 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.

Note: This service was previously called IWMCLSFY for 31-bit addressing only (see "IWMCLSFY — Classify work request" on page 837).

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 IWM4CON.
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31- or 64-bit. If in 64-bit addressing mode, code SYSSTATE AMODE64=YES before invoking this macro.
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	Unlocked or locked.
Control parameters:	Control parameters must be in the primary address space.

Programming requirements

1. The macro IWMYCON must be included to use this macro.
2. The 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 z/OS V2R1, or higher.
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 the current primary must equal the primary at the time that IWM4CON was invoked.
5. Only limited checking is done of the connect token obtained from IWM4CON.

6. Parameters SOURCELU, CLIENTIPADDR and NETID/LUNAME are mutually exclusive.
7. Parameters EWLM_OUTCORR, EWLM_CHCORR and EWLM_CHCTKN are all mutually exclusive.
8. There is no restriction on the length of data passed as a parameter value, but all storage between the start and end must be allocated (getmained).

Input register information

Before issuing the IWM4CLSY macro, the caller must ensure that the following general purpose registers (GPRs) contain the specified information:

Register	**Contents**
13	The address of a 216-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

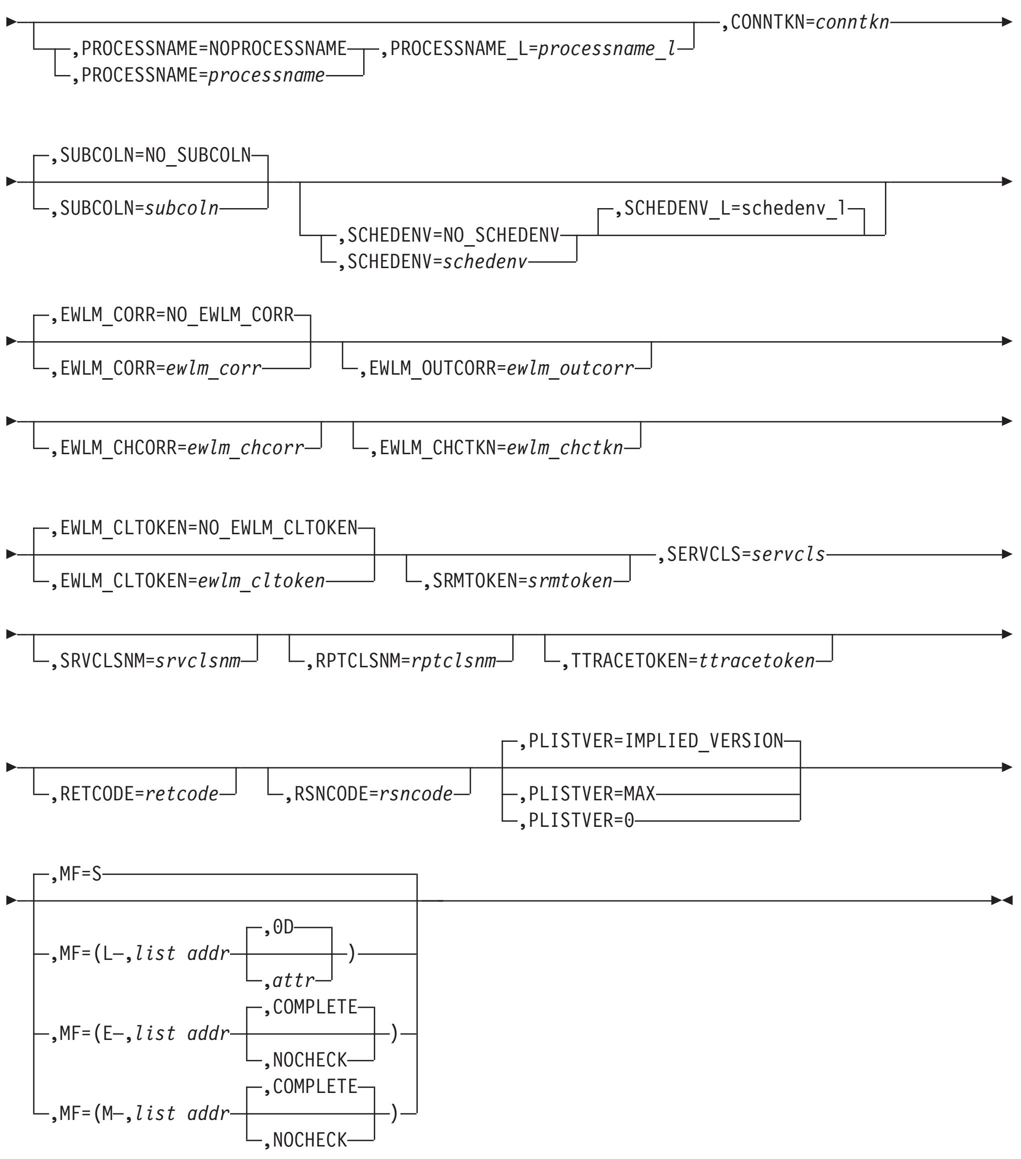
None.

Syntax

The syntax of the IWM4CLSY macro is as follows:

```
                        ┌─TRXNAME=NO_TRXNAME─┐ ┌─,USERID=NO_USERID─┐
►►──┬──────┬──IWM4CLSY──┼────────────────────┼─┼───────────────────┼──────────────────►
    └─name─┘            └─TRXNAME=trxname────┘ └─,USERID=userid────┘
```

```
   ┌─,TRXCLASS=NO_TRXCLASS──┐
►──┼────────────────────────┼──┬──────────────────────────────────────────────────────┬──►
   └─,TRXCLASS=trxclass─────┘  └─┬─,ACCTINFO=NO_ACCTINFO─┬─,ACCTINFO_L=acctinfo_l─┘
                                 └─,ACCTINFO=acctinfo────┘

   ┌─,SOURCELU=NO_SOURCELU───┐  ┌─,NETID=NO_NETID─┐  ┌─,LUNAME=NO_LUNAME─┐
►──┼─────────────────────────┼──┼─────────────────┼──┼───────────────────┼──────────►
   ├─,SOURCELU=sourcelu──────┤  └─,NETID=netid────┘  └─,LUNAME=luname────┘
   └─,SOURCELU_L=sourcelu_l──┘

►──┬──────────────────────────────────────────────────────┬─────────────────────────►
   └─┬─,SUBSYSPM=NO_SUBSYSPM─┬─,SUBSYSPM_L=subsyspm_l─────┘
     └─,SUBSYSPM=subsyspm────┘

►──┬────────────────────────────────────────────────────────────┬───────────────────►
   └─┬─,CLIENTACCT=NO_CLIENTACCT─┬─,CLIENTACCT_L=clientacct_l──┘
     └─,CLIENTACCT=clientacct────┘

   ┌─,CLIENTIPADDR=NO_CLIENTIPADDR─┐
►──┼───────────────────────────────┼────────────────────────────────────────────────►
   └─,CLIENTIPADDR=clientipaddr────┘

►──┬──────────────────────────────────────────────────────────────────┬─────────────►
   └─┬─,CLIENTTRXNM=NO_CLIENTTRXNM─┬─,CLIENTTRXNM_L=clienttrxnm_l─────┘
     └─,CLIENTTRXNM=clienttrxnm────┘

►──┬───────────────────────────────────────────────────────────────────────┬────────►
   └─┬─,CLIENTUSERID=NO_CLIENTUSERID─┬─,CLIENTUSERID_L=clientuserid_l──────┘
     └─,CLIENTUSERID=clientuserid────┘

►──┬───────────────────────────────────────────────────────────────────────┬────────►
   └─┬─,CLIENTWKSTNM=NO_CLIENTWKSTNM─┬─,CLIENTWKSTNM_L=clientwkstnm_l──────┘
     └─,CLIENTWKSTNM=clientwkstnm────┘

                                                             ┌─,PLAN=NO_PLAN─┐
►──┬──────────────────────────────────────────────────────┬──┼───────────────┼─────►
   └─┬─,COLLECTION=NO_COLLECTION─┬─,COLLECTION_L=collection_l─┘  └─,PLAN=plan────┘
     └─,COLLECTION=collection────┘

   ┌─,PACKAGE=NO_PACKAGE───┐  ┌─,CONNECTION=NO_CONNECTION─┐
►──┼───────────────────────┼──┼───────────────────────────┼─────────────────────────►
   ├─,PACKAGE=package──────┤  └─,CONNECTION=connection────┘
   └─,PACKAGE_L=package_l──┘

►──┬────────────────────────────────────────────────────────────────┬───────────────►
   └─┬─,CORRELATION=NO_CORRELATION─┬─,CORRELATION_L=correlation_l───┘
     └─,CORRELATION=correlation────┘

   ┌─,PERFORM=NO_PERFORM─┐
►──┼─────────────────────┼──┬────────────────────────────────────────────┬──────────►
   └─,PERFORM=perform────┘  └─┬─,PRCNAME=NO_PRCNAME─┬─,PRCNAME_L=prcname_l─┘
                              └─,PRCNAME=prcname────┘

   ┌─,PRIORITY=NO_PRIORITY─┐
►──┼───────────────────────┼────────────────────────────────────────────────────────►
   └─,PRIORITY=priority────┘
```



Parameters

The parameters are explained as follows:

name
: An optional symbol, starting in column 1, that is the name on the IWM4CLSY macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

,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.

,ACCTINFO_L=*acctinfo_l*
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.

,CLIENTACCT=*clientacct*
,CLIENTACCT=NO_CLIENTACCT
An optional input parameter, which contains the accounting information. For environments where accounting information is available on some, but not all flows, providing a data area initialized to all blanks is equivalent to specifying NO_CLIENTACCT. The default is NO_CLIENTACCT, 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.

,CLIENTACCT_L=*clientacct_l*
When CLIENTACCT=*clientacct* is specified, a required input parameter, which contains the length of the client accounting information field. The maximum value supported is 512.

To code: Specify the RS-type address, or address in register (2)-(12), of a fullword field, or specify a literal decimal value.

,CLIENTIPADDR=*clientipaddr*
,CLIENTIPADDR=NO_CLIENTIPADDR
An optional input parameter, which contains the source client IPv6 address, which is left-justified and represented as a colon hexadecimal address. An example of an IPv6 address is: X'2001:0DB8:0000:0000:0008:0800:200C:417A'

The compressed format is not supported for classifications. For environments where the source client IPV6 address may be available on some, but not all flows, providing a data area initialized to all blanks is equivalent to specifying NO_CLIENTIPADDR .

SOURCELU, CLIENTIPADDR and NETID/LUNAME are mutually exclusive. The default is NO_CLIENTIPADDR, which indicates that no CLIENTIPADDR name is passed.

To code: Specify the RS-type address, or address in register (2)-(12), of a 39-character field.

,CLIENTTRXNM=*clienttrxnm*
,CLIENTTRXNM=NO_CLIENTTRXNM
An optional input parameter, which contains the client 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, providing a data area initialized to all blanks is equivalent to specifying NO_CLIENTTRXNM. The default is NO_CLIENTTRXNM, which indicates that no client transaction name is passed.

To code: Specify the RS-type address, or address in register (2)-(12), of a character field.

,CLIENTTRXNM_L=*clienttrxnm_l*
When CLIENTTRXNM=*clienttrxnm* is specified, a required input parameter, which contains the length of the client transaction name field. The maximum value supported is 255.

To code: Specify the RS-type address, or address in register (2)-(12), of a fullword field, or specify a literal decimal value.

,CLIENTUSERID=*clientuserid*
,CLIENTUSERID=NO_CLIENTUSERID
An optional input parameter, which contains the value of the client user ID which may be different than USERID and provided from the client information that is specified for the connection. For environments where the client user ID name may be available on some, but not all flows, providing a data area initialized to all blanks is equivalent to specifying NO_CLIENTUSERID. The default is NO_CLIENTUSERID, which indicates that no CLIENTUSERID name is passed.

To code: Specify the RS-type address, or address in register (2)-(12), of a character field.

,CLIENTUSERID_L=*clientuserid_l*
When CLIENTUSERID=*clientuserid* is specified, a required input parameter, which contains the length of the client user ID name field. The maximum value supported is 128.

To code: Specify the RS-type address, or address in register (2)-(12), of a fullword field, or specify a literal decimal value.

,CLIENTWKSTNM=*clientwkstnm*
,CLIENTWKSTNM=NO_CLIENTWKSTNM
An optional input parameter, which contains the value of the client workstation name or host name from the client information that is specified for the connection. For environments where the client workstation name may be available on some, but not all flows, providing a data area initialized to all blanks is equivalent to specifying NO_CLIENTWKSTNM. The default is NO_CLIENTWKSTNM, which indicates that no CLIENTWKSTNM name is passed.

To code: Specify the RS-type address, or address in register (2)-(12), of a character field.

,CLIENTWKSTNM_L=*clientwkstnm_l*
When CLIENTWKSTNM=*clientwkstnm* is specified, a required input parameter, which contains the length of the client workstation name field. The maximum value supported is 255.

To code: Specify the RS-type address, or address in register (2)-(12), of a fullword field, or specify a literal decimal value.

,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, providing a data area initialized to all blanks is equivalent to specifying NO_COLLECTION. The default is NO_COLLECTION, which 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_L=*collection_l*
When COLLECTION=*collection* is specified, a required input parameter, which contains the length of the collection name. The maximum value supported is 18.

To code: Specify the RS-type address, or address in register (2)-(12), of a fullword field, or a literal decimal value.

,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, providing a data area initialized to all blanks is equivalent to specifying NO_CONNECTION . The default is NO_CONNECTION, which 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.

,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, providing a data area initialized to all blanks is equivalent to specifying NO_CORRELATION. The default is NO_CORRELATION, which indicates that no CORRELATION name is passed.

To code: Specify the RS-type address, or address in register (2)-(12), of a character field.

,CORRELATION_L=*correlation_l*
When CORRELATION=*correlation* is specified, a required input parameter, which contains the length of the correlation identifier. The maximum value supported is 12.

To code: Specify the RS-type address, or address in register (2)-(12), of a fullword 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.

Note:
1. Currently z/OS V2R1 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 mutually exclusive.

To code: Specify the RS-type address, or address in register (2)-(12), of a 64-character field.

,EWLM_CLTOKEN=*ewlm_cltoken*
,EWLM_CLTOKEN=NO_EWLM_CLTOKEN

An optional input parameter, which contains internal EWLM classification information to be passed from EWLM to WLM. This parameter is used internally by WLM and must not be used by application programs. The default is NO_EWLM_CLTOKEN, which indicates that no EWLM classification information was passed.

To code: Specify the RS-type address, or address in register (2)-(12), of a 40-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.

Note:

1. The architected length field of an ARM correlator in the first two bytes must contain a value between 4 (X'0004') and 512 (X'0200').
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 IWM4CLSY 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.
- 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.

Note:

1. Specifying EWLM_OUTCORR (unlike EWLM_CHCORR or EWLM_CHCTKN) does not indicate the beginning of a sub work request.

2. Currently z/OS V2R1 only uses the first 64 bytes of the EWLM_OUTCORR field.
3. 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.
4. 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, providing a data area initialized to all blanks is equivalent to specifying NO_LUNAME.

SOURCELU, CLIENTIPADDR and LUNAME are mutually exclusive. The default is NO_LUNAME, which 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 IWM4CLSY in the following order:

- Use IWM4CLSY ...MF=(M,list-addr,COMPLETE) specifying appropriate parameters, including all required ones.
- Use IWM4CLSY ...MF=(M,list-addr,NOCHECK), specifying the parameters that you want to change.
- Use IWM4CLSY ...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, providing a data area initialized to all blanks is equivalent to specifying NO_NETID.

SOURCELU, CLIENTIPADDR and NETID are mutually exclusive with NETID. The default is NO_NETID, which 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, for example, 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, providing a data area initialized to all blanks is equivalent to specifying NO_PACKAGE The default is NO_PACKAGE, which 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.

,PACKAGE_L=*package_l*
When PACKAGE=*package* is specified, a required input parameter, which contains the length of the package information field. The maximum value supported is 128.

To code: Specify the RS-type address, or address in register (2)-(12), of a fullword field, or specify a literal decimal value.

,PERFORM=*perform*

,PERFORM=<u>NO_PERFORM</u>

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, providing a data area initialized to all blanks is equivalent to specifying NO_PERFORM. The default is NO_PERFORM, which 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=<u>NO_PLAN</u>

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, providing a data area initialized to all blanks is equivalent to specifying NO_PLAN. The default is NO_PLAN, which 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=<u>IMPLIED_VERSION</u>

,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

,PRCNAME=*prcname*

,PRCNAME=<u>NO_PRCNAME</u>

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, providing a data area initialized to all blanks is equivalent to specifying 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_L=*prcname_l*

When PRCNAME=*prcname* is specified, a required input parameter, which contains the length of the procedure name. The maximum value supported is 128.

To code: Specify the RS-type address, or address in register (2)-(12), of a fullword field, or specify a literal decimal value.

,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, providing a data area initialized to X'80000000' (the largest negative integer) is equivalent to specifying NO_PRIORITY. The default is NO_PRIORITY, which indicates that no PRIORITY value is passed.

To code: Specify the RS-type address, or address in register (2)-(12), of a fullword field, or specify a literal decimal value.

,PROCESSNAME=*processname*
,PROCESSNAME=NOPROCESSNAME

An optional input parameter, which contains the process name associated with the work request. The default is NOPROCESSNAME, which 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_L=*processnm_l*

When PROCESSNAME=*processname* is specified, a required input parameter, which contains the length of the process name. The maximum value supported is 32.

To code: Specify the RS-type address, or address in register (2)-(12), of a fullword field, or specify a literal decimal value.

,RETCODE=*retcode*

An optional output parameter into which the return code is to be copied from GPR 15. If you specify 15, GPR15, REG15, or R15 (with or without parentheses), the value will be left in GPR 15.

To code: Specify the RS-type address of a fullword field, or register (2)-(12), or (15), (GPR15), (REG15), or (R15).

,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. If you specify 0, 00, GPR0, GPR00, REG0, REG00, or R0 (with or without parentheses), the value will be left in GPR 0.

To code: Specify the RS-type address of a fullword field, or register (0) or (2)-(12), (00), (GPR0), (GPR00), REG0), (REG00), or (R0).

,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, which indicates that no scheduling environment value is passed.

To code: Specify the RS-type address, or address in register (2)-(12), of a character field.

,SCHEDENV_L=*schedenv_l*
When SCHEDENV=*schedenv* is specified, an optional input parameter, which contains the length of the scheduling environment. The maximum value supported is 16.

To code: Specify the RS-type address, or address in register (2)-(12), of a fullword field, or specify a literal decimal value.

,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, 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.

For environments where the LU name may be available on some, but not all flows, providing a data area initialized to all blanks is equivalent to specifying NO_SOURCELU.

SOURCELU, CLIENTIPADDR and NETID/LUNAME are mutually exclusive. The default is NO_SOURCELU, which 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.

,SOURCELU_L=*sourcelu_l*
When SOURCELU=*sourcelu* is specified, a required input parameter, which contains the length of the source LU name field. The maximum value supported is 17.

To code: Specify the RS-type address, or address in register (2)-(12), of a fullword field, or specify a literal decimal value.

,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.

,SUBCOLN=*subcoln*

,SUBCOLN=NO_SUBCOLN
An optional input parameter, which contains the subsystem collection name associated with the work request. The default is NO_SUBCOLN, which indicates 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, providing a data area initialized to all blanks is equivalent to specifying NO_SUBSYSPM. The default is NO_SUBSYSPM, which 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, providing a data area initialized to all blanks is equivalent to specifying NO_TRXCLASS. The default is NO_TRXCLASS, which 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, providing a data area initialized to all blanks is equivalent to specifying NO_TRXNAME. The default is NO_TRXNAME, which 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 user ID associated with the work request. For environments where the user ID is available on some, but not all flows, providing a data area initialized to all blanks is equivalent to specifying NO_USERID. The default is NO_USERID, which indicates that no user ID is passed.

To code: Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

ABEND codes

None.

Return codes and reason codes

When the IWM4CLSY 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 IWM4CLSY Macro

Return Code	Reason Code	Equate Symbol, Meaning, and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.
4	xxxx0409	**Equate Symbol**: IwmRsnCodeNoConn **Meaning**: Connect token does not reflect a successful Connect. **Action**: None required.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0821	**Equate Symbol**: IwmRsnCodeBadConn **Meaning**: Input connect token does not pass validity checking. **Action**: Check for possible storage overlay.
8	xxxx0840	**Equate Symbol**: IwmRsnCodeServiceNotEnabled **Meaning**: The caller's space connection is not enabled for this service. **Action**: Avoid requesting this function under the input connection. IWM4CON options must be specified previously to enable this service.
8	xxxx0894	**Equate Symbol**: IwmRsnCodeInvalidEWLMCorr **Meaning**: 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 (X'0004') or greater than 512 (X'0200'). **Action**: Check the specification of the EWLM correlator in the classification information.

Table 67. Return and Reason Codes for the IWM4CLSY Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx0895	**Equate Symbol**: IwmRsnCodeEWLMServNotEnabled **Meaning**: Service is not enabled because the caller invoked the IWMCONN service with EWLM=NO. **Action**: Specify the parameter EWLM_CORR only when connected with EWLM=YES.
C	—	**Equate Symbol**: IwmRetCodeEnvError **Meaning**: Environmental error.
C	xxxx0C01	**Equate Symbol**: IwmRsnCodeNoStg **Meaning**: Storage is not available for the request. **Action**: There is a storage shortage. The function may work successfully at a later time.
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error. **Action**: No action required. The function may be successful when invoked again.

Examples

Suppose the transactions processed by a subsystem work manager have the following qualifiers:
- User ID
- Transaction name
- Transaction class

To get the service class associated with an incoming work request, specify:

```
IWM4CLSY USERID=AUSERID,TRXCLASS=ATRXCLS,TRXNAME=ATRXNM,
         CONNTKN=(R7),SERVCLS=(R9),
         RETCODE=RETCODE,RSNCODE=RSNCODE
```

Where the following are declared:

```
AUSERID  DS  CL8
ATRXCLS  DS  CL8
ATRXNM   DS  CL8
```

IWM4CON — Connect 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.

Notes:

- 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: This service was previously called IWMCONN for 31-bit addressing only (see "IWMCONN — Connect to workload management" on page 853).

Environment

The requirements for the caller are:

Minimum authorization:	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.
Dispatchable unit mode:	Task
Cross memory mode:	Non-XMEM when input key is a user key or `SERVER_MANAGER = YES`, otherwise XMEM, any P,S,H.
AMODE:	31- or 64-bit. If in 64-bit addressing mode, code `SYSSTATE AMODE64=YES` before invoking this macro.
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	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 not be used during task/address space termination.
2. Only a single connection is allowed to be active for a given address space at any given time.
3. Specification of both `Queue_Manager=Yes` and `Server_Manager=Yes` requires `Server_Type=Queue`; specification of `Server_Type=Routing` is rejected.
4. Specification of both `Router=Yes` and `Server_Manager=Yes` requires `Server_Type=Routing`; specification of `Server_Type=Queue` is rejected.
5. If the caller's 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.
6. 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)
7. 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 IWM4CLSY, IWMRPT, or IWMMNTFY services, when the current primary matches primary at the time IWM4CON is invoked.
8. This macro supports multiple versions. Some keywords are unique to certain versions. See the **`PLISTVER`** parameter description.

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:

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.

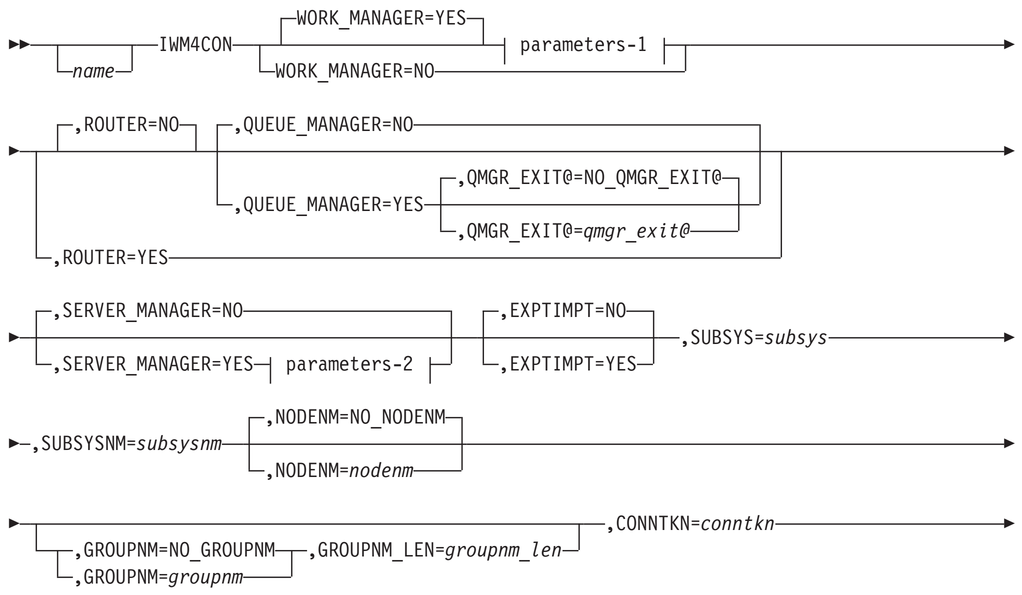
Performance implications

None.

Syntax

The syntax of the IWM4CON macro is as follows:

main diagram



```
                                                          ┌─,PLISTVER=IMPLIED_VERSION─┐
►─┬──────────────────────┬─┬──────────────────────┬──┼───────────────────────────┼──────────────────────►
  └─,RETCODE=retcode─────┘ └─,RSNCODE=rsncode─────┘  ├─,PLISTVER=MAX─────────────┤
                                                          ├─,PLISTVER=0───────────────┤
                                                          └─,PLISTVER=1───────────────┘

  ┌─,MF=S────────────────────────────────┐
►─┼──────────────────────────────────────┼───────────────────────────────────────────────────────────►◄
  │                    ┌─,0D───┐         │
  ├─,MF=(L─,list addr──┼───────┼──)──────┤
  │                    └─,attr─┘         │
  │                    ┌─,COMPLETE─┐     │
  └─,MF=(E─,list addr──┴───────────┴─)───┘
```

parameters-1

```
                                                                            ┌─,EWLM=NO──┐
►►─┬──────────────────────────────────────────────────────────┬──┼───────────┼──────────────────────►
   └─┬─,TOPOLOGY=NO_TOPOLOGY─┬─,NUMBERASCB=numberascb─────┘  └─,EWLM=YES─┘
     └─,TOPOLOGY=topology────┘

►─┬─,CONNTKNKEYP=VALUE─,CONNTKNKEY=conntknkey─┬────────────────────────────────────────────────────►◄
  └─,CONNTKNKEYP=PSWKEY──────────────────────┘
```

parameters-2

```
                          ┌─,DYNAMIC=NO──┐
►►──,APPLENV=applenv──┼──────────────┼──,PARALLEL_EU=parallel_eu───────────────────────────────────►
                          └─,DYNAMIC=YES─┘

►─┬─────────────────────────────────┬──────────────────────────────────────────────────────────────►
  └─,REGION_TOKEN=region_token──────┘

    ┌─,SERVER_TYPE=QUEUE─┐  ┌─,MANAGE_TASKS=NO────────────────────────────────────┐
►─┬─┴────────────────────┴─┼─────────────────────────────────────────────────────┼─┬──────────────►◄
  │                         │                      ┌─,SERVER_LIMIT=1000──────────┐ │ │
  │                         └─,MANAGE_TASKS=YES────┴─,SERVER_LIMIT=server_limit──┴─┘ │
  └─,SERVER_TYPE=ROUTING─,SERVER_DATA=server_data─,SRV_MGR_EXIT@=srv_mgr_exit@──────┘
```

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. It 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 (IWM4CLSY), Report (IWMRPT), Notify (IWMMNTFY). 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.

During the Connect with IWM4CON WLM creates a control block to hold the information. The CONNTKNKEY is the storage key, which is used for the allocation of the control block. The WLM services Classify (IWM4CLSY), Report (IWMRPT) and Notify (IWMMNTFY) get invoked via branch. These WLM services have to run in the same key otherwise these services cannot update the control block allocated during Connect (IWM4CON).

Note however that there are other services that use the connect token, for which the CONNTKNKEY does not relate to PSW update authority but must be a system key (0-7) rather than a user key (8-15).

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 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. This is the default.

,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 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.

,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.

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 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 (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, that is the number of concurrent Begin environments. It is important that this count should 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

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 the following parameter and those from version 0:

  REGION_TOKEN

To code: Specify one of the following:

- IMPLIED_VERSION
- MAX
- A decimal value of 0, or 1

,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.

,REGION_TOKEN=*region_token*
When SERVER_MANAGER=YES is specified, an optional 16-character 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 the work request all needing access to some 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, 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).

,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. 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.

,SERVER_MANAGER=NO
The connecting address space will not use any of the various server-related WLM services documented under SERVER_TYPE. This is the default.

,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.

,SERVER_TYPE=QUEUE

This is the default. 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 the 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, that is 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, that is all servers associated with the Router have identical subsystem names.

If a product choses 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, that is 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 (IWM4CLSY)
- 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.

WORK_MANAGER=YES
The connecting address space will be using the WLM Work Management services. This the default.

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 codes 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 68. Return and Reason Codes for the IWM4CON Macro

Return Code	Reason Code	Equate Symbol, Meaning, and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.

Table 68. Return and Reason Codes for the IWM4CON Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx0801	**Equate Symbol**: IwmRsnCodeSrbMode **Meaning**: The caller is in SRB mode. **Action**: Avoid requesting this function while in SRB mode.
8	xxxx0802	**Equate Symbol**: IwmRsnCodeXmemUserKeyTkn **Meaning**: The caller is in cross-memory mode while the token was requested in user key. **Action**: Avoid requesting this function while in cross-memory mode.
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: The caller is locked. **Action**: Avoid requesting this function while locked.
8	xxxx080B	**Equate Symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list. **Action**: Check for possible storage overlay.
8	xxxx0810	**Equate Symbol**: IwmRsnCodeEutFrr **Meaning**: The caller has EUT FRR established. **Action**: Avoid requesting this function with an EUT FRR set.
8	xxxx0812	**Equate Symbol**: IwmRsnCodeBadAscb **Meaning**: Bad ASCB address passed. **Action**: Check for possible storage overlay.
8	xxxx0823	**Equate Symbol**: IwmRsnCodeDatoff **Meaning**: The caller invoked the service while DATOFF **Action**: Avoid requesting this function in this environment.
8	xxxx0824	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: The caller invoked the service but was in 24-bit addressing mode. **Action**: Request this function only when you are in 31-bit addressing mode.
8	xxxx0825	**Equate Symbol**: IwmRsnCodeAscModeNotPrimary **Meaning**: The caller invoked the service but was not DAT on Primary ASC mode. **Action**: Avoid requesting this function in this environment.

Table 68. Return and Reason Codes for the IWM4CON Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx0826	**Equate Symbol**: IwmRsnCodeTaskTerm **Meaning**: The caller invoked the service while task termination is in progress for the TCB associated with the owner. **Action**: Avoid requesting this function in this environment.
8	xxxx0827	**Equate Symbol**: IwmRsnCodeRsvdNot0 **Meaning**: Reserved field in parameter list was non-zero. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx0828	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: Version number in parameter list is not valid. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx0829	**Equate Symbol**: IwmRsnCodeBadOptions **Meaning**: Parameter list omits required parameters or supplies mutually exclusive parameters or provides data associated with options not selected. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx082C	**Equate Symbol**: IwmRsnCodeBadNumberAscb **Meaning**: NUMBERASCB variable is not a positive value. **Action**: Check for possible storage overlay of the parameter list or variable.
8	xxxx082E	**Equate Symbol**: IwmRsnCodeConnectExists **Meaning**: Connect has already been established for the current home address space. **Action**: Avoid requesting this function when a connection already exists.
8	xxxx0840	**Equate Symbol**: IwmRsnCodeServiceNotEnabled **Meaning**: 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. **Action**: Run the program as a started task.
8	xxxx0841	**Equate Symbol**: IwmRsnCodeXMemMode **Meaning**: The caller is in cross-memory mode. **Action**: Invoke the function in non-cross memory mode.
8	xxxx0847	**Equate Symbol**: IwmRsnCodeOtherSpaceConnected **Meaning**: 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. **Action**: Avoid requesting this function with duplicate values.

Table 68. Return and Reason Codes for the IWM4CON Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx0849	**Equate Symbol**: IwmRsnCodeWLMServBadAPPL **Meaning**: The application environment name (APPLENV=) specified is not the same as the one used by WLM to start the server. **Action**: 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.
8	xxxx084A	**Equate Symbol**: IwmRsnCodeWLMServBadSSN **Meaning**: The subsystem name (SUBSYSNM=) specified is not the same as the one used by WLM to start the server. **Action**: 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.
8	xxxx084B	**Equate Symbol**: IwmRsnCodeWLMServBadSST **Meaning**: The subsystem type (SUBSYS=) specified is not the same as the one used by WLM to start the server. **Action**: 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.
8	xxxx084D	**Equate Symbol**: IwmRsnCodeNotAuthConnect **Meaning**: The caller must be supervisor state or have PSW key mask 0-7 authority to connect to the requested WLM services. **Action**: Avoid requesting this function in this environment.
8	xxxx084E	**Equate Symbol**: IwmRsnCodeWlmServBadType **Meaning**: For WLM started servers, the SERVER_TYPE= is not the one used to start the server. **Action**: Specify the correct SERVER_TYPE.
8	xxxx0853	**Equate Symbol**: IwmRsnCodeWlmQmBadType **Meaning**: There is a queue manager or router environment of the specified subsystem name, but of a different type than that specified by the caller. **Action**: 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.
8	xxxx0855	**Equate Symbol**: IwmRsnCodeBadNumEUMax **Meaning**: PARALLEL_EU variable is greater than the maximum of 65534. **Action**: Specify a value between 1 and 65534.
8	xxxx0856	**Equate Symbol**: IwmRsnCodeBadNumEUMin **Meaning**: PARALLEL_EU is less than the minimum of 1. **Action**: Specify a value between 1 and 65534.

Table 68. Return and Reason Codes for the IWM4CON Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx085C	**Equate Symbol**: IwmRsnCodeWrongNumEU **Meaning**: The caller invoked the service with a PARALLEL_EU value which is different from the PARALLEL_EU of existing servers in the application environment **Action**: Ensure that all servers in the application environment specify the same PARALLEL_EU value. **Note:** If this reason code occurs after changes have been made to the application environment, refer to the section about "Making changes to the Application Environment Servers" in Chapter 13 "Defining Application Environments" in *z/OS MVS Planning: Workload Management*.
8	xxxx0873	**Equate Symbol**: IwmRsnCodeWrongSrvLmt **Meaning**: The caller invoked the service with a SERVER_LIMIT parameter setting which is different from the SERVER_LIMIT of existing servers in the application environment. **Action**: Ensure that all servers in the application environment specify the same SERVER_LIMIT value.
8	xxxx0874	**Equate Symbol**: IwmRsnCodeWrongMngTsk **Meaning**: The caller invoked the service with a MANAGE_TASKS parameter setting which is different from the MANAGE_TASKS of existing servers in the application environment **Action**: Ensure that all servers in the application environment specify the same MANAGE_TASKS value.
8	xxxx0878	**Equate Symbol**: IwmRsnCodeBadNumLimitMax **Meaning**: The caller invoked the service with a SERVER_LIMIT parameter setting which exceeds the maximum number of tasks which can be started in a server address space. The current maximum value is 65534. **Action**: Correct the number or do not specify SERVER_LIMIT parameter in order to use the default.
8	xxxx0879	**Equate Symbol**: IwmRsnCodeBadNumLimitMin **Meaning**: The caller invoked the service with a SERVER_LIMIT parameter setting which is less than what has been defined on the PARALLEL_EU parameter. **Action**: Ensure that SERVER_LIMIT is always greater or equal to PARALLEL_EU.
8	xxxx087A	**Equate Symbol**: IwmRsnCodeNoQServer **Meaning**: The MANAGE_TASKS parameter is not allowed when QUEUE_SERVER=YES has been specified. **Action**: Ensure to use the parameters correctly.
8	xxxx088E	**Equate Symbol**: IwmRsnCodeWlmServBadSSND **Meaning**: For WLM started servers, the NODENM= is not the one used to start the server. **Action**: Specify the correct NODENM.

Table 68. Return and Reason Codes for the IWM4CON Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx088F	**Equate Symbol**: IwmRsnCodeApplNotSSN **Meaning**: The application environment name is defined for use by a different subsystem node. **Action**: Check whether the correct application environment name is being used.
C	—	**Equate Symbol**: IwmRetCodeEnvError **Meaning**: Environmental error.
C	xxxx0C01	**Equate Symbol**: IwmRsnCodeNoStg **Meaning**: Storage is not available for the request. **Action**: There is a storage shortage. The function may work successfully at a later time.
C	xxxx0C09	**Equate Symbol**: IwmRsnCodeNoResmgr **Meaning**: Resource manager could not be established. **Action**: No action required. This condition may be due to a storage shortage condition.
C	xxxx0C14	**Equate Symbol**: IwmRsnCodeNoWorkShutDown **Meaning**: No work selected. Caller is to shutdown. **Action**: The server should shut down (terminate).
C	xxxx0C19	**Equate Symbol**: IwmRsnCodeNotSecAuthConnect **Meaning**: The caller is not authorized by SAF to connect to WLM with SERVER_MANAGER=YES. **Action**: The security administrator must grant access to the appropriate resource.
C	xxxx0C1A	**Equate Symbol**: IwmRsnCodeApplNotDefined **Meaning**: The application environment name is not defined in the active WLM policy. **Action**: 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.
C	xxxx0C1B	**Equate Symbol**: IwmRsnCodeApplNotSST **Meaning**: The application environment name is defined for use by a different subsystem type in the active WLM policy. **Action**: 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.
C	xxxx0C1F	**Equate Symbol**: IwmRsnCodeServerExists **Meaning**: A server exists for the specified application environment which only allows 1 such server in the sysplex. **Action**: 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.

Table 68. Return and Reason Codes for the IWM4CON Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
C	xxxx0C22	**Equate Symbol**: IwmRsnCodeApplEnvQuiesced **Meaning**: The specified application environment has been quiesced, server cannot be started for the request. **Action**: Restart the application environment and then retry the request.
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error. **Action**: No action required. The function may be successful if invoked again.

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
```

IWM4DIS — Disconnect 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 IWM4CLSY and IWM4RPT. 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: This service was previously called IWMDISC for 31-bit addressing only (see "IWMDISC — Disconnect from workload management" on page 872).

Environment

The requirements for the caller are:

Minimum authorization:	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.
Dispatchable unit mode:	Task or SRB When the corresponding connect (IWM4CON) invocation specified `SERVER_MANAGER=YES`, task mode.
Cross memory mode:	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.
AMODE:	31- or 64-bit. If in 64-bit addressing mode, code `SYSSTATE AMODE64=YES` before invoking this macro.
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	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, IWMRSNCODE_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:
   - The 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

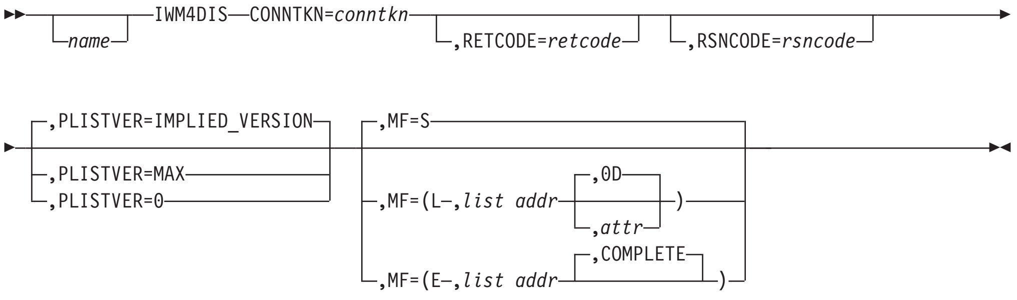
Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

Performance implications

None.

Syntax

The syntax of the IWM4DIS macro is as follows:



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 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.

,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 codes 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 69. Return and Reason Codes for the IWM4DIS Macro

Return Code	Reason Code	Equate Symbol, Meaning, and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.
4	xxxx0409	**Equate Symbol**: IwmRsnCodeNoConn **Meaning**: Input connection token does not reflect an active connection to WLM. **Action**: None required.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0801	**Equate Symbol**: IwmRsnCodeSrbMode **Meaning**: The caller is in SRB mode. **Action**: Avoid requesting this function while in SRB mode.
8	xxxx0802	**Equate Symbol**: IwmRsnCodeXmemUserKeyTkn **Meaning**: The caller is in cross-memory mode while the token was obtained in a user key. **Action**: Avoid requesting this function while in cross-memory mode.
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.

Table 69. Return and Reason Codes for the IWM4DIS Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: The caller is locked. **Action**: Avoid requesting this function while locked.
8	xxxx0809	**Equate Symbol**: IwmRsnCodeSrbUserKeyTkn **Meaning**: The caller is in SRB mode, while the token was obtained in a user key (8-F). **Action**: Avoid requesting this function in SRB mode for tokens associated with user key.
8	xxxx080A	**Equate Symbol**: IwmRsnCodeTcbNotOwnerUserKeyTkn **Meaning**: Current TCB is not the owner, while the token was obtained in a user key (8-F). **Action**: Avoid requesting this function under a TCB other than the owner for a token associated with user key.
8	xxxx080B	**Equate Symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list. **Action**: Check for possible storage overlay.
8	xxxx0810	**Equate Symbol**: IwmRsnCodeEutFrr **Meaning**: The caller has EUT FRR established. **Action**: Avoid requesting this function with an EUT FRR set.
8	xxxx0823	**Equate Symbol**: IwmRsnCodeDatoff **Meaning**: The caller invoked the service while DATOFF **Action**: Avoid requesting this function in this environment.
8	xxxx0824	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: The caller invoked the service but was in 24-bit addressing mode. **Action**: Request this function only when you are in 31-bit addressing mode.
8	xxxx0825	**Equate Symbol**: IwmRsnCodeAscModeNotPrimary **Meaning**: The caller invoked the service but was not DAT on Primary ASC mode. **Action**: Avoid requesting this function in this environment.
8	xxxx0826	**Equate Symbol**: IwmRsnCodeTaskTerm **Meaning**: The caller invoked the service while task termination is in progress for the TCB associated with the owner. **Action**: Avoid requesting this function in this environment.
8	xxxx0827	**Equate Symbol**: IwmRsnCodeRsvdNot0 **Meaning**: Reserved field in parameter list was non-zero. **Action**: Check for possible storage overlay of the parameter list.

Table 69. Return and Reason Codes for the IWM4DIS Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx0828	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: Version number in parameter list is not valid. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx082F	**Equate Symbol**: IwmRsnCodeWrongHome **Meaning**: The caller invoked the service from the wrong home address space. **Action**: Invoke the function with the correct home address space.
8	xxxx0841	**Equate Symbol**: IwmRsnCodeXMemMode **Meaning**: The caller is in cross-memory mode. **Action**: Invoke the function in non-cross memory mode.
8	xxxx084D	**Equate Symbol**: IwmRsnCodeNotAuthConnect **Meaning**: The caller must be supervisor state or have PSW key mask 0-7 authority to disconnect from the requested WLM services. **Action**: Avoid requesting this function in this environment.
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error. **Action**: No action required. The function may be successful if invoked again.

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 46.

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

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=WORKDEPENDENT`, SRM considers the new enclave to be a continuation of the creating unit of work's (TCB or SRB) transaction. The resulting enclave's type depends on the caller's execution environment: If the caller has joined or is scheduled into an enclave of type independent, the resulting enclave will be of type work-dependent and is regarded as an extension of the independent enclave's transaction. Classification and owner address space is adopted from the independent enclave. If the caller has joined or is scheduled into an enclave of type work-dependent, the resulting enclave will be of type work-dependent, as well. It is considered a part of the underlying independent enclave's transaction and inherits owner address space and classification from that independent enclave. If the caller has joined or is scheduled into a dependent enclave, the resulting enclave will be of type dependent. The new enclave is considered part of the creating enclave's (i.e., the enclave the caller is running in) owner address space's transaction and inherits its classification. The creating enclave's owner address space will become the owner of the new enclave. Finally, if no enclave has been joined when the service is called, the resulting enclave will be as if the service had been invoked with `TYPE=DEPENDENT` specified. Note that it is not allowed to invoke this service with `TYPE=WORKDEPENDENT` specified while running in a foreign enclave.

For `TYPE=MONENV` enclaves, SRM considers the enclave to be part of the address space's transaction which is delayed according to the input management monitoring environment, as set when IWM4MINI or IWM4MRLT 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 or 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. Work-dependent enclaves are implicitly deleted when the owning independent enclave is deleted.

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 TCB 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: This service was previously called IWMECREA for 31-bit addressing only (see "IWMECREA — Create an enclave" on page 885).

Environment

The requirements for the caller are:

Minimum authorization:	Supervisor state or program key mask (PKM) allowing keys 0-7.
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31- or 64-bit. If in 64-bit addressing mode, code `SYSSTATE AMODE64=YES` before invoking this macro.
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks may be held.
Control parameters:	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, 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. Prior to release z/OS V1.11, the caller has to assure that work-dependent enclave support is available on the system before calling this macro with

TYPE=WORKDEPENDENT. A flag (SRMSTWDP) in macro IRASRMST (which is returned by sysevent REQSRMST) indicates whether or not work-dependent enclaves can be created.

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 IWM4MCRE 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 IWM4MCRE 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 V2R1 or later levels for the MONTKN64 keyword.
6. This macro may only be used on OS/390 R12 or later levels for EXSTARTDEFER keyword.
7. This macro may only be used on z/OS V1R10 or later levels for the IMPORTANCE keyword.
8. This macro supports multiple versions. Some keywords are unique to certain versions. See the **PLISTVER** parameter description.

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:

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

```
►►──┬──────┬──b──IWM4ECRE──b───────────────────────────────────────────────────►
    └─name─┘

    ┌─TYPE=INDEPENDENT─┐
►───┼──────────────────┴──┤ parameters-1 ├─────────────────┬─,ETOKEN=etoken─────►
    ├─TYPE=DEPENDENT───────────────────────────────────────┤
    ├─TYPE=WORKDEPENDENT───────────────────────────────────┤
    └─TYPE=MONENV──┬─,MONTKN=montkn─────┬──┬─,ACCESS=PRIMARY─┬┘
                   └─,MONTKN64=montkn64─┘  └─,ACCESS=HOME────┘

►───┬─────────────────────────┬──┬───────────────────┬──┬───────────────────┬───►
    └─,IMPORTANCE=importance──┘  └─,RETCODE=retcode──┘  └─,RSNCODE=rsncode──┘

    ┌─,PLISTVER=IMPLIED_VERSION─┐  ┌─,MF=S──────────────────────────────────┐
►───┼───────────────────────────┴──┼────────────────────────────────────────┴──►◄
    ├─,PLISTVER=MAX─────────────┤  │                         ┌─,0D───┐      │
    ├─,PLISTVER=0───────────────┤  ├─,MF=(L─,list addr─┴───────┴─)───┤
    ├─,PLISTVER=1───────────────┤  │                         └─,attr─┘      │
    └─,PLISTVER=2───────────────┘  │                         ┌─,COMPLETE─┐  │
                                   └─,MF=(E─,list addr─┴───────────┴─)┘
```

parameters-1

```
                                            ┌─,INSERVCLS=0─────────┐
►►──,CLSFY=clsfy──┬───────────────────┬──┼──────────────────────┴──────────►
                  └─,SERVCLS=servcls──┘  └─,INSERVCLS=inservcls─┘
```

```
►─,ARRIVALTIME=arrivaltime──,FUNCTION_NAME=function_name─────────────────────►

   ┌─,EXSTARTDEFER=NO──┐    ┌─,ESTRT=IMPLIED─┐
►──┼───────────────────┼──┬─┴────────────────┴──┬───────────────────────┬──┬──────►◄
   └─,EXSTARTDEFER=YES─┘  │                     └─,WORKREQ_HDL=workreq_hdl─┘  │
                          ├─,ESTRT=EXPLICIT─────────────────────────────────┤
                          ├─,ESTRT=EXPLICIT_SINGLE──────────────────────────┤
                          └─,ESTRT=NEVER────────────────────────────────────┘
```

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 IWM4CLSY or IWMCLSFY. NOTE that this name is the data area name, not its pointer. IWM4CLSY MF(M) or IWMCLSFY MF(M) should be used to initialize the area prior to invocation of IWM4ECRE.

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 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=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
Same as option EXPLICIT above, but 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 the application must also add the EWLMMODE=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 (see 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. Moreover 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.

,IMPORTANCE=*importance*

An optional output parameter that will receive the importance value of the service class to which the unit of work is classified.

To code: Specify the RS-type address, or address in register (2)-(12), of a halfword field.

,INSERVCLS=*inservcls*
,INSERVCLS=0

When TYPE=INDEPENDENT is specified, an optional input parameter, which contains the service class token of a previous classification call. The caller must ensure that the classification attributes of the work unit matches the service class token. If the service class token is still valid an enclave is created and associated with the service and report class information contained in the token. If the service class token is not valid IWM4ECRE will perform the full classification by using the information of the CLSFY parameter block and return code 4, reason code IwmRsnCodeNewServcls. The default is 0.

To code: Specify the RS-type address, or address in register (2)-(12), of a fullword field, or specify a literal decimal value.

,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 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.

,MONTKN64=*montkn64*
When TYPE=MONENV is specified, a required input parameter which contains the long 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 64 bit field.

,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:

IMPORTANCE

- **2**, which supports both the following parameters and those from version 0 and 1:

INSERVCLS	MONTKN64
SERVCLS	

To code: Specify one of the following:

- IMPLIED_VERSION
- MAX
- A decimal value of 0, 1, or 2

,RETCODE=*retcode*
An optional output parameter into which the return code is to be copied from GPR 15. If you specify 15, GPR15, REG15, or R15 (within or without parentheses), the value will be left in GPR 15.

To code: Specify the RS-type address of a fullword field, or register (2)-(12) or (15), (GPR15), (REG15), or (R15).

,RSNCODE=*rsncode*
An optional output parameter into which the reason code is to be copied from GPR 0. If you specify 0, 00, GPR0, GPR00, REG0, REG00, or R0 (within or without parentheses), the value will be left in GPR 0.

To code: Specify the RS-type address of a fullword field, or register (0) or (2)-(12), (00), (GPR0), (GPR00), REG0), (REG00), or (R0).

,SERVCLS=*servcls*
When TYPE=INDEPENDENT is specified, an optional output parameter, when CLSFY has been specified which receives the service class token which matches the classification attributes. The token allows the caller to use fast path classification for work units which have the same classification attributes than the current work unit.

To code: Specify the RS-type address, or address in register (2)-(12), of a fullword field.

TYPE=INDEPENDENT
TYPE=DEPENDENT
TYPE=WORKDEPENDENT
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=WORKDEPENDENT
Indicates that the enclave represents a continuation of the creating unit of work's (TCB or SRB) transaction. The resulting enclave's type depends on the caller's execution environment: If the caller has joined or is scheduled into an enclave of type independent, the resulting enclave will be of type work-dependent and is regarded as an extension of the independent enclave's transaction. Classification and owner address space is adopted from the independent enclave. If the caller has joined or is scheduled into an enclave of type work-dependent, the resulting enclave will be of type work-dependent, as well. It is considered a part of the underlying independent enclave's transaction and inherits owner address space and classification from that independent enclave. If the caller has joined or is scheduled into a dependent enclave, the resulting enclave will be of type dependent. The new enclave is considered part of the creating enclave's (i.e., the enclave the caller is running in) owner address space's transaction and inherits its classification. The creating enclave's owner address space will become the owner of the new enclave. Finally, if no enclave has been joined when the service is called, the resulting enclave will be as if the service had been invoked with TYPE=DEPENDENT specified. Note that it is not allowed to invoke this service with TYPE=WORKDEPENDENT specified while running in a foreign enclave.

TYPE=MONENV
indicates that the Enclave represents a continuation of the business unit of work represented by the input management monitoring environment. TYPE=MONENV enclaves cannot be created for report-only monitoring environments.

,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 codes and reason codes

When the IWM4ECRE 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 70. Return and Reason Codes for the IWM4ECRE Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.
4	xxxx044E	**Equate Symbol**: IwmRsnCodeNewServcls **Meaning**: Input service class token is not valid. A new one has been assigned and returned in SERVCLS (if specified). **Action**: None required.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: Caller is locked. **Action**: Avoid requesting this function while locked.
8	xxxx080B	**Equate Symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list. **Action**: Check for possible storage overlay.
8	xxxx080C	**Equate Symbol**: IwmRsnCodeMonEnvLacksData **Meaning**: Input monitoring environment does not contain the necessary information. **Action**: Provide missing information.
8	xxxx0820	**Equate Symbol**: IwmRsnCodeBadMonEnv **Meaning**: Monitoring environment does not pass short form verification. **Action**: Check for possible storage overlay.
8	xxxx0821	**Equate Symbol**: IwmRsnCodeBadConn **Meaning**: Connect token from the input classify parameter list does not pass validity checking. **Action**: Check for possible storage overlay.
8	xxxx0824	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: Caller invoked service but was in 24 bit addressing mode. **Action**: Request this function only when you are in 31 bit addressing mode.

Table 70. Return and Reason Codes for the IWM4ECRE Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0828	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: Version number in parameter list is not valid. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx0836	**Equate Symbol**: IwmRsnCodeMaxEnclave **Meaning**: Enclave could not be created because the Enclave limit has been reached. **Action**: Check for possible problems wherein Enclaves are not being deleted as expected or excessive numbers of Enclaves are being created in a loop.
8	xxxx0837	**Equate Symbol**: IwmRsnCodeUserKeyConntkn **Meaning**: Connect token from the input classify parameter list is associated with a user key. **Action**: Invoke the function with a token associated with a system key.
8	xxxx0838	**Equate Symbol**: IwmRsnCodeClsfyAreaTooBig **Meaning**: Input area associated with classification information is larger than supported. **Action**: Invoke the function with an area of the proper size. Check for possible storage overlay.
8	xxxx0839	**Equate Symbol**: IwmRsnCodeClsfyPlTooSmall **Meaning**: Input Classify parameter list is too small. **Action**: Invoke the function with an area of the proper size. Check for possible storage overlay.
8	xxxx083B	**Equate Symbol**: IwmRsnCodeHomeNotOwnConn **Meaning**: Home address space does not own the connect token from the input classify parameter list. **Action**: Invoke the function with the correct home address space.
8	xxxx0840	**Equate Symbol**: IwmRsnCodeServiceNotEnabled **Meaning**: Caller's space connection is not enabled for this service. **Action**: Avoid requesting this function under the input connection. IWM4CON options must be specified previously to enable this service.
8	xxxx085D	**Equate Symbol**: IwmRsnCodeMonenvNotHome **Meaning**: The input monitoring environment is related to an address space other than home. **Action**: None required.
8	xxxx0872	**Equate Symbol**: IwmRsnCodeForeignEnclave **Meaning**: It is not allowed to create an enclave of type work-dependent from out of a foreign enclave. **Action**: Make sure not to have joined a forein enclave prior to calling IWM4ECRE with TYPE=WORKDEPENDENT.

Table 70. Return and Reason Codes for the IWM4ECRE Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0894	**Equate Symbol**: IwmRsnCodeInvalidEWLMCorr **Meaning**: 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 ('0004'x) or greater than 512 ('0200'x). **Action**: Check the specification of the EWLM correlator in the classification information.
8	xxxx0895	**Equate Symbol**: IwmRsnCodeEWLMServNotEnabled **Meaning**: Service is not enabled because caller invoked the IWM4CON service with EWLM=NO. **Action**: Specify the parameter WORKREQ_HDL only when connected with EWLM=YES.
C	—	**Equate Symbol**: IwmRetCodeEnvError **Meaning**: Environmental error.
C	xxxx0C01	**Equate Symbol**: IwmRsnCodeNoStg **Meaning**: Storage is not available for the request. **Action**: There is a storage shortage. The function may work successfully at a later time.
C	xxxx0C0C	**Equate Symbol**: IwmRsnCodeClassifyFail **Meaning**: Received a non-zero return code from the classification service, IWM4CLSY/IWMCLSFY. **Action**: No action required. Reinvoking the function later may succeed.
C	xxxx0C0D	**Equate Symbol**: IwmRsnCodeBadClsfy **Meaning**: Classification apparently can not access the current policy, possibly due to a policy switch in progress. **Action**: Invoke the function when the conditions are alleviated.
C	xxxx0C20	**Equate Symbol**: IwmRsnCodeDepClassifyFail **Meaning**: Unable to obtain classification attributes for a dependent enclave. **Action**: None required.
C	xxxx0C21	**Equate Symbol**: IwmRsnCodeNoMonEnvErr **Meaning**: Input monitoring token indicates no monitoring environment was established. **Action**: None required.
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error. **Action**: No action required. The function may be successful if invoked again.

Examples

None.

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 preemptible 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.

Note: This service was previously called IWMEDEL for 31-bit addressing only (see "IWMEDELE — Delete an enclave" on page 896).

Environment

The requirements for the caller are:

Minimum authorization:	Supervisor state or program key mask (PKM) allowing keys 0-7.
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31- or 64-bit. If in 64-bit addressing mode, code SYSSTATE AMODE64=YES before invoking this macro.
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks may be held. FRR environments may be established.
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, 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

The syntax of the IWM4EDEL macro is as follows:

```
►►──┬──────┬──IWM4EDEL──ETOKEN=etoken──┬────────────────────────┬──────────►
    └─name─┘                           └─,CPUSERVICE=cpuservice─┘

►──┬──────────────────────────────┬──┬──────────────────┬──────────────────►
   └─,SYSPLEXCPUSRV=sysplexcpusrv─┘  └─,CPUTIME=cputime─┘

►──┬──────────────────────────────────┬──┬──────────────────────────┬──────►
   └─,RESPTIME_RATIO=resptime_ratio───┘  └─,ZAAPSERVICE=zaapservice─┘

►──┬────────────────────┬──┬────────────────────────┬──┬──────────────────────────┬──►
   └─,ZAAPTIME=zaaptime─┘  └─,ZAAPNFACTOR=zaapnfactor┘  └─,ZIIPSERVICE=ziipservice─┘

►──┬────────────────────┬──┬──────────────────┬──┬──────────────────┬──────►
   └─,ZIIPTIME=ziiptime─┘  └─,RETCODE=retcode─┘  └─,RSNCODE=rsncode─┘

   ┌─,PLISTVER=IMPLIED_VERSION─┐  ┌─,MF=S────────────────────────────────┐
►──┼───────────────────────────┼──┼──────────────────────────────────────┼──────►◄
   ├─,PLISTVER=MAX─────────────┤  │                       ┌─,0D────┐     │
   ├─,PLISTVER=0───────────────┤  ├─,MF=(L─,list addr─────┼────────┼─)───┤
   ├─,PLISTVER=1───────────────┤  │                       └─,attr──┘     │
   └─,PLISTVER=2───────────────┘  │                       ┌─,COMPLETE─┐  │
                                  └─,MF=(E─,list addr─────┴───────────┴─)┘
```

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***,<u>0D</u>)**
,MF=(E,*list addr***)**
,MF=(E,*list addr***,<u>COMPLETE</u>)**

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.

,PLISTVER=<u>IMPLIED_VERSION</u>
,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 the following parameter and those from version 0:

  RESPTIME_RATIO
- **2**, which supports 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: act.resp.time / goal * 100 if the enclave has a response time goal (limited to: 1<=RESPTIME_RATIO<=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.

,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 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 codes 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 71. Return and Reason Codes for the IWM4EDEL Macro

Return Code	Reason Code	Equate Symbol, Meaning, and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.
4	xxxx0411	**Equate Symbol**: IwmRsnCodeEnclActive **Meaning**: Input enclave had 1 or more SRBs scheduled or running, or 1 or more TCBs joined to the enclave. **Action**: None required.
4	xxxx0449	**Equate Symbol**: IwmRsnCodeWDEDeleted **Meaning**: Enclave was deleted and one or several associated work-dependent enclaves were physically deleted. **Action**: None required.
4	xxxx044A	**Equate Symbol**: IwmRsnCodeActiveWDEDeleted **Meaning**: Enclave was deleted while it had one or several TCBs joined or SRBs scheduled/running. Additionally, one or several associated work- dependent enclaves were physically deleted. **Action**: None required.
4	xxxx044B	**Equate Symbol**: IwmRsnCodeAWDEDeleted **Meaning**: Enclave was deleted and one or several associated work-dependent enclaves were physically deleted. One or several physically deleted work-dependent enclaves had TCBs joined or SRBs scheduled/running. **Action**: None required.
4	xxxx044C	**Equate Symbol**: IwmRsnCodeActiveAWDEDeleted **Meaning**: Enclave was deleted and one or several associated work-dependent enclaves were physically deleted. The enclave itself and one or several physically deleted work-dependent enclaves had TCBs joined or SRBs scheduled/running **Action**: None required.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: The caller is locked. **Action**: Avoid requesting this function while locked.

Table 71. Return and Reason Codes for the IWM4EDEL Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx080B	**Equate Symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing the parameter list. **Action**: Check for possible storage overlay.
8	xxxx0824	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: The caller invoked the service but was in 24-bit addressing mode. **Action**: Request this function only when you are in 31-bit addressing mode.
8	xxxx0828	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: Version number in parameter list is not valid. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx083A	**Equate Symbol**: IwmRsnCodeBadEnclave **Meaning**: Enclave token does not pass verification. **Action**: Check for possible storage overlay of the enclave token, or asynchronous events which may have deleted the enclave.
8	xxxx0872	**Equate Symbol**: IwmRsnCodeForeignEnclave **Meaning**: The enclave is foreign. **Action**: Use the IWMUIMPT macro to delete a foreign enclave.
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error. **Action**: No action required. The function may be successful if invoked again.

Example

None.

IWM4EQRY — Query an enclave

This service offers three functions:

1. Query the classification attributes for an enclave.
2. Query WLM performance management information for an enclave.
3. Query both the classification attributes and WLM performance management information for an enclave.

The output of this service is mapped by macro IWMECDX.

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.

Note: This service was previously called IWMEQRY for 31-bit addressing only (see “IWMEQRY — Enclave query” on page 904).

Environment

The requirements for the caller are:

Minimum authorization:	Supervisor state or program key mask (PKM) allowing keys 0-7.
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31- or 64-bit. If in 64-bit addressing mode, code `SYSSTATE AMODE64=YES` before invoking this macro.
ASC mode:	Primary or access register (AR) If in Access Register ASC mode, specify `SYSSTATE ASCENV=AR` before invoking this macro.
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks may be held.
Control parameters:	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. This macro may not be used prior to the completion of WLM address space initialization

2. The caller must provide storage for an answer area mapped by macro IWMECDX. 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 IWM4EQRY 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

```
►►─┬──────┬─b─IWM4EQRY─b─ETOKEN=etoken─,ANSAREA=ansarea─,ANSLEN=anslen───►
   └─name─┘

►─,QUERYLEN=querylen─┬─,FUNCTION=CLASSINFO─┬─┬────────────────────┬──────►
                     ├─,FUNCTION=PERFINFO──┤ └─,RETCODE=retcode───┘
                     └─,FUNCTION=ALL───────┘
```

```
                                    ┌─,PLISTVER=IMPLIED_VERSION─┐
►─┬──────────────────┬──────────────┼───────────────────────────┼───────────────────────►
  └─,RSNCODE=rsncode─┘              ├─,PLISTVER=MAX─────────────┤
                                    └─,PLISTVER=0───────────────┘

  ┌─,MF=S───────────────────────────────────────┐
►─┼─────────────────────────────────────────────┼───────────────────────────────────────►◄
  │                           ┌─,0D───┐         │
  ├─,MF=(L─,list addr─────────┼───────┼──)──────┤
  │                           └─,attr─┘         │
  │                           ┌─,COMPLETE─┐     │
  └─,MF=(E─,list addr─────────┴───────────┴──)──┘
```

Parameters

The parameters are explained as follows:

name
: An optional symbol, starting in column 1, that is the name on the IWM4EQRY macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

,ANSAREA=*ansarea*
: A required output parameter, which is to receive the data being returned. The layout of this area is defined by macro IWMECDX.

  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 IWM4EQRY.

  To code: Specify the RS-type address, or address in register (2)-(12), of a fullword field, or specify a literal decimal value.

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.

,FUNCTION=CLASSINFO
,FUNCTION=PERFINFO
,FUNCTION=ALL
: A required parameter that indicates the query function to be executed.

  ,FUNCTION=CLASSINFO
  : Use this function to query the classification attributes of an enclave. This is the same information as is returned by service IWMECQRY.

  ,FUNCTION=PERFINFO
  : Use this function to query the WLM performance management information of an enclave. This data is based on the classification attributes and the active WLM Policy.

  ,FUNCTION=ALL
  : Use this function to query both, the classification attributes and the WLM performance management information of an enclave.

,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`

,QUERYLEN=*querylen*
A required output parameter, variable is to receive the number of bytes needed to contain the data being returned by IWM4EQRY. The length of the area needed to contain the data is dependent on the Function being used. If the ANSLEN is less than the QUERYLEN, then no data is returned in the output area specified by ANSAREA and a return code of 4 is issued.

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. If you specify 15, GPR15, REG15, or R15 (within or without parentheses), the value will be left in GPR 15.

To code: Specify the RS-type address of a fullword field, or register (2)-(12) or (15), (GPR15), (REG15), or (R15).

,RSNCODE=*rsncode*
An optional output parameter into which the reason code is to be copied from GPR 0. If you specify 0, 00, GPR0, GPR00, REG0, REG00, or R0 (within or without parentheses), the value will be left in GPR 0.

To code: Specify the RS-type address of a fullword field, or register (0) or (2)-(12), (00), (GPR0), (GPR00), REG0), (REG00), or (R0).

ABEND codes

None.

Return codes and reason codes

When the IWM4EQRY 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 72. Return and Reason Codes for the IWM4EQRY Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.

Table 72. Return and Reason Codes for the IWM4EQRY Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
4	xxxx040A	**Equate Symbol**: IwmRsnCodeOutputAreaTooSmall **Meaning**: The output area supplied is too small to receive all the available information. **Action**: Reinvoke the service with an output area of sufficient size to receive all information.
4	xxxx043C	**Equate Symbol**: IwmRsnCodeIsReset **Meaning**: 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. Information returned. **Action**: None required.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: Caller is locked. **Action**: Avoid requesting this function while locked.
8	xxxx080B	**Equate Symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list. **Action**: Check for possible storage overlay.
8	xxxx0824	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: Caller invoked service but was in 24 bit addressing mode. **Action**: Request this function only when you are in 31-bit or 64-bit addressing mode.
8	xxxx0828	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: Caller invoked service with an invalid value for PLISTVER. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx0830	**Equate Symbol**: IwmRsnCodeBadAlet **Meaning**: Caller has an invalid ALET. The ALET is used to address the output area specified in parameter ANSAREA. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx083A	**Equate Symbol**: IwmRsnCodeBadEnclave **Meaning**: Enclave token is invalid. **Action**: Check the specification of the ETOKEN parameter.

Table 72. Return and Reason Codes for the IWM4EQRY Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error **Action**: Consider reporting the problem to IBM.

Examples

None.

IWM4HLTH — Setting 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.

Value	**Meaning**
100	The server is fully capable to do work without any health problems.
0	The server is not able to do any work.
Any value between 0 and 100	Indicates the level of health of the server.

IWM4HLTH provides two functions: SET and RESET. With the SET function, which is the main intended use and default function, a caller informs WLM about its view of the health state of a server. WLM then sets the server's health indicator to the minimum number of all the current settings from the different callers of the service since the last RESET.

The RESET function primarily refers to reliability, availability, and serviceability (RAS) considerations regarding a server's health state. RESET restarts setting the health indicator by specifying an initial value and discarding any values reported by other callers before.

Callers can identify themselves by a subsystem type and subsystem name. The service uses these parameters to recognize different callers of the service. If no subsystem type is passed, the job name of the caller address space is used instead. This information will then also be available to callers of the IWM4QHLT (Query Server Health Indicators) service.

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:

Minimum authorization:	**FUNCTION=SET**: Problem state with any PSW key if the address space token specified with `STOKEN=`*`stoken`* equals the address space token of the home address space. That is, the caller provides a health indicator for itself. When providing a health indicator for an address space other than the home address space, the minimum authorization is one of the following: • Supervisor state. • Program key mask (PKM) allowing at least one of the keys 0-7. • The caller has UPDATE authority to the resource IWM.SERVER.HEALTH in the FACILITY class. **FUNCTION=RESET**: The minimum authorization is one of the the following: • Supervisor state. • Program key mask (PKM) allowing at least one of the key 0-7. • The caller has CONTROL authority to the resource IWM.SERVER.HEALTH in the FACILITY class.
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31-bit or 64-bit. If in 64-bit addressing mode, code `SYSSTATE AMODE64=YES` before invoking this macro.
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	The caller may hold locks, but is not required to hold any. FRRs may be established.
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, 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 supports multiple versions. Some keywords are unique to certain versions. See the PLISTVER parameter description.

Input register information

Before issuing the IWM4HLTH 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

The syntax of the IWM4HLTH macro is as follows:

```
                                              ┌─,FUNCTION=SET───┐
►►──┬──────┬──IWM4HLTH──STOKEN=stoken──┼─────────────────┼──,HEALTH=health──────────────►
    └─name─┘                                  └─,FUNCTION=RESET─┘

  ┌─,SUBSYS=NO_SUBSYS─┐   ┌─,SUBSYSNM=NO_SUBSYSNM─┐   ┌─,HEALTHRSN=NO_RSN──────┐
►─┼───────────────────┼───┼───────────────────────┼───┼────────────────────────┼────────►
  └─,SUBSYS=subsys────┘   └─,SUBSYSNM=subsysnm────┘   └─,HEALTHRSN=healthrsn───┘
```

```
                                                        ┌─,PLISTVER=IMPLIED_VERSION─┐
►──┬────────────────────┬──┬────────────────────┬──────┼───────────────────────────┼──────────────────────────►
   └─,RETCODE=retcode───┘  └─,RSNCODE=rsncode───┘      ├─,PLISTVER=MAX─────────────┤
                                                        ├─,PLISTVER=0───────────────┤
                                                        └─,PLISTVER=1───────────────┘

   ┌─,MF=S──────────────────────────────────────┐
►──┼────────────────────────────────────────────┼─────────────────────────────────────────────────────────►◄
   │                          ┌─,0D────┐        │
   ├─,MF=(L─,list addr────────┼────────┼─)──────┤
   │                          └─,attr──┘        │
   │                          ┌─,COMPLETE─┐     │
   └─,MF=(E─,list addr────────┴───────────┴──)──┘
```

Parameters

The parameters are explained as follows:

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.

FUNCTION
An optional parameter which indicates the function to perform. The default is FUNCTION=SET.

FUNCTION=SET
Informs WLM about the caller's view of the health state of a server. WLM then sets the server's health indicator to the minimum number of all the current settings from the different callers of the service since the last RESET.

FUNCTION=RESET
Restarts setting the health indicator by specifying an initial value and discarding any values reported by other callers before. RESET primarily refers to reliability, availability, and serviceability (RAS) considerations regarding a server's health state.

HEALTH
A required input parameter, which contains the health indicator associated with the address space. This value is 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.

SUBSYS
An optional input parameter which contains the generic name or type of the caller of the service. It is used by WLM together with the SUBSYSNM parameter to recognize different callers of this service. This data is also available to callers of the IWM4QHLT service.

The default is NO_SUBSYS which indicates that no type was passed.

To code: Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

SUBSYSNM
An optional input parameter which contains the name of a specific instance of the caller of the service. It is used by WLM together with the SUBSYS

parameter to recognize different callers of this service. This data is also available to callers of the IWM4QHLT service.

The default is NO_SUBSYSNM which indicates that no name was passed.

To code: Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

HEALTHRSN

An optional input parameter that allows the caller to pass additional information, such as the reason for changing the health indicator. This data is available to callers of the IWM4QHLT service.

The format is undefined. The default is NO_RSN which indicates that no additional information is passed.

To code: Specify the RS-type address or address in register (2)-(12) of an of an 16-character field.

RETCODE

An optional output parameter into which the return code is to be copied from GPR 15.

To code: Specify the RS-type address or address in register (2)-(12) of a fullword field.

RSNCODE

An optional output parameter into which the reason code is to be copied from GPR 0.

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

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

Indicates that 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 Indicates to use version 0 of the macro, which supports all parameters except those specifically identified in higher versions.

1 Supports the following parameters and those from version 0:
FUNCTION
SUBSYS
SUBSYSNM
HEALTHRSN

To code: Specify one of the following:
- IMPLIED_VERSION
- MAX
- A decimal value of 0 or 1

,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.

ABEND codes

None.

Return codes 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.

Table 73 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 codes and reason codes for the IWM4HLTH Macro

Return code	Reason code	Equate symbol, meaning, and action
0	—	**Equate symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
8	—	**Equate symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	*xxxx*0807	**Equate symbol**: IwmRsnCodeBadSTOKEN **Meaning**: Bad STOKEN passed. **Action**: Check for possible storage overlay.
8	*xxxx*080B	**Equate symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list. **Action**: Check for possible storage overlay.
8	*xxxx*0823	**Equate symbol**: IwmRsnCodeDatoff **Meaning**: The caller invoked the service while DATOFF **Action**: Avoid requesting this function in this environment.
8	*xxxx*0824	**Equate symbol**: IwmRsnCodeAmode24 **Meaning**: The caller invoked the service but was in 24-bit addressing mode. **Action**: Request this function only when you are in 31-bit addressing mode.
8	*xxxx*0825	**Equate symbol**: IwmRsnCodeAscModeNotPrimary **Meaning**: The caller invoked the service but was not DAT on Primary ASC mode. **Action**: Avoid requesting this function in this environment.
8	*xxxx*0827	**Equate symbol**: IwmRsnCodeRsvdNot0 **Meaning**: Reserved field in parameter list was non-zero. **Action**: Check for possible storage overlay of the parameter list.
8	*xxxx*0828	**Equate symbol**: IwmRsnCodeBadVersion **Meaning**: The version number in the parameter list or the version length field is not valid. Or this service was called on a z/OS release where it is not supported. **Action**: Check for possible storage overlay of the parameter list.

Table 73. Return codes and reason codes for the IWM4HLTH Macro (continued)

Return code	Reason code	Equate symbol, meaning, and action
8	*xxxx*0829	**Equate symbol**: IwmRsnCodeBadOptions **Meaning**: Parameter list omits required parameters, supplies mutually exclusive parameters, or provides data associated with options not selected. **Action**: Check for possible storage overlay of the parameter list.
8	*xxxx*08A2	**Equate symbol**: IwmRsnCodeBadHealth **Meaning**: Health Value out of range **Action**: Check for possible storage overlay.
C	—	**Equate symbol**: IwmRetCodeEnvError **Meaning**: Environmental error.
C	*xxxx*0C01	**Equate symbol**: IwmRsnCodeNoStg **Meaning**: Storage is not available for the request. **Action**: There is a storage shortage. The function may work successfully at a later time.
C	*xxxx*0C0E	**Equate symbol**: IwmRsnCodeInsufAccess **Meaning**: Minimum authorization requirements are not fulfilled. **Action**: Invoke the service with one of the following authorization requirements fulfilled: • Change the caller's authorization to supervisor state or PKM allowing at least one of the keys 0-7. • Give the user ID associated with the program UPDATE authority to the resource profile IWM.SERVER.HEALTH.
10	—	**Equate symbol**: IwmRetCodeCompError **Meaning**: Component error. **Action**: No action required. The function may be successful if invoked again.

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
```

IWM4MABN — Monitor environment abnormal event

The purpose of this service is to indicate that an abnormal event has occurred for the work request represented by the input monitoring environment. This condition will supplement any existing abnormal conditions recorded in the input monitoring environment (multiple conditions may exist).

Note that abnormal conditions are propagated to the parent monitoring environment via IWM4MXFR Function(Return).

Note: This service was previously called IWMMABNL for 31-bit addressing only (see "IWMMABNL — Record abnormal event" on page 911).

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 input monitoring token when `MONENVKEYP(PSWKEY)` is specified. • `MONENVKEYP(VALUE)` may only be specified in supervisor state or with PKM authority to the key specified by **`MONENVKEY`**. Note that the key for IWM4MABN 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. • `MONENVKEYP(UNKNOWN)` may only be specified in supervisor state or with PKM authority to key 0.
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31- or 64-bit. If in 64-bit addressing mode, code `SYSSTATE AMODE64=YES` before invoking this macro.
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	Unlocked
Control parameters:	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.
3. If the key specified on IWM4MCRE for the input environment was a user key (8-F), then either primary OR secondary addressability must exist to the performance block.
4. The caller must serialize to prevent any delay monitoring services from being invoked concurrently for the input monitoring environment.
5. Only limited validity checking is done on the input monitoring token.

6. Caller is responsible for error recovery.
7. This macro may only be used on z/OS V2R1 or higher.

Input register information

Before issuing the IWM4MABN 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 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

```
►►──┬──────┬──b──IWM4MABN──b──ABNORMAL=abnormal──┬─,MONTKN=montkn─────┬──────────────►
    └─name─┘                                     └─,MONTKN64=montkn64─┘

►──┬─,MONENVKEYP=VALUE─,MONENVKEY=monenvkey─┬──┬─,MONENV=NOSWITCH──┬──────────────►
   ├─,MONENVKEYP=PSWKEY─────────────────────┤  └─,MONENV=SECONDARY─┘
   └─,MONENVKEYP=UNKNOWN────────────────────┘
```

```
►─┬──────────────────┬──┬──────────────────┬──────────────────────────────────►
  └─,RETCODE=retcode─┘  └─,RSNCODE=rsncode─┘

                        ┌─,COMPLETE─┐
►─,MF=(M─,list addr─────┼───────────┼─)───────────────────────────────────────►◄
                        └─,NOCHECK──┘
```

Parameters

The parameters are explained as follows:

name
: An optional symbol, starting in column 1, that is the name on the IWM4MABN macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

ABNORMAL=*abnormal*
: A required input parameter, which indicates the abnormal mask to use to reflect the abnormality. Macro IWMYCON contains the defined abnormal masks. The mask variable names begin with IWM4MABN, for example - IWMMABNL_SCOPE_LOCALMVS.

  To code: Specify the RS-type address, or address in register (2)-(12), of a 32 bit field.

,MONENV=NOSWITCH
,MONENV=SECONDARY
: A required parameter, which describes whether a space switch is needed to access the input monitoring environment.

  ,MONENV=NOSWITCH

  indicates that NO space switch is needed to access the input monitoring environment. This would be 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.

  ,MONENV=SECONDARY

  indicates that the input monitoring environment was established in current secondary (for use by a specific user key).

,MONENVKEY=*monenvkey*
: When MONENVKEYP=VALUE is specified, a required input parameter, which contains the key in which the input 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 leftmost, i.e. high order, 4 bits 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.

,MONENVKEYP=VALUE
,MONENVKEYP=PSWKEY

,MONENVKEYP=UNKNOWN
A required parameter, which describes whether a key switch is needed to access the input monitoring environment.

,MONENVKEYP=VALUE

indicates that the key is being passed explicitly via MONENVKEY.

,MONENVKEYP=PSWKEY

indicates that the current PSW key should be used. Use of this keyword value requires that the input monitoring environment was established with the same key as the current PSW.

,MONENVKEYP=UNKNOWN

indicates that the key associated with the input 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.

,MONTKN=*montkn*
A required input parameter which contains the delay monitoring token for the environment affected by the abnormality.

To code: Specify the RS-type address, or address in register (2)-(12), of a 32 bit field.

,MONTKN64=*montkn64*
A required input parameter which contains the long delay monitoring token for the environment affected by the abnormality.

To code: Specify the RS-type address, or address in register (2)-(12), of a 64 bit field.

,RETCODE=*retcode*
An optional output parameter into which the return code is to be copied from GPR 15. If you specify 15, GPR15, REG15, or R15 (within or without parentheses), the value will be left in GPR 15.

To code: Specify the RS-type address of a fullword field, or register (2)-(12) or (15), (GPR15), (REG15), or (R15).

,RSNCODE=*rsncode*
An optional output parameter into which the reason code is to be copied from GPR 0. If you specify 0, 00, GPR0, GPR00, REG0, REG00, or R0 (within or without parentheses), the value will be left in GPR 0.

To code: Specify the RS-type address of a fullword field, or register (0) or (2)-(12), (00), (GPR0), (GPR00), REG0), (REG00), or (R0).

ABEND codes

None.

Return codes and reason codes

When the IWM4MABN 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. IBM support personnel may request the entire reason code, including the *xxxx* value.

Table 74. Return and reason codes for the IWM4MABN macro

Return code	Reason code	Meaning and action
0	—	**Equate symbol:** IwmRetCodeOk **Meaning:** Successful completion.
4	—	**Equate symbol:** IwmRetCodeWarning **Meaning:** Successful completion, unusual conditions noted.
4	*xxxx*0402	**Equate symbol:** IwmRsncodeNoMonEnv **Meaning:** Input monitoring token indicates no monitoring environment was established.
8	—	**Equate symbol:** IwmRetCodeInvocError **Meaning:** Invalid invocation environment or parameters .
8	*xxxx*0820	**Equate symbol:** IwmRsnCodeBadMonEnv **Meaning:** Monitoring environment does not pass verification.

Examples

None.

IWM4MCHS — Change the state of a work request

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.

A work unit started by IWM4MSTR is blocked and unblocked automatically. WAITING states, that allow the specification of the ASYNC keyword, block the work unit. All other states unblock the work unit.

Note: This service was previously called IWMMCHST for 31-bit addressing only (see "IWMMCHST — Monitor change state of work unit" on page 915).

Environment

The requirements for the caller are:

Minimum authorization:	Problem state. PSW key must either be 0 or match the value supplied on IWM4MCRE.
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31- or 64-bit. If in 64-bit addressing mode, code `SYSSTATE AMODE64=YES` before invoking this macro.
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	Suspend locks are allowed.
Control parameters:	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 bits 0-31 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.
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 levels for the following resources
   - RESOURCE(BUFFER_POOL_IO)
7. This macro may only be used on z/OS R8 or higher levels for RESTKN keyword.
8. This macro may only be used on z/OS R10 or higher levels for the following state/resources
   - RESOURCE(TYPE6)
   - RESOURCE(TYPE7)
   - RESOURCE(TYPE8)
   - RESOURCE(TYPE9)
   - RESOURCE(TYPE10)
   - RESOURCE(TYPE11)
   - RESOURCE(TYPE12)
   - RESOURCE(TYPE13)
   - RESOURCE(TYPE14)
   - RESOURCE(TYPE15)
9. This macro may only be used on z/OS V2R1 or higher levels for the MONTKN64 keyword.
10. This macro supports multiple versions. Some keywords are unique to certain versions. See the PLISTVER parameter description.

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:

Register	**Contents**
0	Reason code if GR15 return code is non-zero. The reason code is stored in bits 0-31
1	Used as work registers by the system
2-13	Unchanged
14	Used as work registers by the system
15	Return code stored in bits 0-31

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

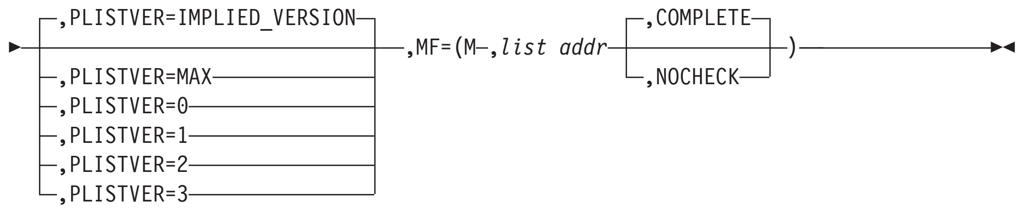
None.

Syntax

main diagram

```
►►──┬──────┬──b──IWM4MCHS──b──────────────────────────────────────────►
    └─name─┘
```

```
                                                                               ┌─,EWLM=NO─┐
►──┬─STATE=FREE──────────────────────────────────────────────────────────┬────┴──────────┴────►
   ├─STATE=ACTIVE────────────────────────────────────────────────────────┤
   ├─STATE=ACTIVE_APPL───────────────────────────────────────────────────┤
   ├─STATE=READY─────────────────────────────────────────────────────────┤
   ├─STATE=IDLE──────────────────────────────────────────────────────────┤
   └─STATE=WAITING──┬─,RESOURCE=LATCH─────────────────────────────────┬──┘
                    ├─,RESOURCE=LOCK──────────────────────────────────┤
                    ├─,RESOURCE=IO────────────────────────────────────┤
                    │                   ┌─,ASYNC=NO──┐                │
                    ├─,RESOURCE=CONV────┴────────────┴────────────────┤
                    │                   └─,ASYNC=YES─┘                │
                    │                      ┌─,ASYNC=NO──┐             │
                    ├─,RESOURCE=DISTRIB────┴────────────┴─────────────┤
                    │                      └─,ASYNC=YES─┘             │
                    │                            ┌─,ASYNC=NO──┐       │
                    ├─,RESOURCE=SESS_LOCALMVS────┴────────────┴───────┤
                    │                            └─,ASYNC=YES─┘       │
                    │                           ┌─,ASYNC=NO──┐        │
                    ├─,RESOURCE=SESS_NETWORK────┴────────────┴────────┤
                    │                           └─,ASYNC=YES─┘        │
                    │                           ┌─,ASYNC=NO──┐        │
                    ├─,RESOURCE=SESS_SYSPLEX────┴────────────┴────────┤
                    │                           └─,ASYNC=YES─┘        │
                    ├─,RESOURCE=TIMER─────────────────────────────────┤
                    │                            ┌─,ASYNC=NO──┐       │
                    ├─,RESOURCE=OTHER_PRODUCT────┴────────────┴───────┤
                    │                            └─,ASYNC=YES─┘       │
                    ├─,RESOURCE=MISC──────────────────────────────────┤
                    ├─,RESOURCE=SSL_THREAD────────────────────────────┤
                    ├─,RESOURCE=REG_THREAD────────────────────────────┤
                    ├─,RESOURCE=REG_TO_WRKTB──────────────────────────┤
                    ├─,RESOURCE=TYPE1─────────────────────────────────┤
                    ├─,RESOURCE=TYPE2─────────────────────────────────┤
                    ├─,RESOURCE=TYPE3─────────────────────────────────┤
                    ├─,RESOURCE=TYPE4─────────────────────────────────┤
                    ├─,RESOURCE=TYPE5─────────────────────────────────┤
                    ├─,RESOURCE=TYPE6─────────────────────────────────┤
                    ├─,RESOURCE=TYPE7─────────────────────────────────┤
                    ├─,RESOURCE=TYPE8─────────────────────────────────┤
                    ├─,RESOURCE=TYPE9─────────────────────────────────┤
                    ├─,RESOURCE=TYPE10────────────────────────────────┤
                    ├─,RESOURCE=TYPE11────────────────────────────────┤
                    ├─,RESOURCE=TYPE12────────────────────────────────┤
                    ├─,RESOURCE=TYPE13────────────────────────────────┤
                    ├─,RESOURCE=TYPE14────────────────────────────────┤
                    ├─,RESOURCE=TYPE15────────────────────────────────┤
                    └─,RESOURCE=BUFFER_POOL_IO────────────────────────┘

                                 ┌─,RESTKN=NORESTKN─┐   ┌─,RUNTIME_VER=SHORT_FORM─┐
►──┬─,MONTKN=montkn──────┬───────┴──────────────────┴───┴─────────────────────────┴────►
   └─,MONTKN64=montkn64──┘       └─,RESTKN=restkn───┘   └─,RUNTIME_VER=MINIMAL────┘

   ┌─,COMPCODE=YES─┐
►──┴───────────────┴───┬───────────────────┬───┬───────────────────┬──────────────────►
   └─,COMPCODE=NO──┘   └─,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 IWM4MCHS macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

,ASYNC=NO
,ASYNC=YES
When RESOURCE=CONV and STATE=WAITING are specified, an optional parameter, which specifies the blocking type of a work unit started by IWM4MSTR. The default is ASYNC=NO.

,ASYNC=NO
indicates that the blocking type is synchronous.

,ASYNC=YES
indicates that the blocking type is asynchronous.

,ASYNC=NO
,ASYNC=YES
When RESOURCE=DISTRIB and STATE=WAITING are specified, an optional parameter, which specifies the blocking type of a work unit started by IWM4MSTR. The default is ASYNC=NO.

,ASYNC=NO
indicates that the blocking type is synchronous.

,ASYNC=YES
indicates that the blocking type is asynchronous.

,ASYNC=NO
,ASYNC=YES
When RESOURCE=SESS_LOCALMVS and STATE=WAITING are specified, an optional parameter, which specifies the blocking type of a work unit started by IWM4MSTR. The default is ASYNC=NO.

,ASYNC=NO
indicates that the blocking type is synchronous.

,ASYNC=YES
indicates that the blocking type is asynchronous.

,ASYNC=NO
,ASYNC=YES
When RESOURCE=SESS_NETWORK and STATE=WAITING are specified, an optional parameter, which specifies the blocking type of a work unit started by IWM4MSTR. The default is ASYNC=NO.

,ASYNC=NO
indicates that the blocking type is synchronous.

,ASYNC=YES
indicates that the blocking type is asynchronous.

,ASYNC=NO
,ASYNC=YES
When RESOURCE=SESS_SYSPLEX and STATE=WAITING are specified, an optional parameter, which specifies the blocking type of a work unit started by IWM4MSTR. The default is ASYNC=NO.

,ASYNC=NO
indicates that the blocking type is synchronous.

,ASYNC=YES
indicates that the blocking type is asynchronous.

,ASYNC=NO
,ASYNC=YES
When RESOURCE=OTHER_PRODUCT and STATE=WAITING are specified, an optional parameter, which specifies the blocking type of a work unit started by IWM4MSTR. The default is ASYNC=NO.

,ASYNC=NO
indicates that the blocking type is synchronous.

,ASYNC=YES
indicates that the blocking type is asynchronous.

,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.

,EWLM=NO
An optional parameter, which indicates if this work manager intents 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.

,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.

,MONTKN64=*montkn64*
A required input parameter which contains the long delay monitoring token.

To code: Specify the RS-type address, or address in register (2)-(12), of a 64 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. 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 all parameters except those specifically referenced in higher versions. No parameters correspond to this version number.
- **2**, which supports all parameters except those specifically referenced in higher versions. No parameters correspond to this version number.
- **3**, which supports both the following parameters and those from version 0,1 and 2:

MONTKN64

To code: Specify one of the following:

- IMPLIED_VERSION
- MAX
- A decimal value of 0, 1, 2, or 3

,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
,RESOURCE=TYPE6
,RESOURCE=TYPE7
,RESOURCE=TYPE8
,RESOURCE=TYPE9
,RESOURCE=TYPE10
,RESOURCE=TYPE11
,RESOURCE=TYPE12

,RESOURCE=TYPE13
,RESOURCE=TYPE14
,RESOURCE=TYPE15
,RESOURCE=BUFFER_POOL_IO
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 (e.g. locks, buffers, etc.).

,RESOURCE=CONV
indicates that the work manager is waiting on a conversation. This may be used in conjunction with IWM4MSWC 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 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 a 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.

,RESOURCE=TYPE6
indicates that the work manager is waiting for resource type 6.

,RESOURCE=TYPE7
indicates that the work manager is waiting for resource type 7.

,RESOURCE=TYPE8
indicates that the work manager is waiting for resource type 8.

,RESOURCE=TYPE9
indicates that the work manager is waiting for resource type 9.

,RESOURCE=TYPE10
indicates that the work manager is waiting for resource type 10.

,RESOURCE=TYPE11
indicates that the work manager is waiting for resource type 11.

,RESOURCE=TYPE12
indicates that the work manager is waiting for resource type 12.

,RESOURCE=TYPE13
indicates that the work manager is waiting for resource type 13.

,RESOURCE=TYPE14
indicates that the work manager is waiting for resource type 14.

,RESOURCE=TYPE15
indicates that the work manager is waiting for resource type 15.

,RESOURCE=BUFFER_POOL_IO
indicates that the work manager is waiting for resource buffer pool IO.

,RESTKN=*restkn*
,RESTKN=NORESTKN
An optional input parameter, which contains the token of the managed resource previously registered with Register Resource (IWM4MREG) The default is NORESTKN which indicates that no resource token is provided.

NORESTKN will preserve 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. If you specify 15, GPR15, REG15, or R15 (within or without parentheses), the value will be left in GPR 15.

To code: Specify the RS-type address of a fullword field, or register (2)-(12) or (15), (GPR15), (REG15), or (R15).

,RSNCODE=*rsncode*
An optional output parameter into which the reason code is to be copied from GPR 0. If you specify 0, 00, GPR0, GPR00, REG0, REG00, or R0 (within or without parentheses), the value will be left in GPR 0.

To code: Specify the RS-type address of a fullword field, or register (0) or (2)-(12), (00), (GPR0), (GPR00), REG0), (REG00), or (R0).

,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 a 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, etc. 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.

STATE=WAITING
indicates that the work manager is waiting for a resource on behalf of the work request described by the the monitoring environment. Some resources the work manager is waiting for cause a blocking of the work unit started by IWM4MSTR. The blocking is terminated when the work unit state is changed into a state that does not cause blocking.

ABEND codes

None.

Return codes and reason codes

When the IWM4MCHS 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 75. Return and Reason Codes for the IWM4MCHS Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.
4	xxxx0402	**Equate Symbol**: IwmRsnCodeNoMonEnv **Meaning**: Input monitoring token indicates no monitoring environment was established. **Action**: None required.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0820	**Equate Symbol**: IwmRsnCodeBadMonEnv **Meaning**: Monitoring environment does not pass short form verification. **Action**: Check for possible storage overlay.

Examples

None.

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, including performance management for buffer pools.

Report-only monitoring environments
Can only be used for performance reporting.

Buffer pool management only environments
If you do not use management monitoring environments, buffer pool management only environments can be used to report buffer pool delays to manage buffer pools with enclaves. If used for buffer pool management, they can also be used to report other delays for performance reporting.

Optionally with this macro, you can use the `REPORTONLY=YES` parameter to specify that the monitoring environment will be used for reporting purposes only.

Optionally with this macro, you can use the `BPMGMTONLY=YES` parameter to specify that the monitoring environment will be used together with enclaves for buffer pool management.

Note: This service was previously called IWMMCREA for 31-bit addressing only (see "IWMMCREA — Create delay monitoring environment" on page 922).

Environment

The requirements for the caller are:

Minimum authorization:	Supervisor state or program key mask (PKM) allowing keys 0-7.
Dispatchable unit mode:	Task
Cross memory mode:	Non-XMEM or XMEM. Any P.S.H.
AMODE:	31- or 64-bit. If in 64-bit addressing mode, code `SYSSTATE AMODE64=YES` before invoking this macro.
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts No (EUT) FRR established.
Locks:	No locks may be held.
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, 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 supports multiple versions. Some keywords are unique to certain versions. See the **PLISTVER** parameter description.

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:

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

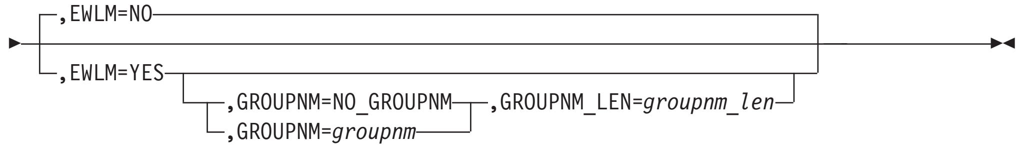
```
►►──┬──────┬──b──IWM4MCRE──b─────────────────────────────────────────────────►
    └─name─┘

        ┌─REQTYPE=SINGLE─┐
►──┬────┴────────────────┴──┬─,MONTKN=montkn─────────────────────────────┬──┬──►
   │                        │                       ┌─,ALLOCATEBELOW=NO──┐ │  │
   │                        └─,MONTKN64=montkn64────┴─,ALLOCATEBELOW=YES─┴─┘  │
   └─REQTYPE=MULTIPLE─┤ parameters-1 ├────────────────────────────────────────┘

     ┌─,REPORTONLY=NO─┐  ┌─,BPMGMTONLY=NO──┐
►──┬─┴────────────────┴──┴─,BPMGMTONLY=YES─┴─┬────────────────────────────────►
   └─,REPORTONLY=YES─────────────────────────┘

►──┬─,SUBSYSP=CONNECT─,CONNTKN=conntkn─┬──────────────────────────────────────►
   └─,SUBSYSP=VALUE─┤ parameters-2 ├───┘

►──┬─,MONTKNKEYP=VALUE─,MONTKNKEY=montknkey─┬──┬──────────────────┬────────────►
   └─,MONTKNKEYP=PSWKEY─────────────────────┘  └─,RETCODE=retcode─┘

                             ┌─,PLISTVER=IMPLIED_VERSION─┐
►──┬────────────────────┬────┼───────────────────────────┼────────────────────►
   └─,RSNCODE=rsncode───┘    ├─,PLISTVER=MAX─────────────┤
                             ├─,PLISTVER=0───────────────┤
                             ├─,PLISTVER=1───────────────┤
                             └─,PLISTVER=2───────────────┘

     ┌─,MF=S──────────────────────────────────┐
►──┬─┴────────────────────────────────────────┴┬──────────────────────────────►◄
   │                         ┌─,0D───┐         │
   ├─,MF=(L─,list addr───────┴─,attr─┴─)───────┤
   │                         ┌─,COMPLETE─┐     │
   └─,MF=(E─,list addr───────┴───────────┴─)───┘
```

parameters-1

```
►►──,AMOUNT=amount──┬─,MONTKN_LIST=montkn_list─────────────────────────────┬──►
                    │                               ┌─,ALLOCATEBELOW=NO──┐ │
                    └─,MONTKN64_LIST=montkn64_list──┴─,ALLOCATEBELOW=YES─┴─┘

►──┬─,LISTLEN=listlen──────────────┬──────────────────────────────────────────►◄
   └─,MONTKN_LISTLEN=montkn_listlen┘
```

parameters-2

```
►►──,SUBSYS=subsys──,SUBSYSNM=subsysnm────────────────────────────────────────►
```



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.

,ALLOCATEBELOW=NO
,ALLOCATEBELOW=YES

When MONTKN64=*montkn64* and REQTYPE=SINGLE are specified, an optional parameter, which indicates whether the virtual storage for the delay monitoring environment is to be obtained below 2 gigabytes. This is especially helpful for callers with 31-bit dependencies. The default is ALLOCATEBELOW=NO.

,ALLOCATEBELOW=NO

indicates that the delay monitoring environment is to be located as 64-bit virtual storage.

,ALLOCATEBELOW=YES

indicates that the virtual storage is to be located below 2 gigabytes.

,ALLOCATEBELOW=NO
,ALLOCATEBELOW=YES

When MONTKN64_LIST=*montkn64_list* and REQTYPE=MULTIPLE are specified, an optional parameter, which indicates whether the virtual storage for the delay monitoring environment is to be obtained below 2 gigabytes. This is especially helpful for callers with 31-bit dependencies. The default is ALLOCATEBELOW=NO.

,ALLOCATEBELOW=NO

indicates that the delay monitoring environment is to be located as 64-bit virtual storage.

,ALLOCATEBELOW=YES

indicates that the virtual storage is to be located below 2 gigabytes.

,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, or specify a literal decimal value.

,BPMGMTONLY=NO

,BPMGMTONLY=YES
When REPORTONLY=NO is specified, an optional parameter, which indicates whether the monitoring environment is used together with enclaves to manage buffer pools (YES) or (NO). The default is BPMGMTONLY=NO.

,BPMGMTONLY=NO
indicates that the monitoring environment is not used together with enclaves to manage buffer pools.

,BPMGMTONLY=YES
indicates that the monitoring environment is used together with enclaves to manage buffer pools.

,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
When SUBSYSP=VALUE is specified, an optional parameter, which indicates if the created monitoring environment is intended to paricipate in Enterprise Workload Management (EWLM). The default is EWLM=NO.

,EWLM=NO
The monitoring environment can not 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, i.e. 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. 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*, 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, or specify a literal decimal value.

,LISTLEN=*listlen*

When REQTYPE=MULTIPLE is specified, a required input parameter which specifies the length (in bytes) of the area identified by the **MONTKN_LIST / MONTKN64_LIST** keyword.

Size of this area must be at least the size of one monitoring token (see **MONTKN / MONTKN64** 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, or specify a literal decimal value.

,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. A single MONTKN has a size of 4 byte.

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

Still supported, but use LISTLEN instead

To code: Specify the RS-type address, or address in register (2)-(12), of a fullword field, or specify a literal decimal value.

,MONTKNKEY=*montknkey*

When MONTKNKEYP=VALUE is specified, a required input parameter, which contains the key in which the delay monitoring environment will be invoked subsequently when using the output MONTKN / MONTKN64. 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 for the monitoring environment 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.

,MONTKN64=*montkn64*

When REQTYPE=SINGLE is specified, a required output parameter which will receive the long delay monitoring token.

To code: Specify the RS-type address, or address in register (2)-(12), of a 64 bit field.

,MONTKN64_LIST=*montkn64_list*

When REQTYPE=MULTIPLE is specified, a required input parameter which specifies an area into which a list of long delay monitoring tokens will be placed. A single MONTKN64 has a size of 8 byte.

To code: Specify the RS-type address, or address in register (2)-(12), of a character field.

,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:

BPMGMTONLY	GROUPNM
EWLM	GROUPNM_LEN

- **2**, which supports both the following parameters and those from version 0 and 1:

ALLOCATEBELOW	LISTLEN	MONTKN64_LIST
MONTKN64		

To code: Specify one of the following:
- IMPLIED_VERSION
- MAX
- A decimal value of 0, 1, or 2

,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. If you specify 15, GPR15, REG15, or R15 (within or without parentheses), the value will be left in GPR 15.

To code: Specify the RS-type address of a fullword field, or register (2)-(12) or (15), (GPR15), (REG15), or (R15).

,RSNCODE=*rsncode*
An optional output parameter into which the reason code is to be copied from GPR 0. If you specify 0, 00, GPR0, GPR00, REG0, REG00, or R0 (within or without parentheses), the value will be left in GPR 0.

To code: Specify the RS-type address of a fullword field, or register (0) or (2)-(12), (00), (GPR0), (GPR00), REG0), (REG00), or (R0).

,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 codes and reason codes

When the IWM4MCRE 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 76. Return and Reason Codes for the IWM4MCRE Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.
4	xxxx0409	**Equate Symbol**: IwmRsnCodeNoConn **Meaning**: 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. **Action**: None required.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0801	**Equate Symbol**: IwmRsnCodeSrbMode **Meaning**: Caller is in SRB mode. **Action**: Avoid requesting this function while in SRB mode.
8	xxxx0802	**Equate Symbol**: IwmRsnCodeXmemUserKeyTkn **Meaning**: Caller is in cross-memory mode while the token was requested in user key. **Action**: Avoid requesting this function while in cross-memory mode.
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: Caller is locked. **Action**: Avoid requesting this function while locked.
8	xxxx080B	**Equate Symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list. **Action**: Check for possible storage overlay.
8	xxxx0810	**Equate Symbol**: IwmRsnCodeEutFrr **Meaning**: Caller has EUT FRR established. **Action**: Avoid requesting this function with an EUT FRR set.
8	xxxx0821	**Equate Symbol**: IwmRsnCodeBadConn **Meaning**: Input connect token does not pass validity checking. **Action**: Check for possible storage overlay.

Table 76. Return and Reason Codes for the IWM4MCRE Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0823	**Equate Symbol**: IwmRsnCodeDatoff **Meaning**: Caller invoked service while DATOFF **Action**: Avoid requesting this function in this environment.
8	xxxx0824	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: Caller invoked service but was in 24 bit addressing mode. **Action**: Request this function only when you are in 31 bit addressing mode.
8	xxxx0825	**Equate Symbol**: IwmRsnCodeAscModeNotPrimary **Meaning**: Caller invoked service but was not DAT on Primary ASC mode. **Action**: Avoid requesting this function in this environment.
8	xxxx0826	**Equate Symbol**: IwmRsnCodeTaskTerm **Meaning**: Caller invoked service while task termination is in progress for the TCB associated with the owner. **Action**: Avoid requesting this function in this environment.
8	xxxx0827	**Equate Symbol**: IwmRsnCodeRsvdNot0 **Meaning**: Reserved field in parameter list was non-zero. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx0828	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: Version number in parameter list is not valid. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx0829	**Equate Symbol**: IwmRsnCodeBadOptions **Meaning**: Parameter list omits required parameters or supplies mutually exclusive parameters or provides data associated with options not selected. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx0844	**Equate Symbol**: IwmRsnCodeBadMonTknListLen **Meaning**: The storage area length specified on the MONTKN_LISTLEN parameter is not large enough to contain the data being returned. No data is returned. **Action**: Invoke the function with an output area sufficient to receive the data.
C	—	**Equate Symbol**: IwmRetCodeEnvError **Meaning**: Environmental error.

Table 76. Return and Reason Codes for the IWM4MCRE Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
C	xxxx0C01	**Equate Symbol**: IwmRsnCodeNoStg **Meaning**: Storage is not available for the request. **Action**: There is a storage shortage. The function may work successfully at a later time.
C	xxxx0C09	**Equate Symbol**: IwmRsnCodeNoResmgr **Meaning**: Resource manager could not be established. **Action**: No action required. This condition may be due to a storage shortage condition.
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error. **Action**: No action required. The function may be successful if invoked again.

Examples

None.

IWM4MDEL — Delete delay monitoring environment

The purpose of this service is to delete a delay monitoring environment, so that MVS state sampling will no longer monitor for new work requests to be associated with the input monitoring token.

The delete 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 delete macro is provided in PL/AS and assembler formats.

The parameter list must be in the caller's primary address space.

Note: This service was previously called IWMMDELE for 31-bit addressing only (see "IWMMDELE — Delete the monitoring environment" on page 932).

Environment

The requirements for the caller are:

Minimum authorization:	Supervisor state or program key mask (PKM) allowing keys 0-7.
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31- or 64-bit. If in 64-bit addressing mode, code `SYSSTATE AMODE64=YES` before invoking this macro.
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	Unlocked, but FRRs are allowed.
Control parameters:	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. If the key specified on IWM4MCRE 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 IWM4MCRE 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 IWM4MCRE was invoked.
3. The caller must serialize to prevent any delay monitoring services from being invoked concurrently or subsequently for the environment represented by the monitoring token
4. This service should not be invoked while in a RTM termination routine (resource manager) for the TCB 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.

5. This macro may only be used on z/OS V2R1 or higher.

Input register information

Before issuing the IWM4MDEL 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 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

```
►►──┬──────┬──b──IWM4MDEL──b──┬─MONTKN=montkn───────┬──┬──────────────────┬──────────►
    └─name─┘                  └─,MONTKN64=montkn64─┘  └─,RETCODE=retcode─┘
```

```
                                  ┌─,PLISTVER=IMPLIED_VERSION─┐
►──┬───────────────────┬──────────┼───────────────────────────┼──────────────────────────►
   └─,RSNCODE=rsncode──┘          ├─,PLISTVER=MAX─────────────┤
                                  └─,PLISTVER=0───────────────┘

   ┌─,MF=S─────────────────────────────────────┐
►──┼───────────────────────────────────────────┼─────────────────────────────────────────►◄
   │                           ┌─,0D───┐       │
   ├─,MF=(L─,list addr─────────┼───────┼─)─────┤
   │                           └─,attr─┘       │
   │                           ┌─,COMPLETE─┐   │
   └─,MF=(E─,list addr─────────┴───────────┴─)─┘
```

Parameters

The parameters are explained as follows:

name
An optional symbol, starting in column 1, that is the name on the IWM4MDEL 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.

MONTKN=*montkn*

A required input parameter which contains the delay monitoring token for the environment to be deleted.

To code: Specify the RS-type address, or address in register (2)-(12), of a 32 bit field.

,MONTKN64=*montkn64*

A required input parameter which contains the long delay monitoring token for the environment to be deleted.

To code: Specify the RS-type address, or address in register (2)-(12), of a 64 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. If you specify 15, GPR15, REG15, or R15 (within or without parentheses), the value will be left in GPR 15.

To code: Specify the RS-type address of a fullword field, or register (2)-(12) or (15), (GPR15), (REG15), or (R15).

,RSNCODE=*rsncode*

An optional output parameter into which the reason code is to be copied from GPR 0. If you specify 0, 00, GPR0, GPR00, REG0, REG00, or R0 (within or without parentheses), the value will be left in GPR 0.

To code: Specify the RS-type address of a fullword field, or register (0) or (2)-(12), (00), (GPR0), (GPR00), REG0), (REG00), or (R0).

ABEND codes

None.

Return codes and reason codes

When the IWM4MDEL 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. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 77. Return and Reason Codes for the IWM4MDEL Macro

Return Code	Reason Code	Meaning and Action
0	—	**Equate symbol:** IwmRetCodeOk **Meaning:** Successful completion.
4	—	**Equate symbol:** IwmRetCodeWarning **Meaning:** Successful completion, unusual conditions noted.
4	xxxx0402	**Equate symbol:** IwmRsnCodeNoMonEnv **Meaning:** Input monitoring token indicates no monitoring environment was established.
4	xxxx0403	**Equate symbol:** IwmRsnCodeMonEnvNotAlloc **Meaning:** Input monitoring token does not reflect an allocated monitoring environment owned by the current home address space.
8	—	**Equate symbol:** IwmRetCodeInvocError **Meaning:** Invalid invocation environment or parameters.
8	xxxx0802	**Equate symbol:** IwmRsnCodeXmemUserKeyTkn **Meaning:** Caller is in cross-memory mode while the token was obtained in user key.
8	xxxx0803	**Equate symbol:** IwmRsnCodeDisabled **Meaning:** Caller is disabled.
8	xxxx0804	**Equate symbol:** IwmRsnCodeLocked **Meaning:** Caller is locked.
8	xxxx0805	**Equate symbol:** IwmRsnCodeMonEnvSwitchCont **Meaning:** Input monitor token reflects a switch continuation.
8	xxxx0806	**Equate symbol:** IwmRsnCodeMonEnvParent **Meaning:** Input monitoring token reflects a continuation to a dependent monitoring environment.
8	xxxx0808	**Equate symbol:** IwmRsnCodeMonEnvDepCont **Meaning:** Input monitoring token reflects a continuation from a parent monitoring environment.

Table 77. Return and Reason Codes for the IWM4MDEL Macro (continued)

Return Code	Reason Code	Meaning and Action
8	xxxx0809	**Equate symbol:** IwmRsnCodeSrbUserKeyTkn **Meaning:** Caller is in SRB mode, while the token was obtained in a user key (8-F).
8	xxxx080A	**Equate symbol:** IwmRsnCodeTcbNotOwnerUserKeyTkn **Meaning:** Current TCB is not the current owner, while the token was obtained in a user key (8-F).
8	xxxx080B	**Equate symbol:** IwmRsnCodeBadPl **Meaning:** Error accessing parameter list.
8	xxxx0823	**Equate symbol:** IwmRsnCodeDatoff **Meaning:** Caller invoked service while DATOFF.
8	xxxx0824	**Equate symbol:** IwmRsnCodeAmode24 **Meaning:** Caller invoked service but was in 24-bit addressing mode.
8	xxxx0825	**Equate symbol:** IwmRsnCodeAscModeNotPrimary **Meaning:** Caller invoked service but was not DAT on Primary ASC mode.
8	xxxx0826	**Equate symbol:** IwmRsnCodeTaskTerm **Meaning:** Caller invoked service while task termination is in progress for the TCB associated with the owner.
8	xxxx0827	**Equate symbol:** IwmRsnCodeRsvdNot0 **Meaning:** Reserved field in parameter list was non-zero.
8	xxxx0828	**Equate symbol:** IwmRsnCodeBadVersion **Meaning:** Version number in parameter list is not valid.
8	xxxx082A	**Equate symbol:** IwmRsnCodeMonEnvRelated **Meaning:** Input monitor token is related to a parent monitoring environment.
10	—	**Equate symbol:** IwmRetCodeCompError **Meaning:** Component error.

Examples

None.

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:

Minimum authorization:	Supervisor state or program key mask (PKM) allowing keys 0-7.
Dispatchable unit mode:	Task
Cross memory mode:	PASN=HASN=SASN. The current HASN must match the HASN at the time that IWM4MREG was used to register the resource.
AMODE:	31- or 64-bit. If in 64-bit addressing mode, code `SYSSTATE AMODE64=YES` before invoking this macro.
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks may be held.
Control parameters:	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

1. NO FRRs may be established.
2. 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

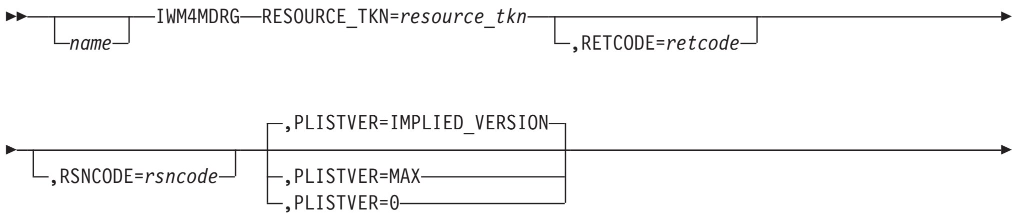
Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

Performance implications

None.

Syntax

The syntax of the IWM4MDRG macro is as follows:



```
   ┌─,MF=S──────────────────────────────┐
►──┼────────────────────────────────────┼──────────────────────────────────►◄
   │                        ┌─,0D──┐    │
   ├─,MF=(L─,list addr──────┴─,attr─┴─)─┤
   │                        ┌─,COMPLETE─┐│
   └─,MF=(E─,list addr──────┴───────────┴)┘
```

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 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 codes 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 78. Return and Reason Codes for the IWM4MDRG Macro

Return Code	Reason Code	Equate Symbol, Meaning, and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.
4	xxxx041D	**Equate Symbol**: IwmRsnCodeBadResTkn **Meaning**: The input resource token is not valid. The system may have deregistered due to errors associated with an exit. **Action**: Verify that the resource token passed has the intended value.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0801	**Equate Symbol**: IwmRsnCodeSRBMode **Meaning**: The caller is in SRB mode. **Action**: Avoid requesting this function in SRB mode.
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: The caller is locked. **Action**: Avoid requesting this function while locked.
8	xxxx080B	**Equate Symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list. **Action**: Check for possible storage overlay.
8	xxxx0810	**Equate Symbol**: IwmRsnCodeEutFrr **Meaning**: The caller has EUT FRR established. **Action**: Avoid requesting this function with an EUT FRR.
8	xxxx0823	**Equate Symbol**: IwmRsnCodeDatoff **Meaning**: The caller invoked the service while DATOFF **Action**: Avoid requesting this function in this environment.
8	xxxx0824	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: The caller invoked the service but was in 24-bit addressing mode. **Action**: Request this function only when you are in 31-bit or in 64-bit addressing mode.
8	xxxx0825	**Equate Symbol**: IwmRsnCodeAscModeNotPrimary **Meaning**: The caller invoked the service but was not DAT on Primary ASC mode. **Action**: Avoid requesting this function in this environment.

Table 78. Return and Reason Codes for the IWM4MDRG Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx0828	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: Version number or version length field in parameter list is not valid. **Action**: Check for possible overlay of the parameter list.
8	xxxx082F	**Equate Symbol**: IwmRsnCodeWrongHome **Meaning**: The caller invoked the service from the wrong home space. **Action**: Invoke the service from the owning address space.
8	xxxx0841	**Equate Symbol**: IwmRsnCodeXmemMode **Meaning**: The caller invoked the service but was in cross-memory mode. **Action**: Avoid requesting this function in cross-memory mode.
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error. **Action**: Contact your system programmer.

Example

To deregister a resource from monitoring, specify the following:

```
         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
```

IWM4MGDD — Define descriptions for generic delay states

Using this service, a subsystem can define descriptions for its generic delay states. The term *generic delay states* in this context is related to service IWM4MCHS. It means the case when `STATE=WAITING` is specified and the resource that is specified is `RESOURCE=TYPE`*x*, where *x* is a number between 1 and 15.

With this service, descriptions can be defined that might be more intuitive to a human reader than the generic terms. If defined, these descriptions will be accessible to performance monitors in the IWMWRCAA data area as a result of a call to the IWMRCOLL service.

Note, that for a subsystem that allows multiple instances (address spaces) to be active on the same z/OS system, only one set of descriptions can be in effect. If more that one instance of such subsystem defines a set of definitions, the last one defined will be effective.

For a subsystem that allows the customer to run multiple instances at possibly different release levels on the same system, it might be helpful if each instance first checks whether there are already some descriptions defined for that subsystem, before the instance decides whether to define its own descriptions.

Environment

The requirements for the caller are:

Minimum authorization:	Supervisor state or program key mask (PKM) allowing keys 0-7.
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31- or 64-bit. If in 64-bit addressing mode, code `SYSSTATE AMODE64=YES` before invoking this macro.
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks may be held.
Control parameters:	Control parameters must be in the primary address space or, for AR-mode callers, must be in an address/data space that is addressable through a public entry on the caller's dispatchable unit access list (DU-AL).

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

None.

Input register information

Before issuing the IWM4MGDD 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.

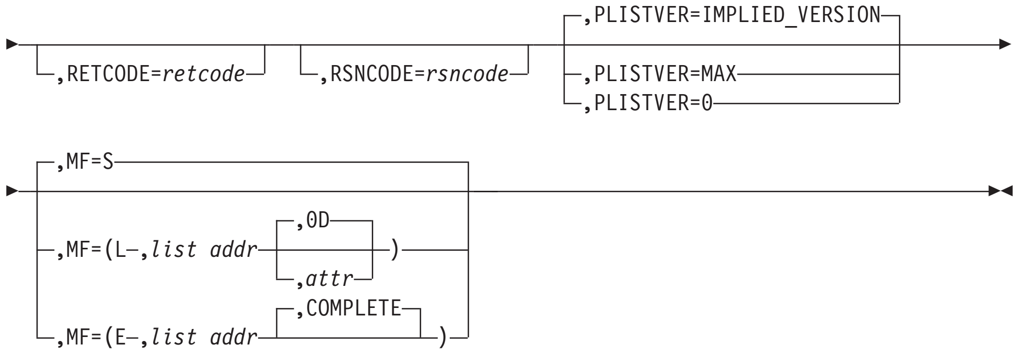
Performance implications

None.

Syntax

main diagram

```
►►──┬──────┬──b──IWM4MGDD──b──┬─REQTYPE=DEFINE───┬──,DESCRIPTIONS=descriptions───►
    └─name─┘                  └─REQTYPE=RETRIEVE─┘
```



Parameters

The parameters are explained as follows:

name

An optional symbol, starting in column 1, that is the name on the IWM4MGDD macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

,DESCRIPTIONS=*descriptions*

A required input parameter,

For REQTYPE=DEFINE, the address specifies the input area to the request that contains descriptions for generic delay states to be defined for a particular subsystem. The layout of the data area must adhere to the mapping defined by macro IWMWGDD.

Note: With one DEFINE request, a subsystem defines a set of descriptions for the generic delay states it uses. A subsequent DEFINE request will override the currently existing definitions. If IWMWGNUM is set to 0, then the currently existing definitions are deleted.

Following fields in the data area must be set correctly. For their meaning refer to the header of macro IWMWGDD directly.

- IWMWGEYE
- IWMWGVER
- IWMWGTYP
- IWMWGNUM
- IWMWGTNUM

Following field in the data area must not be set, i.e. it must have a value of zero. .br;For its meaning refer to the header of macro IWMWGDD directly.

- IWMWGNXT

If any of these fields is not set correctly, the request is terminated with return code IwmRetCodeInvocError and reason code IwmRsnCodeBadRequestList. The field in error is identified by setting IWMWGRC appropriately, refer to macro IWMWGDD for the possible values in IWMWGRC.

For REQTYPE=RETRIEVE, the address specifies the input/output area to the request. As input to the request the area contains the type of the subsystem for which the descriptions are to be retrieved. After execution of the request the area contains the descriptions for generic delay states as they are currently defined for the input subsystem. The layout of the data area must adhere to the mapping defined by macro IWMWGDD.

Following fields in the data area must be set correctly. For their meaning refer to the header of macro IWMWGDD directly.

- IWMWGEYE
- IWMWGVER
- IWMWGTYP
- IWMWGNUM

Following field in the data area must not be set

- IWMWGNXT

If any of these fields is not set correctly, the request is terminated with return code IwmRetCodeInvocError and reason code IwmRsnCodeBadRequestList. The field in error is identified by setting IWMWGRC appropriately, refer to macro IWMWGDD for the possible values in IWMWGRC.

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*,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

REQTYPE=DEFINE
REQTYPE=RETRIEVE

A required parameter that indicates whether the request is to define or to retrieve generic delay state descriptions.

REQTYPE=DEFINE

The request is to define the generic delay state descriptions for a particular subsystem.

REQTYPE=RETRIEVE

The request is to retrieve the generic delay state descriptions of a particular subsystem.

,RETCODE=*retcode*

An optional output parameter into which the return code is to be copied from GPR 15. If you specify 15, GPR15, REG15, or R15 (within or without parentheses), the value will be left in GPR 15.

To code: Specify the RS-type address of a fullword field, or register (2)-(12) or (15), (GPR15), (REG15), or (R15).

,RSNCODE=*rsncode*

An optional output parameter into which the reason code is to be copied from GPR 0. If you specify 0, 00, GPR0, GPR00, REG0, REG00, or R0 (within or without parentheses), the value will be left in GPR 0.

To code: Specify the RS-type address of a fullword field, or register (0) or (2)-(12), (00), (GPR0), (GPR00), REG0), (REG00), or (R0).

ABEND codes

None.

Return codes and reason codes

When the IWM4MGDD 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 79. Return and Reason Codes for the IWM4MGDD Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: Caller is locked. **Action**: Avoid requesting this function while locked.
8	xxxx080B	**Equate Symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list. **Action**: Check for possible storage overlay.
8	xxxx0823	**Equate Symbol**: IwmRsnCodeDatoff **Meaning**: Caller invoked service while DATOFF **Action**: Avoid requesting this function in this environment.
8	xxxx0824	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: Caller invoked service but was in 24 bit addressing mode. **Action**: Request this function only when you are in 31 bit addressing mode.
8	xxxx0828	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: Version number in parameter list is not valid. **Action**: Check for possible storage overlay of the parameter list.

Table 79. Return and Reason Codes for the IWM4MGDD Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0829	**Equate Symbol**: IwmRsnCodeBadOptions **Meaning**: Parameter list omits required parameters or supplies mutually exclusive parameters or provides data associated with options not selected. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx0888	**Equate Symbol**: IwmRsnCodeBadRequestList **Meaning**: The data area mapped by IWMWGDD does not pass verification. **Action**: Check the return and reason codes in IWMWGRC.
C	—	**Equate Symbol**: IwmRetCodeEnvError **Meaning**: Environmental error.
C	xxxx0C01	**Equate Symbol**: IwmRsnCodeNoStg **Meaning**: Storage is not available for the request. **Action**: There is a storage shortage. The function may work successfully at a later time.
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error. **Action**: No action required. The function may be successful if invoked again.

Example

None.

IWM4MINI — Monitoring environment initialization

IWM4MINI allows the caller to supply MVS with some or all of the work request attributes needed for the monitoring environment. The attributes include user ID, transaction name, transaction class, source LU, and LU 6.2 token.

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 environments provide only buffer pool performance management for enclaves.

Use the `REPORTONLY=YES` parameter to specify the monitoring environment will be used for reporting purposes only.

If you invoke IWM4MINI with the `REPORTONLY=YES` parameter, you must specify `ASSOCIATE=ENCLAVE` or `ASSOCIATE=ADDRESS_SPACE` to associate the monitoring environment with an enclave or an address space.

Use the `BPMGMTONLY=YES` parameter to specify the monitoring environment will be used for buffer pool management for enclaves only.

If you invoke IWM4MINI with the `BPMGMTONLY=YES` parameter, you must specify `ASSOCIATE=ENCLAVE` to associate the monitoring environment with an enclave.

For management monitoring environments, where possible, you should 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 for a management monitoring environment, 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.

Optionally with this macro, you can use the **`FROM_SUBSYSNM`** parameter to specify the subsystem name from where a request came in. This allows you to identify an address space as work provider or consumer.

Note: This service was previously called IWMMINIT for 31-bit addressing only (see "IWMMINIT — Initialize monitoring environment" on page 943).

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.
Dispatchable unit mode:	Task or SRB
Cross memory mode:	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.
AMODE:	31- or 64-bit. If in 64-bit addressing mode, code `SYSSTATE AMODE64=YES` before invoking this macro.
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	Locked or unlocked
Control parameters:	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 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.
4. For ASSEMBLER programmers an Assembler that supports the new z/Architecture instructions and the relative Branch (Jxx) instructions (such as High-Level Assembler Release 4 or higher) is required.

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. This macro may only be used on z/OS V1R13 or later levels for **`FROM_SUBSYSNM`** keyword.
7. This macro may only be used on z/OS V2R1 or later levels for the MONTKN64 keyword.

8. This macro supports multiple versions. Some keywords are unique to certain versions. See the **PLISTVER** parameter description.

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

```
                                   ┌─EWLM=NO─┐
►►──┬──────┬──b──IWM4MINI──b───────┴─────────┴──┬─,MONTKN=montkn─────┬──────────────────────────►
    └─name─┘                                    └─,MONTKN64=montkn64─┘

►──┬─,MODE=RESET──┤ parameters-1 ├──────────────────────────────────────────────────────┬────────►
   │              ┌─,DURATION=PREV_VALUE───┐                                            │
   └─,MODE=RETAIN─┼────────────────────────┼──┬─,DISPTYPE=SAVEDTYPE───────────────────┬─┘
                  ├─,DURATION=EXECUTION────┤  │                 ┌─,TCB=NO_TCB─┐       │
                  └─,DURATION=BEGIN_TO_END─┘  ├─,DISPTYPE=TCB───┴─,TCB=tcb────┴───────┤
                                              └─,DISPTYPE=SRB─────────────────────────┘
```

```
                                           ┌─,OWNER_TOKEN=NO_OWNER_TOKEN─┐
►──┬─,CONTINUEP=YES──┬─,FROM=NONE─────┬──┬─┴─────────────────────────────┴──────────────►
   │                 ├─,FROM=LOCALMVS─┤  │  └─,OWNER_TOKEN=owner_token───┘
   │                 ├─,FROM=SYSPLEX──┤  │
   │                 └─,FROM=NETWORK──┘  │
   └─,CONTINUEP=NO───────────────────────┘

   ┌─,OWNER_DATA=NO_OWNER_DATA─┐   ┌─,FROM_SUBSYSNM=NO_SUBSYSNM───┐
►──┴───────────────────────────┴───┴──────────────────────────────┴────────────────────►
   └─,OWNER_DATA=owner_data────┘   └─,FROM_SUBSYSNM=from_subsysnm─┘

     ┌─,REPORTONLY=NO─┐
►──┬─┴────────────────┴──┤ parameters-2 ├──────────────────────────────┬───────────────►
   └─,REPORTONLY=YES──┬─,ASSOCIATE=ENCLAVE─,ENCLAVETOKEN=enclavetoken─┤
                      └─,ASSOCIATE=ADDRESS_SPACE─,ASID=asid───────────┘

                   ┌─,TRXNAME=NO_TRXNAME─┐   ┌─,USERID=NO_USERID─┐
►──┬─,SCOPE=SHARED─┬─┴─────────────────────┴───┴───────────────────┴───────────────────►
   └─,SCOPE=SINGLE─┘ └─,TRXNAME=trxname────┘   └─,USERID=userid────┘

   ┌─,TRXCLASS=NO_TRXCLASS─┐   ┌─,TTRACETOKEN=NO_TTRACETOKEN─┐
►──┴───────────────────────┴───┴─────────────────────────────┴─────────────────────────►
   └─,TRXCLASS=trxclass────┘   └─,TTRACETOKEN=ttracetoken────┘

   ┌─,SOURCELU=NO_SOURCELU─┐
►──┴───────────────────────┴───────────────────────────────────────────────────────────►
   └─,SOURCELU=sourcelu────┘

►──┬────────────────────────────────────────────────────────────────────────┬─────────►
   └─┬─,LU62TKN=NO_LU62TKN─┬─┬─,LU62TKN_FMT=LU_NO_CC_27───────────────────┬─┘
     └─,LU62TKN=lu62tkn────┘ ├─,LU62TKN_FMT=FULL_LU_NO_CC_27──────────────┤
                             ├─,LU62TKN_FMT=FULL_LU_0_CC_28───────────────┤
                             ├─,LU62TKN_FMT=FULL_LU_CC_36─────────────────┤
                             └─,LU62TKN_FMT=OTHER─,LU62TKN_LEN=lu62tkn_len┘

                                              ┌─,PLISTVER=IMPLIED_VERSION─┐
►──┬──────────────────┬──┬──────────────────┬─┼───────────────────────────┼───────────►
   └─,RETCODE=retcode─┘  └─,RSNCODE=rsncode─┘ ├─,PLISTVER=MAX─────────────┤
                                              ├─,PLISTVER=0───────────────┤
                                              ├─,PLISTVER=1───────────────┤
                                              └─,PLISTVER=2───────────────┘

                          ┌─,COMPLETE─┐
►─,MF=(M─,list addr─┴───────────┴─)────────────────────────────────────────────────►◄
                          └─,NOCHECK──┘
```

parameters-1

```
     ┌─,DURATION=EXECUTION────┐                   ┌─,TCB=NO_TCB─┐
►►───┴────────────────────────┴──┬─,DISPTYPE=TCB──┴─────────────┴──┬──────────────────►
     └─,DURATION=BEGIN_TO_END─┘  │                └─,TCB=tcb────┘  │
                                 └─,DISPTYPE=SRB───────────────────┘

►──┬─,ARRIVALTIMEP=CURRENT─────────────────────────┬───────────────────────────────────►
   └─,ARRIVALTIMEP=YES─,ARRIVALTIME=arrivaltime────┘

   ┌─,EWLM_PACORR=NO_EWLM_PACORR─┐  ┌─,EWLM_PACTKN=NO_EWLM_PACTKN─┐
►──┴─────────────────────────────┴──┴─────────────────────────────┴───────────────────►◄
   └─,EWLM_PACORR=ewlm_pacorr────┘  └─,EWLM_PACTKN=ewlm_pactkn────┘
```

parameters-2

```
    ┌─,BPMGMTONLY=NO─┐                                               ┌─,SERVCLS=NO_SERVCLS─┐
►►──┬────────────────┴──┬─,EXSTARTTIMEP=NO────────────────────────┬──┴─┬───────────────────┬─┴──┬───────►◄
    │                   ├─,EXSTARTTIMEP=CURRENT───────────────────┤    └─,SERVCLS=servcls──┘    │
    │                   └─,EXSTARTTIMEP=YES─,EXSTARTTIME=exstarttime┘                           │
    └─,BPMGMTONLY=YES──,ASSOCIATE=ENCLAVE─,ENCLAVETOKEN=enclavetoken────────────────────────────┘
```

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. This keyword is not applicable for report-only or bufferpool-only monitoring environments.

  ,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
: 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.

,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.

,BPMGMTONLY=NO
,BPMGMTONLY=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 IWM4MNTF 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, i.e. the execution phase. Typically a monitoring environment with this duration would be passed to IWM4MNTF 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.

,ENCLAVETOKEN=*enclavetoken*
When ASSOCIATE=ENCLAVE, BPMGMTONLY=YES and REPORTONLY=NO 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.

,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=NO

An optional parameter, which indicates if this work manager intents 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.

,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 80 on page 604 for further information. If the corrrelator is a valid ARM correlator, but cannot be understood by EWLM (no EWLM format), it is silently ignored.
- Parameters EWLM_PACORR and EWLM_PACTKN are mutually exclusive.

The default is NO_EWLM_PACORR. 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 (see Return Codes section). The default is NO_EWLM_PACTKN. 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) will preserve the existing execution start time, if any.

,EXSTARTTIMEP=CURRENT

indicates that the current time should be supplied by the service.

,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.

,FROM_SUBSYSNM=*from_subsysnm*
,FROM_SUBSYSNM=NO_SUBSYSNM

An optional input parameter, which contains the subsystem name from where the request came in. The default is NO_SUBSYSNM which indicates that no FROM_SUBSYSNM is provided.

If MODE(RETAIN) is specified, NO_SUBSYSNM will preserve the existing FROM_SUBSYSNM, if any.

To code: Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

,LU62TKN=*lu62tkn*
,LU62TKN=NO_LU62TKN

An optional input parameter, which contains LU 6.2 token for the work request. This is not a 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 a 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.

,MONTKN64=*montkn64*
A required input parameter which contains the long delay monitoring token.

To code: Specify the RS-type address, or address in register (2)-(12), of a 64 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/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 will preserve 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
,PLISTVER=2
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

- **2**, which supports both the following parameters and those from version 0 and 1:

FROM_SUBSYSNM	MONTKN64

To code: Specify one of the following:
- IMPLIED_VERSION
- MAX
- A decimal value of 0, 1, or 2

,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. If you specify 15, GPR15, REG15, or R15 (within or without parentheses), the value will be left in GPR 15.

To code: Specify the RS-type address of a fullword field, or register (2)-(12) or (15), (GPR15), (REG15), or (R15).

,RSNCODE=*rsncode*
An optional output parameter into which the reason code is to be copied from GPR 0. If you specify 0, 00, GPR0, GPR00, REG0, REG00, or R0 (within or without parentheses), the value will be left in GPR 0.

To code: Specify the RS-type address of a fullword field, or register (0) or (2)-(12), (00), (GPR0), (GPR00), REG0), (REG00), or (R0).

,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. indicates that no service class token was passed.

If MODE(RETAIN) is specified, NO_SERVCLS will preserve 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, i.e. 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 will determine 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 will copy the first 17 bytes.

In the PL/AS form, the rules for the PL/AS compiler will determine when to pad with blanks, i.e. 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. indicates that no source LU name was passed.

If MODE(RETAIN) is specified, NO_SOURCELU will preserve 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.

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.

,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. indicates that no transaction trace token was passed.

If MODE(RETAIN) is specified, NO_TTRACETOKEN will preserve 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 will preserve 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 codes and reason codes

When the IWM4MINI 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 80. Return and Reason Codes for the IWM4MINI Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	**Equate Symbol:** IwmRetCodeOk **Meaning:** Successful completion.
4	—	**Equate Symbol:** IwmRetCodeWarning **Meaning:** Successful completion, unusual conditions noted.
4	xxxx0402	**Equate Symbol**: IwmRsnCodeNoMonEnv **Meaning**: Monitoring token indicates that no monitoring environment exists.
8	—	**Equate Symbol:** IwmRetCodeInvocError: Invalid invocation environment or parameters
8	xxxx081E	**Equate Symbol**: IwmRsnCodeBadLU62TknLen **Meaning**: The length byte of the LU62 token has an invalid value. Only values 1-36 (decimal) are valid.
8	xxxx0820	**Equate Symbol**: IwmRsnCodeBadMonEnv: **Meaning**: Monitoring environment does not pass short form verification.
8	xxxx0894	**Equate Symbol**: IwmRsnCodeInvalidEWLMCorr **Meaning**: 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 gretater than 512 ('0200'x). **Action**: Check the specification of the correlator information.

Table 80. Return and Reason Codes for the IWM4MINI Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0895	**Equate Symbol**: IwmRsnCodeEWLMServNotEnabled **Meaning**: Service is not enabled because the Monitoring Environment was not created with EWLM=YES (either on IWM4CON or IWM4MCRE). **Action**: Specify the parameter EWLM_PACORR or EWLM_PACTKN only when the Monitoring Environment was created with EWLM=YES (either on IWM4CON or IWM4MCRE).
C	—	**Equate Symbol:** IwmRetCodeEnvError **Meaning:** Environmental error.
C	xxxx0C07	**Equate Symbol**: IwmRsnCodeNoArrTime: **Meaning**: No arrival time was supplied to the service and STCK gave a non-zero condition code.
C	xxxx0C08	**Equate Symbol**: IwmRsnCodeNoExTime: **Meaning**: No execution start time was supplied to the service and STCK 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
```

IWM4MNTF — Notify of work execution completion

The primary purpose of this service is to notify MVS that the execution phase for a work request that is associated with a monitoring environment just completed. Processor consumption data can also be provided. The execution phase might represent the entire work request or a subset of it. .

An indication is also given regarding whether the monitoring environment should be disassociated from the work request. When `DISASSOCIATE(YES)` is specified, this service renders the information that is associated with the monitoring environment unpredictable. To associate a work request with the monitoring environment after use of `DISASSOCIATE(YES)`, first use `Initialize Mode(Reset)` or Relate/Transfer

Note: This service was previously called IWMMNTFY for 31 bit addressing only (see "IWMMNTFY — Notify of work execution completion" on page 958).

Environment

The requirements for the caller are:

Minimum authorization:	Supervisor state. PSW key must either be 0 or match the value that is supplied on IWM4CON when a connect token is passed. PSW key must either be 0 or match the value that is supplied on IWM4MCRE. PSW key must be 0-7. See "Restrictions" on page 709.
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31- or 64-bit. If in 64-bit addressing mode, code `SYSSTATE AMODE64=YES` before invoking this macro.
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	LOCAL lock is held
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. The high-order halfword of register 0, and the reason code variable when specified, might be nonzero and represents diagnostic data that 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, can be used for this purpose.

Restrictions

1. Caller is responsible for error recovery
2. Though the caller is required to be enabled, this condition is not checked. Violation of this restriction might cause disabled program checks, which would be the responsibility of the caller's recovery to handle.

3. The monitoring environment must contain the information that is saved by IWM4MINI, not IWM4MRLT
4. The current PSW key must be 0 or match the key that is specified on IWM4MCRE provided the latter is a system key (0-7).
5. IWM4MNTF 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 IWM4MNTF, 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 that is represented by the monitoring token.
9. This macro can be used only on z/OS V2R1 or later.

Input register information

Before issuing the IWM4MNTF macro, the caller must ensure that the following general-purpose registers (GPRs) contain the specified information:

Register	Contents
13	The address of a 216 byte standard save area in the primary address space

Before issuing the IWM4MNTF macro, the caller does not have to place any information into any AR unless the caller is 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
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system.

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

```
►►──┬──────┬──b──IWM4MNTF──b──┬─MONTKN=montkn─────┬──┬─,COMPLETION=YES─┬──────────►
    └─name─┘                  └─MONTKN64=montkn64─┘  └─,COMPLETION=NO──┘

                                ┌─,WORKREQ_STA=IWMEWLMARMSTATUSNONE─┐     ┌─,EWLM=NO─┐
►──┬─,DISASSOCIATE=YES──────────┼───────────────────────────────────┼──┬──┴──────────┴──►
   │                            └─,WORKREQ_STA=workreq_sta──────────┘  │
   └─,DISASSOCIATE=NO──────────────────────────────────────────────────┘

   ┌─,CONNTKN=NO_CONNTKN─┐
►──┼─────────────────────┼────────────────────────────────────────────────────────►
   └─,CONNTKN=conntkn────┘

►──┬────────────────────────────────────────────────────────────────┬─────────────►
   └─,CPUTIME=cputime──,TIMEONCP=timeoncp──,OFFLOADONCP=offloadoncp─┘

   ┌─,ENDTIME=CURRENT─┐
►──┼──────────────────┼──┬──────────────────┬──┬──────────────────┬───────────────►
   └─,ENDTIME=endtime─┘  └─,RETCODE=retcode─┘  └─,RSNCODE=rsncode─┘

   ┌─,PLISTVER=IMPLIED_VERSION─┐  ┌─,MF=S──────────────────────────────────┐
►──┼───────────────────────────┼──┼────────────────────────────────────────┼──────►◄
   ├─,PLISTVER=MAX─────────────┤  │                         ┌─,0D──┐       │
   ├─,PLISTVER=0───────────────┤  ├─,MF=(L─,list addr───────┼──────┼─)─────┤
   └─,PLISTVER=1───────────────┘  │                         └─,attr┘       │
                                  │                         ┌─,COMPLETE─┐  │
                                  └─,MF=(E─,list addr───────┴───────────┴─)┘
```

Parameters

The parameters are explained as follows:

name
: An optional symbol, starting in column 1, that is the name on the IWM4MNTF 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 major execution phase or phases are now complete.

,COMPLETION=YES
: Indicates that execution for an entire phase of processing is now complete. Typically IWM4MNTF Completion(Yes) would be issued as a result of the

completion of the transaction program for the work request. When a work request is composed of several (typically cascaded) transaction programs, each would correspond to an invocation of IWM4MNTF Completion(Yes).

The execution time, as given by the difference between the IWM4MNTF ENDTIME value and the Execution start time (established by the use of IWM4MINI), is added to the running total execution time for the service class. There might still be "output" processing that is left to perform for the work request, which time would be accounted for through the use of Iwmrpt. There might 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 that is 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 IWM4MNTF ENDTIME value and the Execution start time (established through the use of IWM4MINI), is 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, which 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.

,CPUTIME=*cputime*
An optional input parameter that contains the total CPU time, in STCK format, for the current work request.

To code: Specify the RS-type address, or address in register (2)-(12), of a 64-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, which 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=NO
An optional parameter, which indicates whether 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.

,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 can 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 that is 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 that contains the delay monitoring token.

To code: Specify the RS-type address, or address in register (2)-(12), of a 32-bit field.

,MONTKN64=*montkn64*
A required input parameter that contains the long delay monitoring token.

To code: Specify the RS-type address, or address in register (2)-(12), of a 64-bit field.

,OFFLOADONCP=*offloadoncp*
When CPUTIME is specified, a required input parameter that contains the CPU time on standard CP that was offload eligible, in STCK format, for the current work request.

To code: Specify the RS-type address, or address in register (2)-(12), of a 64-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 that are 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 that are 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 parameters that are referenced in the description of higher versions.
- **1**, which supports the parameters that are supported by 0, and CPUTIME, TIMEONCP, and OFFLOADONCP.

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. If you specify 15, GPR15, REG15, or R15 (within or without parentheses), the value is left in GPR 15.

To code: Specify the RS-type address of a fullword field, or register (2)-(12) or (15), (GPR15), (REG15), or (R15).

,RSNCODE=*rsncode*
An optional output parameter into which the reason code is to be copied from GPR 0. If you specify 0, 00, GPR0, GPR00, REG0, REG00, or R0 (within or without parentheses), the value is left in GPR 0.

To code: Specify the RS-type address of a fullword field, or register (0) or (2)-(12), (00), (GPR0), (GPR00), REG0), (REG00), or (R0).

,TIMEONCP=*timeoncp*
When CPUTIME is specified, a required input parameter that contains the CPU time on standard CP, in STCK request format, for the current work.

| **To code:** Specify the RS-type address, or address in register (2)-(12), of a 64-bit
| field.

,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, which 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 through the use of the IWM4MABN service, the completion status IwmEwlmArmStatusGood is reported to EWLM. If an abnormal event was reported through the use of IWM4MABN or COMPLETION=NO was specified, the completion status IwmEwlmArmStatusFailed is reported to EWLM.

To code: Specify the RS-type address, or address in register (2)-(12), of a fullword field, or specify a literal decimal value.

ABEND codes

None.

Return codes and reason codes

When the IWM4MNTF 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 that is associated with each reason code. IBM support personnel might request the entire reason code, including the **xxxx** value.

Table 81. Return and Reason Codes for the IWM4MNTF Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.

Table 81. Return and Reason Codes for the IWM4MNTF Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
4	xxxx0405	**Equate Symbol**: IwmRsnCodeGoalNoMonEnv **Meaning**: System is in goal mode but the input monitoring token indicates that no monitoring environment was established, hence MVS did not receive the information. **Action**: None required.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx080C	**Equate Symbol**: IwmRsnCodeMonEnvLacksData **Meaning**: Input monitoring environment does not contain the necessary information. **Action**: Ensure that the monitoring environment was established with the necessary information.
8	xxxx080F	**Equate Symbol**: IwmRsnCodeNoUserKeyNtfy **Meaning**: User key routine is not allowed to issue Notify. **Action**: Avoid requesting this function in user key.
8	xxxx0820	**Equate Symbol**: IwmRsnCodeBadMonEnv **Meaning**: Input monitoring environment does not pass short form validity checking. **Action**: Check for possible storage overlay.
8	xxxx0821	**Equate Symbol**: IwmRsnCodeBadConn **Meaning**: Input connect token does not pass validity checking. **Action**: Check for possible storage overlay.
8	xxxx082D	**Equate Symbol**: IwmRsnCodeExStTimeGTEndTime **Meaning**: Input execution start time later than end time. **Action**: Check for possible storage overlay of the parameter list or variable.
8	xxxx0840	**Equate Symbol**: IwmRsnCodeServiceNotEnabled **Meaning**: Caller's space connection is not enabled for this service. **Action**: Avoid requesting this function under the input connection. IWM4CON options must be specified previously to enable this service.
8	xxxx087E	**Equate Symbol**: IwmRsnCodeRoMonEnv **Meaning**: Input monitoring environment is Report-Only. **Action**: Avoid requesting this function for Report-Only PBs.

Table 81. Return and Reason Codes for the IWM4MNTF Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0895	**Equate Symbol**: IwmRsnCodeEWLMServNotEnabled **Meaning**: Service is not enabled because monitoring environment cannot be associated with EWLM work requests. **Action**: Specify the parameter WORKREQ_STA only when the monitoring environment is created with IWM4MCRE EWLM=YES or the address space is connected with IWMCONN EWLM=YES and the connect token is passed to IWM4MCRE when creating the monitoring environment.
8	xxxx0897	**Equate Symbol**: IwmRsnCodeTranStatusInvalid **Meaning**: Passed work request completion status is not valid. **Action**: Check the EWLM ARM interface specification for valid completion status values.
C	—	**Equate Symbol**: IwmRetCodeEnvError **Meaning**: Environmental error.
C	xxxx0C04	**Equate Symbol**: IwmRsnCodeNtfyNoWorkElt **Meaning**: Notify routine invoked, but no work element was available to save the input information. **Action**: Invoke the function when the conditions are alleviated. This condition may be due to a common storage shortage condition.
C	xxxx0C06	**Equate Symbol**: IwmRsnCodeNoEndTime **Meaning**: No end time was supplied to the service and STCK gave a nonzero condition code. **Action**: No action is required.

Example

None.

IWM4MREG — Register a resource for monitoring

The IWM4MREG service registers 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:

Minimum authorization:	Supervisor state or program key mask (PKM) allowing keys 0-7. The caller's PSW key must be in the range 0-7.
Dispatchable unit mode:	Task
Cross-memory mode:	PASN=HASN=SASN
AMODE:	31- or 64-bit. If in 64-bit addressing mode, code `SYSSTATE AMODE64=YES` before invoking this macro.
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks may be held.
Control parameters:	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

The syntax of the IWM4MREG macro is as follows:

main diagram

```
►►──┬──────┬──IWM4MREG──RESOURCE_NAME=resource_name──────────────────────────►
    └─name─┘

►─,RESOURCE_TYPE=BUFFER_POOL─┤ parameters-1 ├─,RESOURCE_TKN=resource_tkn─────────►
```

```
                                                  ┌─,PLISTVER=IMPLIED_VERSION─┐
►──┬──────────────────┬──┬──────────────────┬──┼───────────────────────────┼──────────────►
   └─,RETCODE=retcode─┘  └─,RSNCODE=rsncode─┘  ├─,PLISTVER=MAX─────────────┤
                                                  └─,PLISTVER=0───────────────┘

   ┌─,MF=S──────────────────────────────┐
►──┼─────────────────────────────────────┼──────────────────────────────────────────────►◄
   │                           ┌─,0D───┐  │
   ├─,MF=(L─,list addr─┼────────┼─)───┤
   │                           └─,attr─┘  │
   │                           ┌─,COMPLETE─┐│
   └─,MF=(E─,list addr─┴──────────┴─)┘
```

parameters-1

```
►►──,RES_ADJ_SIZE=res_adj_size──,RES_MIN_SIZE=res_min_size──────────────────►

►─,RES_MAX_SIZE=res_max_size──,RES_ADJ_EXIT@=res_adj_exit@─────────────────►

►─,RES_DATA_EXIT@=res_data_exit@──,OWNER_TKN=owner_tkn─────────────────────►◄
```

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 used, for example, by the change state service (IWMMCHST) and the deregister a resource service (IWM4MDRG).

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 codes 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 82. Return and Reason Codes for the IWM4MREG Macro

Return Code	Reason Code	Equate Symbol, Meaning, and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0801	**Equate Symbol**: IwmRsnCodeSRBMode **Meaning**: The caller is in SRB mode. **Action**: Avoid requesting this function in SRB mode.
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: Caller is locked. **Action**: Avoid requesting this function while locked.
8	xxxx080B	**Equate Symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list. **Action**: Check for possible storage overlay.
8	xxxx0810	**Equate Symbol**: IwmRsnCodeEutFrr **Meaning**: The caller has EUT FRR established. **Action**: Avoid requesting this function with an EUT FRR.
8	xxxx0823	**Equate Symbol**: IwmRsnCodeDatoff **Meaning**: The caller invoked the service while DATOFF **Action**: Avoid requesting this function in this environment.

Table 82. Return and Reason Codes for the IWM4MREG Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx0824	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: The caller invoked the service but was in 24-bit addressing mode. **Action**: Request this function only when you are in 31-bit addressing mode.
8	xxxx0825	**Equate Symbol**: IwmRsnCodeAscModeNotPrimary **Meaning**: The caller invoked the service but was not DAT on Primary ASC mode. **Action**: Avoid requesting this function in this environment.
8	xxxx0828	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: Version number or version length field in parameter list is not valid. **Action**: Check for possible overlay of the parameter list.
8	xxxx0841	**Equate Symbol**: IwmRsnCodeXmemMode **Meaning**: The caller invoked the service but was in cross-memory mode. **Action**: Avoid requesting this function in cross-memory mode.
8	xxxx0846	**Equate Symbol**: IwmRsnCodeNoUserKeyReg **Meaning**: The caller invoked the service but was in user key. **Action**: Request this function in system key (0-7).
8	xxxx08A1	**Equate Symbol**: IwmRsnCodeBadBPMinMaxSize **Meaning**: Maximum resource size is lower than the minimum size. **Action**: Specify a maximum size value at least as high as the minimum size value.
C	xxxx0C01	**Equate Symbol**: IwmRetCodeEnvError: **Meaning**: Environmental error. **Action**: Storage is not available for the request.
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error. **Action**: Contact your system programmer.

Example

To register a resource for delay monitoring, specify the following:

```
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
*                              can be adjusted
RESMINSIZE DS  FL4             contains the minimum size
*                              associated with the specified
*                              resource
RESMAXSIZE 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
```

IWM4MRLT — Relate monitoring environments (PBs)

The calling subsystem work manager can use IWM4MRLT to relate two different monitoring environments that are associated with the same work request. IWM4MRLT 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 IWM4MRLT when you do not have direct access to the information required by IWM4MINI. If the caller has the monitoring token for a parent environment that is previously established for the same work request, you provide it in the **PARENTMONTKN** or **PARENTMONTKN64** parameter. If the caller does not pass the monitoring token, you can use PARENTP=FINDACTIVE to specify that the parent monitoring environment is the active monitoring environment owned by the address space and which is associated with the TCB provided via **PARENTTCB**.

IWM4MRLT must be used together with IWM4MXFR 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 can use the token/data to keep your own information.

Note: This service was previously called IWMMRELA for 31-bit addressing only (see "IWMMRELA — Relate monitoring environment service" on page 967).

Environment

The requirements for the caller are:

Minimum authorization:	Problem state or supervisor state. PSW key must either be 0 or match the value supplied on IWM4MCRE.
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31- or 64-bit. If in 64-bit addressing mode, code SYSSTATE AMODE64=YES before invoking this macro.
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	Unlocked when PARENT=FINDACTIVE is specified, otherwise, no restrictions.
Control parameters:	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** / **MONTKN64** 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** / **MONTKN64**). You could do this 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.
  - You cannot specify the list form of this macro. With `PARENTP=YES`, IWM4MRLT 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

This macro may only be used on z/OS V2R1 or later.

Input register information

Before issuing the IWM4MRLT 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 following general purpose registers (GPRs) have to contain the specified information:

Register	Contents
13	The address of a 216-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

```
►►──┬──────┬──b──IWM4MRLT──b──┬─FUNCTION=CREATE─┤ parameters-1 ├─┬──────────────────►
    └─name─┘                  └─FUNCTION=DELETE──────────────────┘


                             ┌─,EWLM=NO─┐
►──┬─,MONTKN=montkn─────┬────┴──────────┴────┬────────────────────┬─────────────────►
   └─,MONTKN64=montkn64─┘                    └─,RETCODE=retcode───┘

                             ┌─,PLISTVER=IMPLIED_VERSION─┐
►──┬─────────────────────┬───┼───────────────────────────┼──────────────────────────►
   └─,RSNCODE=rsncode────┘   ├─,PLISTVER=MAX─────────────┤
                             └─,PLISTVER=0───────────────┘

   ┌─,MF=S────────────────────────────────────┐
►──┼──────────────────────────────────────────┼─────────────────────────────────────►◄
   │                         ┌─,0D───┐        │
   ├─,MF=(L─,list addr───────┼───────┼─)──────┤
   │                         └─,attr─┘        │
   │                         ┌─,COMPLETE─┐    │
   └─,MF=(E─,list addr───────┴───────────┴─)──┘
```

parameters-1

```
     ┌─,OWNER_TOKEN=NO_OWNER_TOKEN─┐  ┌─,OWNER_DATA=NO_OWNER_DATA─┐
►►───┼─────────────────────────────┼──┼───────────────────────────┼─────────────────►
     └─,OWNER_TOKEN=owner_token────┘  └─,OWNER_DATA=owner_data────┘

►──┬─,DISPTYPE=TCB─,TCB=tcb───────────┬─────────────────────────────────────────────►
   └─,DISPTYPE=SRB──┬─,SAMEDU=YES─────┤
                    └─,SAMEDU=NO──────┘

►──┬─,PARENTP=YES──┬─,PARENTMONTKN=parentmontkn─────┬──┬─,PARENTENV=NOSWITCH──┬──┬──────►◄
   │               └─,PARENTMONTKN64=parentmontkn64─┘  ├─,PARENTENV=SECONDARY─┤  │
   │                                                   └─,PARENTENV=HOME──────┘  │
   └─,PARENTP=FINDACTIVE─,PARENTTCB=parenttcb────────────────────────────────────┘
```

Parameters

The parameters are explained as follows:

name
An optional symbol, starting in column 1, that is the name on the IWM4MRLT 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** / **MONTKN64**).

,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.

,EWLM=NO
An optional parameter, which indicates if this work manager intents 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.

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. Thus the **MF** keyword is not applicable when this option is specified, and is not allowed. 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.

,MONTKN64=*montkn64*

A required input parameter which contains the long delay monitoring token for the dependent environment.

To code: Specify the RS-type address, or address in register (2)-(12), of a 64 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/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/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, i.e. 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.

,PARENTMONTKN64=*parentmontkn64*

When PARENTP=YES and FUNCTION=CREATE are specified, a required input parameter which contains the long delay monitoring token for the parent environment, i.e. 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 64 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. Thus the **MF** keyword is not applicable when this option is specified, and is not allowed. 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. If you specify 15, GPR15, REG15, or R15 (within or without parentheses), the value will be left in GPR 15.

To code: Specify the RS-type address of a fullword field, or register (2)-(12) or (15), (GPR15), (REG15), or (R15).

,RSNCODE=*rsncode*
An optional output parameter into which the reason code is to be copied from GPR 0. If you specify 0, 00, GPR0, GPR00, REG0, REG00, or R0 (within or without parentheses), the value will be left in GPR 0.

To code: Specify the RS-type address of a fullword field, or register (0) or (2)-(12), (00), (GPR0), (GPR00), REG0), (REG00), or (R0).

,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** / **MONTKN64** 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, also known as 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.

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).

To code: Specify the RS-type address, or address in register (2)-(12), of a character field.

ABEND codes

None.

Return codes and reason codes

When the IWM4MRLT 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 83. Return and Reason Codes for the IWM4MRLT Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion.
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.

Table 83. Return and Reason Codes for the IWM4MRLT Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
4	*xxxx*0402	**Equate Symbol**: IwmRsncodeNoMonEnv **Meaning**: Input monitoring token indicates no monitoring environmen was established.
4	*xxxx*0406	**Equate Symbol**: IwmRsncodeNoParEnv **Meaning**: No parent monitoring environment was established. The input dependent monitoring environment is now related to the Home address space.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters
8	*xxxx*0802	**Equate Symbol**: IwmRsnCodeXmemUserKeyTkn **Meaning**: Caller is in cross-memory mode while the token was obtained in user key.
8	*xxxx*0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled
8	*xxxx*0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: Caller is locked
8	*xxxx*080B	**Equate Symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list.
8	*xxxx*081A	**Equate Symbol**: IwmRsnCodeCallerNotAuthDepEnv **Meaning**: Caller is not authorized to update the dependent monitoring environment
8	*xxxx*0820	**Equate Symbol**: IwmRsnCodeBadMonEnv **Meaning**: Monitoring environment does not pass verification.
8	*xxxx*0822	**Equate Symbol**: IwmRsnCodeBadParEnv **Meaning**: Parent monitoring environment does not pass verification.
8	*xxxx*0823	**Equate Symbol**: IwmRsnCodeDatoff **Meaning**: Caller invoked service while DATOFF
8	*xxxx*0824	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: Caller invoked service but was was in 24 bit addressing mode.
8	*xxxx*0825	**Equate Symbol**: IwmRsnCodeAscModeNotPrimary **Meaning**: Caller invoked service but was not DAT on Primary ASC mod
8	*xxxx*0827	**Equate Symbol**: IwmRsnCodeRsvdNot0 **Meaning**: Reserved field in parameter list was non-zero
8	*xxxx*0828	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: Version number in parameter list is not valid.
8	*xxxx*0829	**Equate Symbol**: IwmRsnCodeBadOptions **Meaning**: Parameter list omits required parameters or supplies mutually exclusive parameters or provides data associated with options not selected.

Table 83. Return and Reason Codes for the IWM4MRLT Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	*xxxx*087E	**Equate Symbol**: IwmRsnCodeRoMonEnv **Meaning**: Monitoring environment is report only
8	*xxxx*087F	**Equate Symbol**: IwmRsnCodeRoParEnv **Meaning**: Parent monitoring environment is report only
8	*xxxx*08A4	**Equate Symbol**: IwmRsnCodeBPParEnv **Meaning**: Parent monitoring environment is buffer pool management only
10	—	**Equate Symbol**: IwmRetCodeCompError: **Meaning**: Component error

Example

To relate two monitoring environments where an address space switch is not required, specify:

```
IWM4MRLT FUNCTION=CREATE,MONTKN64=(R7),PARENTP=YES,
      PARENTMONTKN64=(R8),PARENTENV=NOSWITCH,
      DISPTYPE=SRB,SAMEDU=YES,
      RETCODE=RCODE,RSNCODE=RSN
```

IWM4MSTO — Stops a work unit

The purpose of this service is to stop a work unit which has been started by IWM4MSTR. A work unit started by IWM4MINI is not affected by this service. The work unit is unblocked, if it is blocked at the time you issue this macro.

Note: This service was previously called IWMMSTOP for 31-bit addressing only (see “IWMMSTOP — Stop a work unit” on page 976).

Environment

The requirements for the caller are:

Minimum authorization:	Supervisor state or program key mask (PKM) allowing keys 0-7.
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31- or 64-bit. If in 64-bit addressing mode, code `SYSSTATE AMODE64=YES` before invoking this macro.
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No restriction.
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 bits 0-31 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 prior to the completion of WLM address space initialization
2. Caller must have issued the IWM4MSTR macro successfully.
3. Caller is responsible for error recovery
4. The current PSW key must be 0 or match the key specified on IWM4MCRE provided the latter is a system key (0-7).
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 V2R1 or later.

Input register information

Before issuing the IWM4MSTO 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 following general purpose registers (GPRs) have to contain the specified information:

Register	Contents
13	The address of a 216-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 the return code in GPR 15 is not 0, otherwise, used as a work register by the system. The reason code is stored in bits 0-31
1	Used as work register by the system
2-13	Unchanged
14	Used as work register by the system
15	Return code is stored in bits 0-31

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.

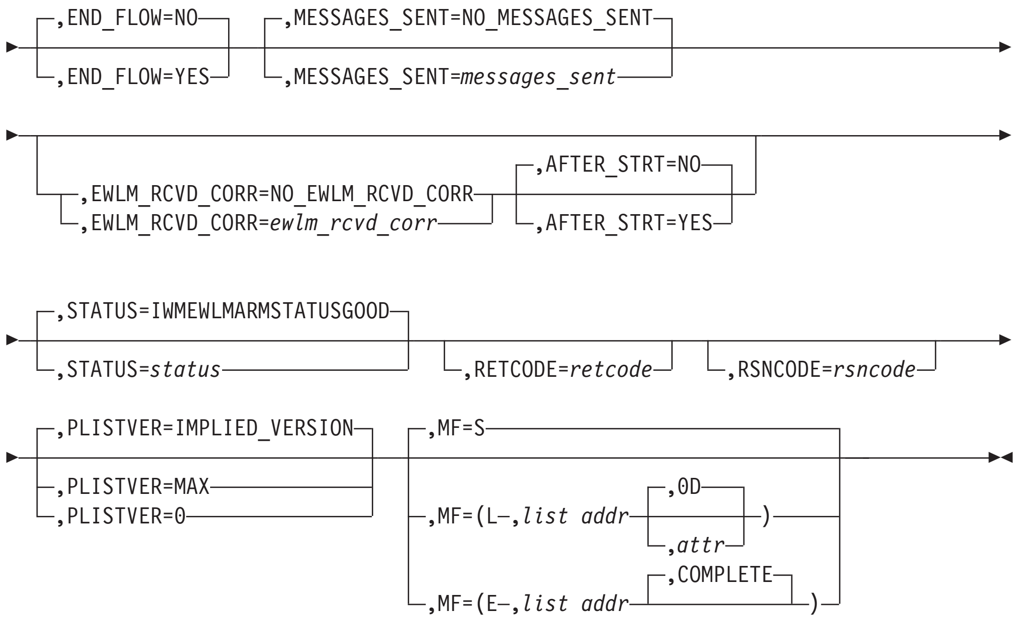
Performance implications

None.

Syntax

main diagram

```
                                                    ┌─,EWLM=NO─┐
►►──┬──────┬──b──IWM4MSTO──b──┬─MONTKN=montkn─────┬─┴──────────┴──────────────►
    └─name─┘                  └─MONTKN64=montkn64─┘
```



Parameters

The parameters are explained as follows:

name

An optional symbol, starting in column 1, that is the name on the IWM4MSTO macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

,AFTER_STRT=NO
,AFTER_STRT=YES

When EWLM_RCVD_CORR=*ewlm_rcvd_corr* is specified, an optional parameter, which indicates the moment the correlator has been received. The default is AFTER_STRT=NO.

,AFTER_STRT=NO

indicates that the correlator has been received before this work unit has been started by IWM4MSTR.

,AFTER_STRT=YES

indicates that the correlator has arrived within the scope of this work unit that means after issuing IWM4MSTR.

,END_FLOW=NO
,END_FLOW=YES

An optional parameter, which indicates the completion of a message flow. The default is END_FLOW=NO.

,END_FLOW=NO

indicates that a message flow has not completed.

,END_FLOW=YES

indicates that a message flow has completed. Specify END_FLOW=YES, if you know that the running work unit is the last one in a work unit flow. This indication can not be cleared, if it has been set.

,EWLM=NO
An optional parameter, which indicates if this work manager intents 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.

,EWLM_RCVD_CORR=*ewlm_rcvd_corr*
,EWLM_RCVD_CORR=NO_EWLM_RCVD_CORR
An optional input parameter, which contains a cross platform Enterprise Workload Management (EWLM) correlator received from another application. Workflows often have multiple parent work units that must complete before a new work unit can be initiated. You can pass only 1 parent correlator to the IWM4MSTR macro and one additional parent correlator to the IWM4MSTO macro. You have to issue the IWM4MUPD macro, if more than two parent correlators should be assigned to a work unit. This correlator is ignored, if it is an unknown EWLM correlator. The default is NO_EWLM_RCVD_CORR. indicates that parameter EWLM_RCVD_CORR has not been specified.

To code: Specify the RS-type address, or address in register (2)-(12), of a character field.

,MESSAGES_SENT=*messages_sent*
,MESSAGES_SENT=NO_MESSAGES_SENT
An optional input parameter, which contains the number of messages sent to other applications. This value is added to the total messages_sent value of the work unit. The total messages_sent value should not exceed 32767. The default is NO_MESSAGES_SENT. indicates that parameter MESSAGES_SEND has not been specified.

To code: Specify the RS-type address, or address in register (2)-(12), of a fullword field, or specify a literal decimal value.

,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.

To code: Specify the RS-type address, or address in register (2)-(12), of a 32 bit field.

,MONTKN64=*montkn64*
A required input parameter which contains the long delay monitoring token.

To code: Specify the RS-type address, or address in register (2)-(12), of a 64 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. If you specify 15, GPR15, REG15, or R15 (within or without parentheses), the value will be left in GPR 15.

To code: Specify the RS-type address of a fullword field, or register (2)-(12) or (15), (GPR15), (REG15), or (R15).

,RSNCODE=*rsncode*
An optional output parameter into which the reason code is to be copied from GPR 0. If you specify 0, 00, GPR0, GPR00, REG0, REG00, or R0 (within or without parentheses), the value will be left in GPR 0.

To code: Specify the RS-type address of a fullword field, or register (0) or (2)-(12), (00), (GPR0), (GPR00), REG0), (REG00), or (R0).

,STATUS=*status*
,STATUS=IWMEWLMARMSTATUSGOOD
An optional input parameter, which contains the completion status code of the work unit. 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 IWMEWLMARMSTATUSGOOD. indicates that the work unit completed successfully.

To code: Specify the RS-type address, or address in register (2)-(12), of a fullword field, or specify a literal decimal value.

ABEND codes

None.

Return codes and reason codes

When the IWM4MSTO 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 84. Return and Reason Codes for the IWM4MSTO Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.
4	xxxx0402	**Equate Symbol**: IwmRsnCodeNoMonEnv **Meaning**: The input monitoring token indicates no monitoring environment was established. **Action**: Establish a monitoring environment by macro IWM4MCRE.

Table 84. Return and Reason Codes for the IWM4MSTO Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
4	xxxx0441	**Equate Symbol**: IwmRsnCodeTooManyMsgCorrs **Meaning**: The correlator passed to **EWLM_RCVD_CORR** is ignored, since the maximum number of supported correlators has been reached. **Action**: None required.
4	xxxx0443	**Equate Symbol**: IwmRsnCodeTooManyMsgsSent **Meaning**: The value passed to MESSAGES_SENT is ignored, since the maximum number of messages sent is reached. **Action**: None required.
4	xxxx0444	**Equate Symbol**: IwmRsnCodeTooManyMsgsReceived **Meaning**: The **EWLM_RCVD_CORR** parameter has been specified too often. The correlated counter is not increased. **Action**: None required.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0820	**Equate Symbol**: IwmRsnCodeBadMonEnv **Meaning**: Input monitoring environment does not pass short form validity checking. **Action**: Check for possible storage overlay.
8	xxxx0897	**Equate Symbol**: IwmRsnCodeTranStatusInvalid **Meaning**: An unsupported value has been passed to the STATUS parameter. **Action**: Specify a supported value.
8	xxxx08AC	**Equate Symbol**: IwmRsnCodeTranNotStarted **Meaning**: No work unit has been started by IWM4MSTR for the specified monitoring environment. **Action**: Start a work unit by IWM4MSTR macro, before issuing this macro.

Example

None.

IWM4MSTR — Indicate the start of a work unit

The purpose of this service is to indicate that a work unit is beginning execution. The work unit runs under the specified monitoring environment, but is reported to EWLM completely independent from a potentially running transaction on the same monitoring environment that is defined by IWM4MINI and IWM4RPT/IWM4MNTF calls. You can use the set of services IWM4MSTR, IWM4MSTO, IWM4MUPD to provide data for "mini work units" running within a long-running transaction. In addition a work unit started by IWM4MSTR can participate in asynchronous and synchronous work unit flows.

Note: This service was previously called IWMMSTRT for 31-bit addressing only (see "IWMMSTRT — Indicate the start of a work unit" on page 983).

Environment

The requirements for the caller are:

Minimum authorization:	Supervisor state or program key mask (PKM) allowing keys 0-7.
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31- or 64-bit. If in 64-bit addressing mode, code `SYSSTATE AMODE64=YES` before invoking this macro.
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No restriction.
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 bits 0-31 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 prior to the completion of WLM address space initialization
2. The caller must have issued the IWM4MCRE macro successfully and the created delay monitoring environment must be enabled for EWLM support. This means the delay monitoring environment must be created by one of the following ways:
   - `IWM4CON EWLM=YES ...` and `IWM4MCRE SUBSYSP=CONNECT ...`
   - `IWM4MCRE SUBSYSP=VALUE EWLM=YES ...`
3. The caller is responsible for error recovery.

4. The current PSW key must be 0 or match the key specified on IWM4MCRE provided the latter is a system key (0-7).
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 V2R1 or later.

Input register information

Before issuing the IWM4MSTR 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 following general purpose registers (GPRs) have to contain the specified information:

Register	Contents
13	The address of a 216-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 the return code in GPR 15 is not 0, otherwise, used as a work register by the system. The reason code is stored in bits 0-31
1	Used as work register by the system
2-13	Unchanged
14	Used as work register by the system
15	Return Code stored in bits 0-31

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

```
                                                    ┌─,END_FLOW=NO──┐
►►──┬──────┬──b──IWM4MSTR──b──┬─MONTKN=montkn─────┬──┼───────────────┼───────────────►
    └─name─┘                  └─MONTKN64=montkn64─┘  └─,END_FLOW=YES─┘

  ┌─,MSG_RECEIVED=NO──┐  ┌─,EWLM=NO─┐  ┌─,ARRIVALTIME=NO_ARRIVALTIME─┐
►─┼───────────────────┼──┴──────────┴──┼─────────────────────────────┼──────────────►
  └─,MSG_RECEIVED=YES─┘                └─,ARRIVALTIME=arrivaltime────┘

►──┬─,EWLM_S_PACORR=ewlm_s_pacorr─────────────────────────────────────────────┬─────►
   │                                  ┌─,EWLM_RCVD_CORR=NO_EWLM_RCVD_CORR─┐    │
   └─,EWLM_S_CURCORR=ewlm_s_curcorr───┴─,EWLM_RCVD_CORR=ewlm_rcvd_corr────┴────┘

                                                 ┌─,PLISTVER=IMPLIED_VERSION─┐
►──┬────────────────────┬──┬────────────────────┬─┼───────────────────────────┼────►
   └─,RETCODE=retcode───┘  └─,RSNCODE=rsncode───┘ ├─,PLISTVER=MAX─────────────┤
                                                  └─,PLISTVER=0───────────────┘

  ┌─,MF=S──────────────────────────────┐
►─┼────────────────────────────────────┼───────────────────────────────────────────►◄
  │                      ┌─,0D───┐      │
  ├─,MF=(L─,list addr────┼───────┼─)───┤
  │                      └─,attr─┘      │
  │                      ┌─,COMPLETE─┐  │
  └─,MF=(E─,list addr────┴───────────┴─)┘
```

Parameters

The parameters are explained as follows:

name
  An optional symbol, starting in column 1, that is the name on the IWM4MSTR 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 a timestamp in STCK format. This timestamp is subtracted from the current timestamp and assigned as queued time to the work unit. For example, you may use this parameter, if the work unit is started by receipt of a message from a queue and you know the put time (the timestamp when the message has been put onto the queue). The default is NO_ARRIVALTIME. indicates that parameter ARRIVALTIME has not been specified.

  To code: Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

,END_FLOW=NO
,END_FLOW=YES
  An optional parameter, which indicates the completion of a message flow. The default is END_FLOW=NO.

  ,END_FLOW=NO

    indicates that a message flow has not completed.

  ,END_FLOW=YES

indicates that a message flow has completed. Specify END_FLOW=YES, if you know that this work unit is the last one in a work unit flow. This indication can not be cleared, if it has been set.

,EWLM=NO

An optional parameter, which indicates if this work manager intents 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.

,EWLM_RCVD_CORR=*ewlm_rcvd_corr*
,EWLM_RCVD_CORR=NO_EWLM_RCVD_CORR

When EWLM_S_CURCORR=*ewlm_s_curcorr* is specified, an optional input parameter, which contains a cross platform Enterprise Workload Management (EWLM) correlator received from another application. Normally this is a received correlator which has the independent flag and the asynchronous flag set. It should not be passed to the EWLM_S_PACORR or the EWLM_S_CURCORR parameter. If you pass this correlator to one of them then the started work unit is not reclassified and runs under the classification of this correlator. When you receive a correlator with the independent flag set then you should:

1. Reclassify the work unit by issuing IWM4CLFY EWLM_CORR=r_corr EWLM_CHCORR=c_corr. r_corr is the received correlator and c_corr is the correlator created by IWM4CLFY.
2. Start the work unit by issuing IWM4MSTR EWLM_S_CURCORR=c_corr EWLM_RCVD_CORR=r_corr .

The default is NO_EWLM_RCVD_CORR. indicates that parameter EWLM_RCVD_CORR has not been specified.

To code: Specify the RS-type address, or address in register (2)-(12), of a character field.

,EWLM_S_CURCORR=*ewlm_s_curcorr*

A required input parameter which contains a cross platform Enterprise Workload Management (EWLM) correlator for the current application. The correlator passed to this parameter is used as the current correlator of this work unit. It has usually been created by means of a previous IWM4CLFY call with the EWLM_CHCORR parameter (see below).

To code: Specify the RS-type address, or address in register (2)-(12), of a character field.

,EWLM_S_PACORR=*ewlm_s_pacorr*

A required input parameter which contains a cross platform Enterprise Workload Management (EWLM) parent correlator received from another application.

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***,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.

To code: Specify the RS-type address, or address in register (2)-(12), of a 32 bit field.

,MONTKN64=*montkn64*
A required input parameter which contains the long delay monitoring token.

To code: Specify the RS-type address, or address in register (2)-(12), of a 64 bit field.

,MSG_RECEIVED=NO
,MSG_RECEIVED=YES
An optional parameter, which indicates whether this work unit has been started as a result of a receipt of a message. The default is MSG_RECEIVED=NO.

,MSG_RECEIVED=NO

indicates that this work unit has not been started by receipt of a message.

,MSG_RECEIVED=YES

indicates that this work unit has been started as a result of a receipt of a message.

,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. If you specify 15, GPR15, REG15, or R15 (within or without parentheses), the value will be left in GPR 15.

To code: Specify the RS-type address of a fullword field, or register (2)-(12) or (15), (GPR15), (REG15), or (R15).

,RSNCODE=*rsncode*

An optional output parameter into which the reason code is to be copied from GPR 0. If you specify 0, 00, GPR0, GPR00, REG0, REG00, or R0 (within or without parentheses), the value will be left in GPR 0.

To code: Specify the RS-type address of a fullword field, or register (0) or (2)-(12), (00), (GPR0), (GPR00), REG0), (REG00), or (R0).

ABEND codes

None.

Return codes and reason codes

When the IWM4MSTR 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 85. Return and Reason Codes for the IWM4MSTR Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.
4	xxxx0402	**Equate Symbol**: IwmRsnCodeNoMonEnv **Meaning**: The input monitoring token indicates no monitoring environment was established. **Action**: Establish a monitoring environment by macro IWM4MCRE.
4	xxxx0442	**Equate Symbol**: IwmRsnCodeCorrelatorUnknown **Meaning**: A unknown correlator has been passed to the EWLM_RCVD_CORR parameter. It is ignored. **Action**: None required.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0820	**Equate Symbol**: IwmRsnCodeBadMonEnv **Meaning**: Input monitoring environment does not pass short form validity checking. **Action**: Check for possible storage overlay.
8	xxxx0894	**Equate Symbol**: IwmRsnCodeInvalidEwlmCorr **Meaning**: An unknown EWLM correlator has been passed to the EWLM_S_PACORR or EWLM_S_CURCORR parameter. **Action**: Specify a supported correlator. You can create a supported correlator by macro IWM4CLFY.
8	xxxx0895	**Equate Symbol**: IwmRsnCodeEWLMServNotEnabled **Meaning**: The EWLM service was not enabled for a delay monitoring environment. **Action**: Create a monitoring environment with EWLM=YES (either on macro IWM4CON or macro IWM4MCRE).
8	xxxx08AD	**Equate Symbol**: IwmRsnCodeAlreadyActive **Meaning**: A work unit started by IWM4MSTR is already active. **Action**: Stop the active work unit by macro IWM4MSTO before creating a new one.
8	xxxx08AF	**Equate Symbol**: IwmRsnCodeArrTimeGTStartTime **Meaning**: The arrivaltime passed is greater than the current timestamp. **Action**: Check the format of the passed arrivaltime.

Example

None.

IWM4MSWC — Monitoring environment switch

The purpose of this service is to reflect that the delay information for a work request may now also reside in another monitoring environment which is not Related to the current environment (Continue) OR that there is no further information for the current work request beyond the current environment (Return).

The scope of this service is restricted to the input monitoring environment and no other monitoring environments are accessed or otherwise involved.

It is preferable to use IWM4MXFR, where the necessary information is available, and the restrictions can be met, since this gives more specific information to MVS about the status of the work request.

Environment

The requirements for the caller are:

Minimum authorization:	Problem state. PSW key must either be 0 or match the value supplied on IWM4MCRE.
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31- or 64-bit. If in 64-bit addressing mode, code `SYSSTATE AMODE64=YES` before invoking this macro.
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	Suspend locks are allowed, as are FRRs
Control parameters:	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.
3. Caller is responsible for error recovery
4. Only limited checking is done against the input monitoring token.
5. 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.
6. FUNCTION(CONTINUE) may not be used when there is an outstanding continuation established by use of Transfer Continue
7. The caller must serialize to prevent any delay monitoring services from being invoked concurrently for the input monitoring environment.
8. This macro may only be used on z/OS V2R1 or later.

Input register information

Before issuing the IWM4MSWC 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 following general purpose registers (GPRs) have to contain the specified information:

Register	**Contents**
13	The address of a 216-byte standard save area.

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.

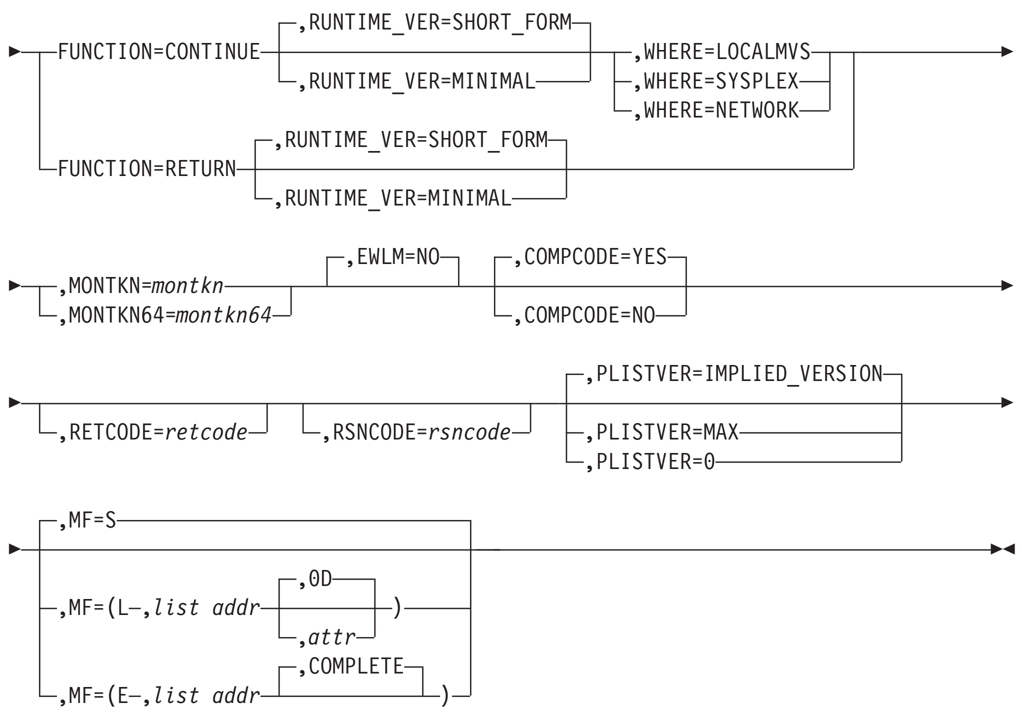
Performance implications

None.

Syntax

main diagram

```
►►──┬──────┬──b──IWM4MSWC──b──────────────────────────────────────────►
    └─name─┘
```



Parameters

The parameters are explained as follows:

name
An optional symbol, starting in column 1, that is the name on the IWM4MSWC 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.

,EWLM=NO
An optional parameter, which indicates if this work manager intents 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.

FUNCTION=CONTINUE
FUNCTION=RETURN
A required parameter, which indicates where there may be one or more other

monitoring environments which represent current information about the work request. This is meant to cover further continuations of the work request, and does not deal with any parent environment that may exist.

FUNCTION=CONTINUE
indicates that the current environment is creating only a single continuation elsewhere.

FUNCTION=RETURN
indicates that any continuations of the work request have completed. These continuations may have been established through use of Transfer or Switch.

,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 current environment.

To code: Specify the RS-type address, or address in register (2)-(12), of a 32 bit field.

,MONTKN64=*montkn64*
A required input parameter which contains the long delay monitoring token for the current environment.

To code: Specify the RS-type address, or address in register (2)-(12), of a 64 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. If you specify 15, GPR15, REG15, or R15 (within or without parentheses), the value will be left in GPR 15.

To code: Specify the RS-type address of a fullword field, or register (2)-(12) or (15), (GPR15), (REG15), or (R15).

,RSNCODE=*rsncode*

An optional output parameter into which the reason code is to be copied from GPR 0. If you specify 0, 00, GPR0, GPR00, REG0, REG00, or R0 (within or without parentheses), the value will be left in GPR 0.

To code: Specify the RS-type address of a fullword field, or register (0) or (2)-(12), (00), (GPR0), (GPR00), REG0), (REG00), or (R0).

,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.

,WHERE=LOCALMVS
,WHERE=SYSPLEX
,WHERE=NETWORK
When FUNCTION=CONTINUE is specified, a required parameter, which indicates where there may be another monitoring environment

,WHERE=LOCALMVS
indicates that such an environment may exist on the current MVS.

,WHERE=SYSPLEX
indicates that such an environment may exist in the current syplex, but is not expected to be on the current MVS image.

,WHERE=NETWORK
indicates that such an environment may exist, but is not expected to be in the current sysplex.

ABEND codes

None.

Return codes and reason codes

When the IWM4MSWC 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. IBM support personnel may request the entire reason code, including the **xxxx** value.

Table 86. Return and Reason Codes for the IWM4MSWC Macro

Return Code	Reason Code	Meaning and Action
0	—	**Equate symbol:** IwmRetCodeOk **Meaning:** Successful completion.
4	—	**Equate symbol:** IwmRetCodeWarning **Meaning:** Successful completion, unusual conditions noted.

Table 86. Return and Reason Codes for the IWM4MSWC Macro (continued)

Return Code	Reason Code	Meaning and Action
4	xxxx0402	**Equate symbol:** IwmRsncodeNoMonEnv **Meaning:** Input monitoring token indicates no monitoring environment was established.
4	xxxx0407	**Equate symbol:** IwmRsncodeReturnCont **Meaning:** Switch Return was from a monitoring environment with an outstanding continuation.
8	—	**Equate symbol:** IwmRetCodeInvocError **Meaning:** Invalid invocation environment or parameters.
8	xxxx081C	**Equate symbol:** IwmRsnCodeContExists **Meaning:** Outstanding continuation exists.
8	xxxx0820	**Equate symbol:** IwmRsnCodeBadMonEnv **Meaning:** Monitoring environment does not pass short form verification.
10	—	**Equate symbol:** IwmRetCodeCompError **Meaning:** Component Error

Example

None.

IWM4MUPD — Update data for a work unit

The purpose of this service is to update data about a work unit which has been started by IWM4MSTR. A work unit started by IWM4MINI is not affected by this service.

Note: This service was previously called IWMMUPD for 31-bit addressing only (see "IWMMUPD — Update data for a work unit" on page 995).

Environment

The requirements for the caller are:

Minimum authorization:	Supervisor state or program key mask (PKM) allowing keys 0-7.
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31- or 64-bit. If in 64-bit addressing mode, code `SYSSTATE AMODE64=YES` before invoking this macro.
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks may be held. No restriction.
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 bits 0-31 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 prior to the completion of WLM address space initialization
2. Caller must have issued the IWM4MSTR macro successfully
3. Caller is responsible for error recovery
4. The current PSW key must be 0 or match the key specified on IWM4MCRE provided the latter is a system key (0-7)
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 V2R1 or later.

Input register information

Before issuing the IWM4MUPD 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 following general purpose registers (GPRs) have to contain the specified information:

Register	Contents
13	The address of a 216-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 the return code in GPR 15 is not 0, otherwise, used as a work register by the system. The reason code is stored in bits 0-31
1	Used as work register by the system
2-13	Unchanged
14	Used as work register by the system
15	Return code stored in bits 0-31

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.

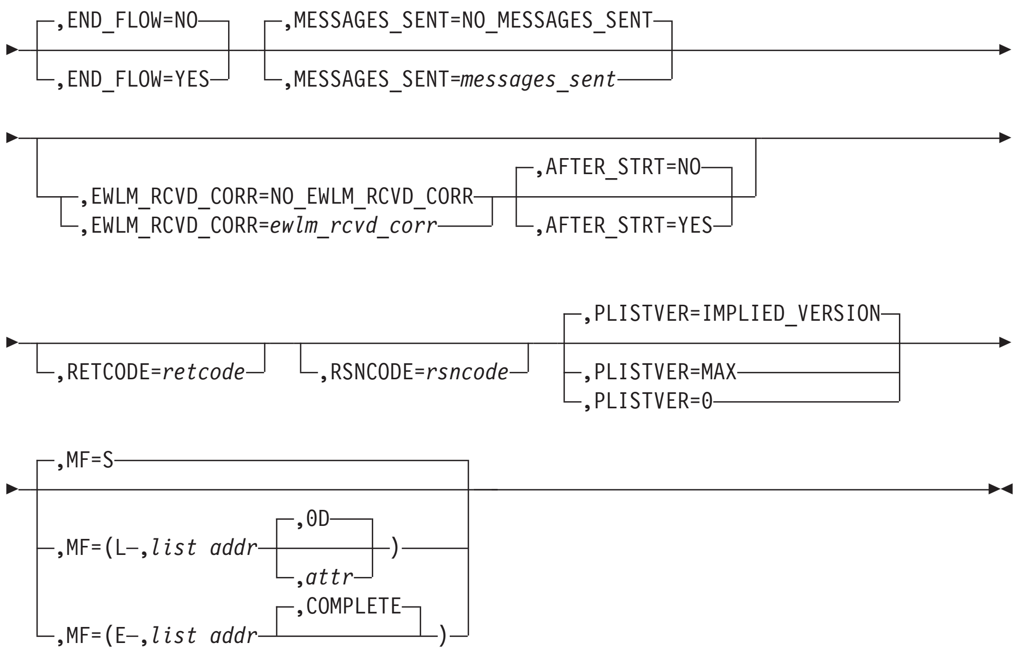
Performance implications

None.

Syntax

main diagram

```
                                                    ┌─,EWLM=NO─┐
►►─┬──────┬─b─IWM4MUPD─b─┬─MONTKN=montkn─────┬─┴──────────┴──────────────►
   └─name─┘              └─MONTKN64=montkn64─┘
```



Parameters

The parameters are explained as follows:

name
An optional symbol, starting in column 1, that is the name on the IWM4MUPD macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

,AFTER_STRT=NO
,AFTER_STRT=YES
When EWLM_RCVD_CORR=*ewlm_rcvd_corr* is specified, an optional parameter, which indicates the moment the correlator has been received. The default is AFTER_STRT=NO.

,AFTER_STRT=NO

indicates that the correlator has been received before this work unit has been started by IWM4MSTR.

,AFTER_STRT=YES

indicates that the correlator has arrived within the scope of this work unit that means after issuing IWM4MSTR.

,END_FLOW=NO
,END_FLOW=YES
An optional parameter, which indicates the completion of a message flow. The default is END_FLOW=NO.

,END_FLOW=NO

indicates that a message flow has not completed.

,END_FLOW=YES

indicates that a message flow has completed. Specify END_FLOW=YES, if you know that the running work unit is the last one in a work unit flow. This indication can not be cleared, if it has been set.

,EWLM=NO
An optional parameter, which indicates if this work manager intents 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.

,EWLM_RCVD_CORR=*ewlm_rcvd_corr*
,EWLM_RCVD_CORR=NO_EWLM_RCVD_CORR
An optional input parameter, which contains a cross platform Enterprise Workload Management (EWLM) correlator received from another application. Workflows often have multiple parent work units that must complete before a new work unit can be initiated. You can pass only 1 parent correlator to the IWM4MSTR macro and one additional parent correlator to the IWM4MSTO macro. You have to issue this macro, if more than two parent correlators should be assigned to a work unit. This correlator is ignored, if it is an unknown EWLM correlator. The default is NO_EWLM_RCVD_CORR. indicates that parameter EWLM_RCVD_CORR has not been specified.

To code: Specify the RS-type address, or address in register (2)-(12), of a character field.

,MESSAGES_SENT=*messages_sent*
,MESSAGES_SENT=NO_MESSAGES_SENT
An optional input parameter, which contains the number of messages sent to other applications. This value is added to the total messages_sent value of the work unit. The total messages_sent value should not exceed 32767. The default is NO_MESSAGES_SENT. indicates that parameter MESSAGES_SEND has not been specified.

To code: Specify the RS-type address, or address in register (2)-(12), of a fullword field, or specify a literal decimal value.

,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.

To code: Specify the RS-type address, or address in register (2)-(12), of a 32 bit field.

,MONTKN64=*montkn64*
A required input parameter which contains the long delay monitoring token.

To code: Specify the RS-type address, or address in register (2)-(12), of a 64 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. If you specify 15, GPR15, REG15, or R15 (within or without parentheses), the value will be left in GPR 15.

To code: Specify the RS-type address of a fullword field, or register (2)-(12) or (15), (GPR15), (REG15), or (R15).

,RSNCODE=*rsncode*

An optional output parameter into which the reason code is to be copied from GPR 0. If you specify 0, 00, GPR0, GPR00, REG0, REG00, or R0 (within or without parentheses), the value will be left in GPR 0.

To code: Specify the RS-type address of a fullword field, or register (0) or (2)-(12), (00), (GPR0), (GPR00), REG0), (REG00), or (R0).

ABEND codes

None.

Return codes and reason codes

When the IWM4MUPD 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 87. Return and Reason Codes for the IWM4MUPD Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.
4	xxxx0402	**Equate Symbol**: IwmRsnCodeNoMonEnv **Meaning**: The input monitoring token indicates no monitoring environment was established. **Action**: Establish a monitoring environment by macro IWM4MCRE.
4	xxxx0441	**Equate Symbol**: IwmRsnCodeTooManyMsgCorrs **Meaning**: The correlator passed to EWLM_RCVD_CORR is ignored, since the maximum number of supported correlators has been reached. **Action**: None required.
4	xxxx0443	**Equate Symbol**: IwmRsnCodeTooManyMsgsSent **Meaning**: The value passed to MESSAGES_SENT is ignored, since the maximum number of messages sent is reached. **Action**: None required.

Table 87. Return and Reason Codes for the IWM4MUPD Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
4	xxxx0444	**Equate Symbol**: IwmRsnCodeTooManyMsgsReceived **Meaning**: The messages received counter has reached the maximum value. **Action**: None required.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0820	**Equate Symbol**: IwmRsnCodeBadMonEnv **Meaning**: Input monitoring environment does not pass short form validity checking. **Action**: Check for possible storage overlay.
8	xxxx08AC	**Equate Symbol**: IwmRsnCodeTranNotStarted **Meaning**: No work unit has been started by IWM4MSTR for the specified monitoring environment. **Action**: Start a work unit by IWM4MSTR macro, before issuing this macro.

Example

None.

IWM4MXFR — Monitoring environment transfer

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 IWM4MRLT 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. The **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.

Note: This service was previously called IWMMXFER for 31-bit addressing only (see "IWMMXFER — Transfer monitoring environment" on page 1001).

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 IWM4MXFR 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 delay monitoring token is associated with an ARM work request (EWLM=YES was specified on IWMCONN and the monitoring environment was created using that CONNTKN), the caller must be in supervisor state or have PKM authority to key 0.
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31- or 64-bit. If in 64-bit addressing mode, code SYSSTATE AMODE64=YES before invoking this macro.
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks may be held.
Control parameters:	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. 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(CONTINUE|RETURN) 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 IWM4MXFR or IWM4MSWC) 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 IWM4MXFR FUNCTION(CONTINUE) is used and the time that either IWM4MXFR FUNCTION(RETURN) is used against the dependent monitoring environment OR IWM4MSWC FUNCTION(RETURN) is used against the parent monitoring environment.
8. Only limited validity checking is done on the input monitoring tokens.
9. This macro may only be used on z/OS V2R1 or later.

Input register information

Before issuing the IWM4MXFR 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 following general purpose registers (GPRs) have to contain the specified information:

Register
Contents

13 The address of a 216-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

```
►►──┬──────┬──b──IWM4MXFR──b──────────────────────────────────────────────────►
    └─name─┘

                      ┌─,RUNTIME_VER=SHORT_FORM─┐
►──┬─FUNCTION=CONTINUE─┼─────────────────────────┼──────────────────────────────┬──►
   │                  └─,RUNTIME_VER=MINIMAL────┘                              │
   │                ┌─,RUNTIME_VER=SHORT_FORM─┐  ┌─,WORKREQ_STA=IWMEWLMARMSTATUSNONE─┐ │
   └─FUNCTION=RETURN─┼─────────────────────────┼──┼──────────────────────────────────┼─┘
                    └─,RUNTIME_VER=MINIMAL────┘  └─,WORKREQ_STA=workreq_sta────────┘

►──┬─,MONTKN=montkn─────┬──┬─,PARENTKEYP=VALUE─,PARENTKEY=parentkey─┬──┬─,PARENTENV=NOSWITCH──┬──►
   └─,MONTKN64=montkn64─┘  ├─,PARENTKEYP=PSWKEY────────────────────┤  └─,PARENTENV=SECONDARY─┘
                           └─,PARENTKEYP=UNKNOWN───────────────────┘

   ┌─,EWLM=NO─┐  ┌─,COMPCODE=YES─┐
►──┴──────────┴──┼───────────────┼──┬────────────────┬──┬────────────────┬──────►
                 └─,COMPCODE=NO──┘  └─,RETCODE=retcode─┘  └─,RSNCODE=rsncode─┘

                         ┌─,COMPLETE─┐
►─,MF=(M─,list addr──────┼───────────┼─)──────────────────────────────────────►◄
                         └─,NOCHECK──┘
```

Parameters

The parameters are explained as follows:

name
An optional symbol, starting in column 1, that is the name on the IWM4MXFR 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.

,EWLM=NO
An optional parameter, which indicates if this work manager intents 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.

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.

,MONTKN64=*montkn64*
A required input parameter which contains the long delay monitoring token for the dependent environment.

To code: Specify the RS-type address, or address in register (2)-(12), of a 64 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.

,RETCODE=*retcode*
An optional output parameter into which the return code is to be copied from

GPR 15. If you specify 15, GPR15, REG15, or R15 (within or without parentheses), the value will be left in GPR 15.

To code: Specify the RS-type address of a fullword field, or register (2)-(12) or (15), (GPR15), (REG15), or (R15).

,RSNCODE=*rsncode*
An optional output parameter into which the reason code is to be copied from GPR 0. If you specify 0, 00, GPR0, GPR00, REG0, REG00, or R0 (within or without parentheses), the value will be left in GPR 0.

To code: Specify the RS-type address of a fullword field, or register (0) or (2)-(12), (00), (GPR0), (GPR00), REG0), (REG00), or (R0).

,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 IWM4MABN service, the completion status IwmEwlmArmStatusGood will be reported to EWLM. If an abnormal event was reported via IWM4MABN, 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, or specify a literal decimal value.

ABEND codes

None.

Return codes and reason codes

When the IWM4MXFR 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 88. Return and Reason Codes for the IWM4MXFR Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.
4	xxxx0402	**Equate Symbol**: IwmRsnCodeNoMonEnv **Meaning**: Input monitoring token indicates no monitoring environment was established. **Action**: None required.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx081F	**Equate Symbol**: IwmRsnCodeNoRelate **Meaning**: NO Parent environment exists since Relate Function(Continue) has not been performed or has not been performed subsequent to a Relate Function(Delete). **Action**: Check for possible storage overlay and whether Relate Function(Continue) has been used properly.
8	xxxx0820	**Equate Symbol**: IwmRsnCodeBadMonEnv **Meaning**: Monitoring environment does not pass verification.
8	xxxx0822	IwmRsnCodeBadParEnv: Parent monitoring environment does not pass verification. **Action**: Check for possible storage overlay.

Table 88. Return and Reason Codes for the IWM4MXFR Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0895	**Equate Symbol**: IwmRsnCodeEWLMServNotEnabled **Meaning**: Service is not enabled because monitoring environment can not be associated with EWLM work requests. **Action**: Specify the parameter WORKREQ_STA only when the monitoring environment is created with IWM4MCRE EWLM=YES or the address space is connected with IWMCONN EWLM=YES and the connect token is passed to IWM4MCRE when creating the monitoring environment.
8	xxxx0897	**Equate Symbol**: IwmRsnCodeTranStatusInvalid **Meaning**: Passed work request completion status is not valid. **Action**: Check the EWLM ARM interface specification for valid completion status values.

Example

None.

IWM4MXTR — Monitoring environment extract service

The purpose of this service is to extract information about the monitoring environment which was previously passed through IWM4MINI/IWM4MRLT. When IWM4MRLT 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 IWM4MRLT, 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.

The **EWLM_S_CURCORR** keyword should be specified only, if a work unit has been started by IWM4MSTR.

When no output keywords are specified, the service merely checks whether a monitoring environment was established and passes short form checking.

Note: This service was previously called IWMMEXTR for 31-bit addressing only (see "IWMMEXTR — Monitoring environment extract" on page 936).

Environment

The requirements for the caller are:

Minimum authorization:	Either problem state or supervisor state. Any PSW key.
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31- or 64-bit. If in 64-bit addressing mode, code `SYSSTATE AMODE64=YES` before invoking this macro.
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	Suspend locks are allowed.
Control parameters:	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 bits 0-31 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.
5. This macro may only be used on z/OS V2R1 or later.

Input register information

Before issuing the IWM4MXTR 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 following general purpose registers (GPRs) have to contain the specified information:

Register	Contents
13	The address of a 216-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. The reason code is stored in bits 0-31
1	Used as work register by the system
2-13	Unchanged
14	Used as work register by the system
15	Return code stored in bits 0-31

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

```
                                                          ┌─,EWLM=NO─┐
►►──┬──────┬──b──IWM4MXTR──b──┬─MONTKN=montkn─────┬──────┴──────────┴──────────────────────►
    └─name─┘                  └─MONTKN64=montkn64─┘

►──┬────────────────┬──┬────────────────────┬──┬───────────────────────────┬───────────────►
   └─,OMONTKN=omontkn┘  └─,OMONTKN64=omontkn64┘  └─,OWNER_TOKEN=owner_token─┘

►──┬─────────────────────────┬──┬─────────────────────────────┬────────────────────────────►
   └─,OWNER_DATA=owner_data─┘  └─,ARRIVALTIME=arrivaltime─┘

►──┬──────────────────┬──┬────────────────┬──┬──────────────────┬──────────────────────────►
   └─,TRXNAME=trxname─┘  └─,USERID=userid─┘  └─,SERVCLS=servcls─┘

►──┬───────────┬──┬─────────────────────────────┬──────────────────────────────────────────►
   └─,ASID=asid┘  └─,ENCLAVE_TOKEN=enclave_token─┘

►──┬──────────────────────────┬──┬───────────────────────────────┬─────────────────────────►
   └─,TTRACETOKEN=ttracetoken─┘  └─,ABNORMAL_COND=abnormal_cond─┘

►──┬──────────────────────────┬──┬─────────────────────────┬───────────────────────────────►
   └─,EWLM_CHCORR=ewlm_chcorr─┘  └─,EWLM_PACTKN=ewlm_pactkn─┘

►──┬────────────────────────────────┬──┬──────────────────┬────────────────────────────────►
   └─,EWLM_S_CURCORR=ewlm_s_curcorr─┘  └─,RETCODE=retcode─┘

                                                 ┌─,COMPLETE─┐
►──┬──────────────────┬──,MF=(M─,list addr──┴───────────┴──)─────────────────────────────►◄
   └─,RSNCODE=rsncode─┘                         └─,NOCHECK──┘
```

Parameters

The parameters are explained as follows:

`name`
: An optional symbol, starting in column 1, that is the name on the IWM4MXTR 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 IWM4MXFR 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.

,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.

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=NO
An optional parameter, which indicates if this work manager intents 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.

,EWLM_CHCORR=*ewlm_chcorr*
An optional output parameter, which contains the cross platform Enterprise Workload Management (EWLM) correlator of the work request created by IWM4MINI. Normally this correlator is different from the current correlator of the work unit created by IWM4MSTR.

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.

,EWLM_S_CURCORR=*ewlm_s_curcorr*
An optional output parameter, which contains the current correlator of the work unit started by IWM4MSTR. Normally this correlator is different from the child correlator of the work request created by IWM4MINI.

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.

,MONTKN64=*montkn64*
A required input parameter which contains the long delay monitoring token.

To code: Specify the RS-type address, or address in register (2)-(12), of a 64 bit field.

,OMONTKN=*omontkn*
An optional output parameter, which is to receive the delay monitoring token. This option can be used to convert a long delay monitoring token into the short form.

To code: Specify the RS-type address, or address in register (2)-(12), of a 32 bit field.

,OMONTKN64=*omontkn64*
An optional output parameter, which is to receive the long delay monitoring token. This option can be used to convert a short delay monitoring token into the long form.

To code: Specify the RS-type address, or address in register (2)-(12), of a 64 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.

,RETCODE=*retcode*
An optional output parameter into which the return code is to be copied from GPR 15. If you specify 15, GPR15, REG15, or R15 (within or without parentheses), the value will be left in GPR 15.

To code: Specify the RS-type address of a fullword field, or register (2)-(12) or (15), (GPR15), (REG15), or (R15).

,RSNCODE=*rsncode*
An optional output parameter into which the reason code is to be copied from GPR 0. If you specify 0, 00, GPR0, GPR00, REG0, REG00, or R0 (within or without parentheses), the value will be left in GPR 0.

To code: Specify the RS-type address of a fullword field, or register (0) or (2)-(12), (00), (GPR0), (GPR00), REG0), (REG00), or (R0).

,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.

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 codes and reason codes

When the IWM4MXTR 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 89. Return and Reason Codes for the IWM4MXTR Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.
4	xxxx0402	**Equate Symbol**: IwmRsnCodeNoMonEnv **Meaning**: Monitoring token indicates that no monitoring environment exists. **Action**: None required.
4	xxxx040A	**Equate Symbol**: IwmRsnCodeOutputAreaTooSmall **Meaning**: The output area is too small to contain all the available information.
4	xxxx040C	**Equate Symbol**: IwmRsnCodeMonEnvLacksInfo **Meaning**: Input monitoring environment does not contain the necessary information to return the data requested. **Action**: None required.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0820	**Equate Symbol**: IwmRsnCodeBadMonEnv **Meaning**: Monitoring environment does not pass short form verification. **Action**: Check for possible storage overlay.
8	xxxx0895	**Equate Symbol**: IwmRsnCodeEWLMServNotEnabled **Meaning**: Service is not enabled because monitoring environment can not be associated with EWLM work requests. **Action**: Specify the parameter **EWLM_CHCORR** or **EWLM_PACTKN** only when the monitoring environment is created with IWM4MCRE EWLM=YES or the address space is connected with IWM4CON EWLM=YES and the connect token is passed to IWM4MCRE when creating the monitoring environment.
8	xxxx08AC	**Equate Symbol**: IwmRsnCodeTranNotStarted **Meaning**: A work unit has not been started. **Action**: Start a work unit by IWM4MSTR macro, before specifying the **EWLM_S_CURCORR** parameter.

Example

None.

IWM4OPTQ — Query IEAOPT*xx* parameters

The IWM4OPTQ service queries the current IEAOPT*xx* settings in the system and returns a list of the IEAOPT*xx* parameters, including the actual value, unit, default value, and description.

The caller of IWM4OPTQ must provide storage to contain all of the parameter information. This storage area must reside in the caller's primary address space.

It is possible that the storage required by IWM4OPTQ can change such that multiple calls to IWM4OPTQ are required to obtain data. Users of IWM4OPTQ should take this into consideration when obtaining an amount of storage for the IWM4OPTQ service to use.

If the caller does not provide enough storage to contain all of the parameter information, the IWM4OPTQ service returns a return code and reason code pair to indicate that the storage area specified by the **`OPTINFO_BLOCK`** input parameter is too small. The **`QUERYLEN`** output parameter will be set to the required size for the storage area specified by **`OPTINFO_BLOCK`**. No IEAOPT*xx* parameter information is returned.

Applications that monitor the current system environment can use this service to display the SRM and WLM parameter settings.

The output of the IWM4OPTQ service is a data area mapped by the IWMWOPTI macro and provides a point-in-time snapshot of the parameter settings on the current system.

Environment

The requirements for the caller are:

Minimum authorization:	Problem state, any PSW key.
Dispatchable unit mode:	Task
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31-bit or 64-bit. If in 64-bit addressing mode, code `SYSSTATE AMODE=YES` before invoking this macro.
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts.
Locks:	No locks may be held.
Control parameters:	Control parameters must be in the primary address space.

Programming requirements

1. The macro CVT must be included to use this macro. The macro IWMYCON must be included to use this macro.

   Note: 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 IWM4OPTQ 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.

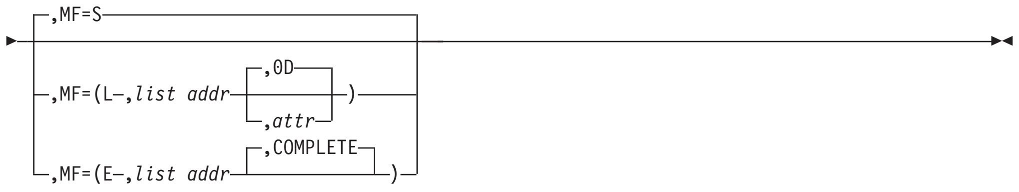
Performance implications

None.

Syntax

The syntax of the IWM4OPTQ macro is as follows:

```
►►──┬──────┬──IWM4OPTQ──OPTINFO_BLOCK=optinfo_block──,ANSLEN=anslen──,QUERYLEN=querylen──────►
    └─name─┘

                                         ┌─,PLISTVER=IMPLIED_VERSION─┐
►──┬─────────────────┬──┬─────────────────┬──┼───────────────────────────┼──────────────────►
   └─,RETCODE=retcode─┘  └─,RSNCODE=rsncode─┘  ├─,PLISTVER=MAX─────────────┤
                                         └─,PLISTVER=1───────────────┘
```



Parameters

The parameters are explained as follows:

name
: An optional symbol, starting in column 1, that is the name on the IWM4OPTQ macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

,ANSLEN=*anslen*
: A required input parameter that contains the length of the **OPTINFO_BLOCK** storage area, 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.

OPTINFO_BLOCK=*optinfo_block*
Is the name (RS-type), or address in register (2)-(12), of a required character input of an output area to contain information provided by this service. The format of this area is mapped by IWMWOPTI and should only be considered valid upon return code zero from this service.

OPTINFO_BLOCK is not the address of a pointer but the address of the output area

,PLISTVER=IMPLIED_VERSION
,PLISTVER=MAX
,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 that are 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.
- 1, if you use the currently available parameters.

To code: Specify one of the following:

- IMPLIED_VERSION
- MAX
- A decimal value of 1

,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 system's IEAOPT*xx* parameter descriptions (that is, the amount of data returned by the IWM4OPTQ 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).

ABEND codes

None.

Return codes and reason codes

When the IWM4OPTQ 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.

Table 90 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 90. Return codes and reason codes for the IWM4OPTQ macro

Return code	Reason code	Equate symbol, meaning, and action
0	—	**Equate symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
4	—	**Equate symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.
4	*xxxx*040A	**Equate symbol**: IwmRsnCodeOutputAreaTooSmall **Meaning**: The output area supplied is too small to receive all the available information. The variable specified by the **QUERYLEN** keyword will contain the size of the storage required to hold the returned data area. **Action**: None required. If necessary, invoke the service again with an output area of sufficient size to receive all information.
8	—	**Equate symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	*xxxx*0801	**Equate symbol**: IwmRsnCodeSrbMode **Meaning**: The caller is in SRB mode. **Action**: Avoid requesting this function while in SRB mode.
8	*xxxx*0803	**Equate symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.
8	*xxxx*0804	**Equate symbol**: IwmRsnCodeLocked **Meaning**: The caller is locked. **Action**: Avoid requesting this function while locked.
8	*xxxx*080B	**Equate symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list. **Action**: Check for possible storage overlay.
8	*xxxx*0823	**Equate symbol**: IwmRsnCodeDatoff **Meaning**: The caller invoked the service while DATOFF **Action**: Avoid requesting this function in this environment.

Table 90. Return codes and reason codes for the IWM4OPTQ macro (continued)

Return code	Reason code	Equate symbol, meaning, and action
8	*xxxx*0824	**Equate symbol**: IwmRsnCodeAmode24 **Meaning**: The caller invoked the service but was in 24-bit addressing mode. **Action**: Request this function only when you are in 31-bit addressing mode.
8	*xxxx*0828	**Equate symbol**: IwmRsnCodeBadVersion **Meaning**: Version number in parameter list is not valid or the length specified is incorrect. **Action**: Check for possible storage overlay of the parameter list.

Example

To query the IEAOPT*xx* settings for the current system, specify:

```
         IWM4OPTQ OPTINFO_BLOCK=OPTINFO,
                  ANSLEN=ALEN,
                  QUERYLEN=QLEN,
                  RETCODE=RC,RSNCODE=RSN
*
* Storage areas
*
OPTINFO  DS    CL8192          Output area
ALEN     DS    F               Length of output area
QLEN     DS    F               Length of returned data
RC       DS    F               Return code
RSN      DS    F               Reason code
```

IWM4QDE — Delete a request from the queue for an execution address space

This service deletes a work request that was previously inserted using the IWM4QIN service, if it has not been selected using the IWM4SSL service.

Note: This service was previously called IWMQDEL for 31-bit addressing only (see "IWMQDEL — Delete a request from the queue for an execution address space" on page 1009).

Environment

The requirements for the caller are:

Minimum authorization:	Supervisor state or program key mask (PKM) allowing keys 0-7.
Dispatchable unit mode:	Task or SRB
Cross memory mode:	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).
AMODE:	31- or 64-bit. If in 64-bit addressing mode, code `SYSSTATE AMODE64=YES` before invoking this macro.
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks may be held.
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, 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.

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

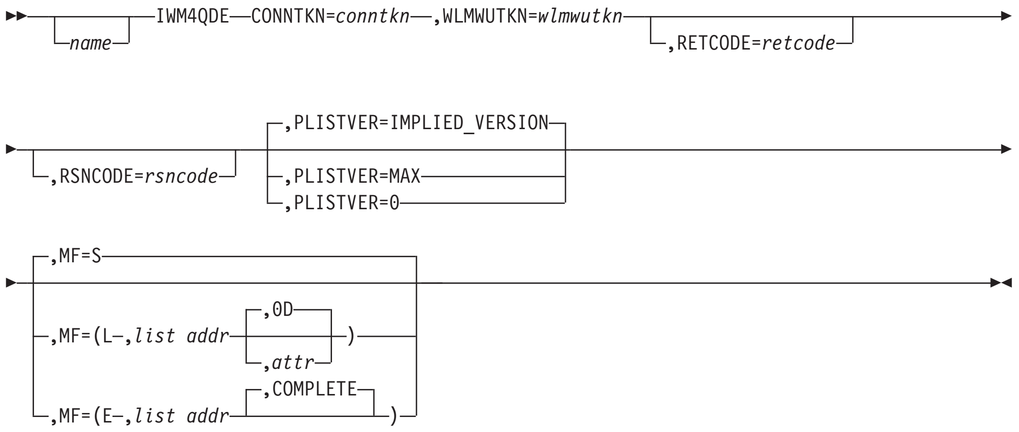
Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

Performance implications

None.

Syntax

The syntax of the IWM4QDE macro is as follows:



Parameters

The parameters are explained as follows:

name
An optional symbol, starting in column 1, that is the name on the IWM4QDE 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 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 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 codes 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 91. Return and Reason Codes for the IWM4QDE Macro

Return Code	Reason Code	Equate Symbol, Meaning, and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.
4	xxxx0408	**Equate Symbol**: IwmRsnCodeWorkNotFound: **Meaning**: No work matching the input search criteria was found. **Action**: None required.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: The caller is locked. **Action**: Avoid requesting this function while locked.
8	xxxx080B	**Equate Symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list. **Action**: Check for possible storage overlay.
8	xxxx0821	**Equate Symbol**: IwmRsnCodeBadConn **Meaning**: Input connect token does not pass validity checking. **Action**: 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	**Equate Symbol**: IwmRsnCodeDatoff **Meaning**: The caller invoked the service while DATOFF **Action**: Avoid requesting this function in this environment.
8	xxxx0824	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: The caller invoked the service but was in 24-bit addressing mode. **Action**: Request this function only when you are in 31-bit addressing mode.
8	xxxx0825	**Equate Symbol**: IwmRsnCodeAscModeNotPrimary **Meaning**: The caller invoked the service but was not DAT on Primary ASC mod **Action**: Avoid requesting this function in this environment.

Table 91. Return and Reason Codes for the IWM4QDE Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx0827	**Equate Symbol**: IwmRsnCodeRsvdNot0 **Meaning**: Reserved field in parameter list was non-zero. **Action**: Check for use of keywords that are not supported by the MVS release on which the program is running.
8	xxxx0828	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: Version number in parameter list is not valid. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx083F	**Equate Symbol**: IwmRsnCodePrimaryNotOwnConn **Meaning**: Primary address space does not own the passed connect token. **Action**: Avoid requesting this function while primary address space does not own the connect token.
8	xxxx0840	**Equate Symbol**: IwmRsnCodeServiceNotEnabled **Meaning**: Caller's space connection is not enabled for this service. **Action**: Make sure that Queue_Manager=Yes, or Server_Manager=Yes with Server_Type=Queue is specified on the IWM4CON request to enable this service.
8	xxxx0848	**Equate Symbol**: IwmRsnCodeBadWorkUnitToken **Meaning**: The work unit token is not valid. **Action**: Check the specification of the WLMWUTKN parameter.
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error. **Action**: Contact your system programmer.

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
```

IWM4QHLT — Query server health indicators

The IWM4QHLT service provides information about health indicators that were set for server address spaces via the IWM4HLTH or IWMSRSRG services. IWM4QHLT allows the caller to obtain the health information for:

- All address spaces for which health indicators were provided
- A list of particular address spaces

The caller must provide an area of storage in the **ANSAREA** parameter and the length of that area in the **ANSLEN** parameter for IWM4QHLT to place the health information. IWM4QHLT returns the actual length of the information in the **QUERYLEN** parameter.

If a caller does not know the size of the answer area that is required by the service, it should issue IWM4QHLT with **ANSLEN** set to zero. The length of the answer area is placed in **QUERYLEN**.

The answer area is mapped by the IWMWQHAA data area, which is described in *z/OS MVS Data Areas, Volume 4*.

The returned information is not serialized upon return to the caller, so it might be outdated due to a change in health indicators.

Environment

The requirements for the caller are:

Minimum authorization:	Supervisor state or program key mask (PKM) allowing keys 0-7
Dispatchable unit mode:	Task
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31-bit or 64-bit. If in 64-bit addressing mode, code SYSSTATE AMODE64=YES before invoking this macro.
ASC mode:	Primary or access register (AR). If in AR mode, specify SYSSTATE ASCENV=AR before invoking this macro.
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks can be held. FRR environments can be established.
Control parameters:	The caller of IWM4QHLT must provide storage for an answer area that is mapped by IWMWQHAA. This answer area might 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).

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. The high-order halfword of register 0, and the reason code variable when specified, might be non-zero and represents diagnostic data that is NOT part of the external interface. The high-order halfword should thus be excluded from

comparison with the reason code values that are 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-aligned and padded with blanks on the right, as needed, to occupy the specified number of bytes.

Restrictions

None.

Input register information

Before issuing the IWM4QHLT 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.

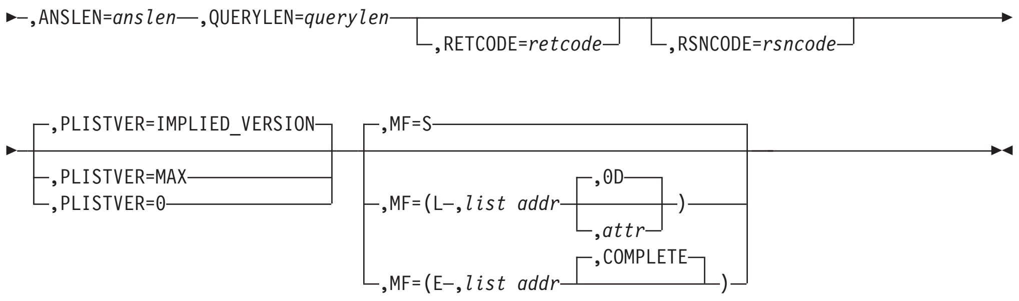
Performance implications

None.

Syntax

The syntax of the IWM4QHLT macro is as follows:

```
                                ┌─ASID_LIST=NO_ASID_LIST──────────────────┐
►►─┬──────┬─IWM4QHLT──┴─ASID_LIST=asid_list,ASID_NUM=asid_num─┴─,ANSAREA=ansarea──────►
   └─name─┘
```



Parameters

The parameters are explained as follows:

name
An optional symbol, starting in column 1, that is the name on the IWM4QHLT macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

,ANSAREA=*ansarea*
A required output parameter that specifies an answer area, *ansarea*, to contain the data that is returned by the query service. The area is mapped by the IWMWQHAA mapping macro. If the length of *ansarea* is insufficient to hold all of the data to be returned, no data is returned.

To code: Specify an RS-type address or address in register (2)-(12) of a character field.

,ANSLEN=anslen
A required input parameter that contains the length of the answer area (*ansarea*).

To code: Specify the RS-type address or address in register (2)-(12) of a fullword field.

ASID_LIST=NO_ASID_LIST
ASID_LIST=*asid_list*
An optional input parameter that specifies an area for the list of ASIDs for which health information should be returned. Each entry (ASID) is a halfword field in hexadecimal format. A maximum number of 100 entries in the list is supported. The default value is `NO_ASID_LIST`, which indicates that no ASID list was passed and health information for all address spaces for which health indicators was provided should be returned.

To code: Specify the RS-type name or address in register (2)-(12) of a field specifying the area for the list of ASIDs.

,ASID_NUM=*asid_num*
A required input parameter for `ASID_LIST=asid_list` that contains the number of ASIDs in *asid_list*.

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 can 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 that is 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 that are used for a request and with the same value on all of the macro forms. The values are:

IMPLIED_VERSION
The lowest version that allows all parameters that are specified on the request to be processed. If you omit the **PLISTVER** parameter, IMPLIED_VERSION is the default.

MAX
Indicates that 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 Indicates to 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 that contains the length of the storage area that is required by the IWM4QHLT service. The length of the area might 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 or address in register (2)-(12) of a fullword 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 or address in register (2)-(12) of a fullword field.

ABEND codes

None.

Return codes and reason codes

When the IWM4QHLT 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.

Table 92 identifies the hexadecimal return and reason codes and the equate symbol that is associated with each reason code. IBM support personnel might request the entire reason code, including the *xxxx* value.

Table 92. Return and reason codes for the IWM4QHLT macro

Return code	Reason code	Equate symbol, meaning, and action
0	—	**Equate symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
4	—	**Equate symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.
4	*xxxx*0408	**Equate symbol**: IwmRsnCodeWorkNotFound **Meaning**: No address spaces matching the input ASIDs were found, or none of the input ASIDs have health information. . **Action**: None required.

Table 92. Return and reason codes for the IWM4QHLT macro (continued)

Return code	Reason code	Equate symbol, meaning, and action
4	*xxxx*040A	**Equate symbol**: IwmRsnCodeOutputAreaTooSmall **Meaning**: The supplied output area is too small to receive all the available information. **Action**: None required. If necessary, reinvoke the service with an output area of sufficient size (returned in **QUERYLEN**) to receive all information.
8	—	**Equate symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	*xxxx*0801	**Equate symbol**: IwmRsnCodeSrbMode **Meaning**: Caller is in SRB mode. **Action**: Avoid requesting this function while in SRB mode.
8	*xxxx*0803	**Equate symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.
8	*xxxx*0804	**Equate symbol**: IwmRsnCodeLocked **Meaning**: Caller is locked. **Action**: Avoid requesting this function while locked.
8	*xxxx*080B	**Equate symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list. **Action**: Check for possible storage overlay.
8	*xxxx*0823	**Equate symbol**: IwmRsnCodeDatoff **Meaning**: The caller invoked the service while DATOFF. **Action**: Avoid requesting this function in this environment.
8	*xxxx*0824	**Equate symbol**: IwmRsnCodeAmode24 **Meaning**: The caller invoked the service but was in 24-bit addressing mode. **Action**: Request this function only in 31-bit or 64-bit addressing mode.
8	*xxxx*0827	**Equate symbol**: IwmRsnCodeRsvdNot0 **Meaning**: Reserved field in parameter list was non-zero. **Action**: Check for possible storage overlay of the parameter list.
8	*xxxx*0828	**Equate symbol**: IwmRsnCodeBadVersion **Meaning**: The version number in the parameter list or the version length field is not valid, or this service was called on a z/OS release where it is not supported. **Action**: Check for possible storage overlay of the parameter list.

Table 92. Return and reason codes for the IWM4QHLT macro (continued)

Return code	Reason code	Equate symbol, meaning, and action
8	*xxxx*0829	**Equate symbol**: IwmRsnCodeBadOptions **Meaning**: Parameter list omits required parameters, supplies mutually exclusive parameters, provides data that is associated with options not selected, or specifies more than 100 ASIDs in **ASID_LIST**. **Action**: Check for possible storage overlay of the parameter list.
8	*xxxx*0830	**Equate symbol**: IwmRsnCodeBadAlet **Meaning**: Caller specified an invalid ALET for the storage pointed to by the **ANSAREA** parameter. **Action**: Check for possible storage overlay of the parameter list or variable.
C	—	**Equate symbol**: IwmRetCodeEnvError **Meaning**: Environmental error.
C	*xxxx*0C01	**Equate symbol**: IwmRsnCodeNoStg **Meaning**: Storage is not available for the request. **Action**: There is a storage shortage. The function may work successfully at a later time.
10	—	**Equate symbol**: IwmRetCodeCompError **Meaning**: Component error. **Action**: No action is required. The service might be successful if invoked again.

Example

To query server health indicators, specify:

```
         IWM4QHLT ASID_LIST=ASLST,ASID_NUM=ASNUM,
                  ANSAREA=AAREA,
                  ANSLEN=ALEN,
                  QUERYLEN=QLEN,
                  RETCODE=RC,RSNCODE=RSN
*
* Storage areas
*
ASLST    DS    Cxx             List of server ASIDs to query
ASNUM    DS    F               Number of ASIDs in list
AAREA    DS    CLxx            Answer area
ALEN     DS    F               Length of answer area
QLEN     DS    F               Length of returned data
RC       DS    F               Return code
RSN      DS    F               Reason code
```

IWM4QIN — Insert 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.

Note: This service was previously called IWMQINS for 31-bit addressing only (see "IWMQINS — Insert a request onto the queue for an execution address space" on page 1016).

Environment

The requirements for the caller are:

Minimum authorization:	Supervisor state or program key mask (PKM) allowing keys 0-7.
Dispatchable unit mode:	Task or SRB
Cross memory mode:	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`).
AMODE:	31- or 64-bit. If in 64-bit addressing mode, code `SYSSTATE AMODE64=YES` before invoking this macro.
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks may be held.
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, 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

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.

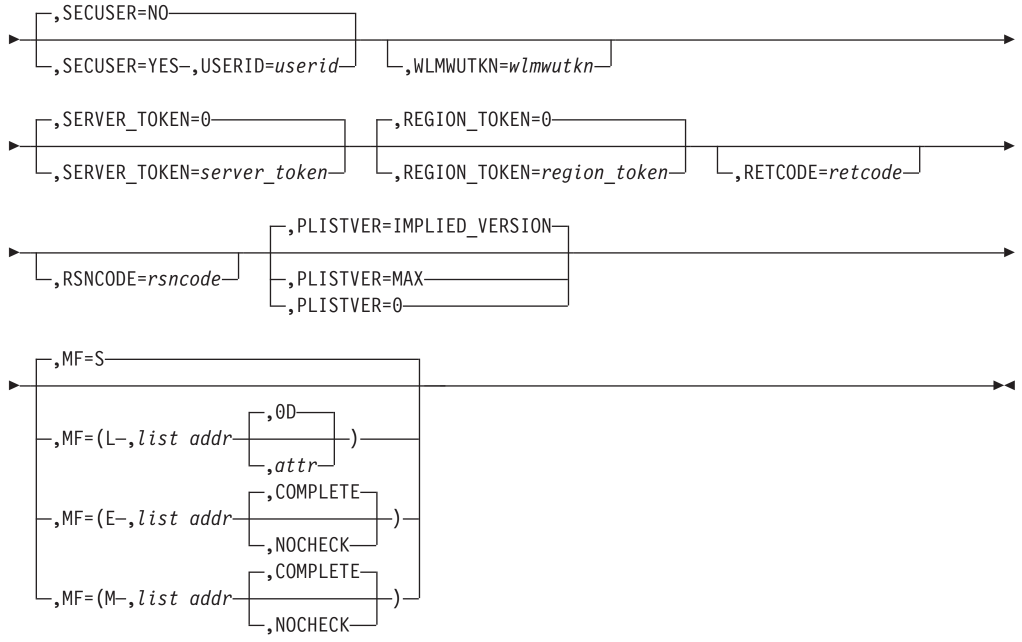
Performance implications

None.

Syntax

The syntax of the IWM4QIN macro is as follows:

```
►►─┬──────┬─IWM4QIN──CONNTKN=conntkn──,ETOKEN=etoken──,USERDATA=userdata──────────►
   └─name─┘

►─,APPLENV=applenv─┬─,DYNAMIC=NO──┬─┬─,DEPENDENT=NO──┬────────────────────────────►
                   └─,DYNAMIC=YES─┘ └─,DEPENDENT=YES─┘
```



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 defintion. 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)**
,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 IWM4QIN in the following order:

-
- Use `IWM4QIN ...MF=(M,`*list-addr*`,COMPLETE)` specifying appropriate parameters, including all required ones.
- Use `IWM4QIN ...MF=(M,`*list-addr*`,NOCHECK)`, specifying the parameters that you want to change.

- Use `IWM4QIN ...MF=(E,`*`list-addr`*`,NOCHECK)` to execute the 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`
  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` 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 IWM4CON and 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.

,SECUSER=YES
Use the specified user ID 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 user ID.

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 codes 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 93. Return and Reason Codes for the IWM4QIN Macro

Return Code	Reason Code	Equate Symbol, Meaning, and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.
4	xxxx042E	**Equate Symbol**: IwmRsnCodeServerNotFound **Meaning**: The server token does not identify an existing server tas The server task may have terminated since the token was obtained. **Action**: If the server task has not terminated, check that the correct token is specified.

Table 93. Return and Reason Codes for the IWM4QIN Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
4	xxxx043A	**Equate Symbol**: IwmRsnCodeRegionNotFound **Meaning**: The region token does not identify a valid server region. **Action**: Please specify the correct region token.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: Caller is locked. **Action**: Avoid requesting this function while locked.
8	xxxx080B	**Equate Symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list. **Action**: Check for possible storage overlay.
8	xxxx0821	**Equate Symbol**: IwmRsnCodeBadConn **Meaning**: Input connect token does not pass validity checking. **Action**: 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	**Equate Symbol**: IwmRsnCodeDatoff **Meaning**: The caller invoked the service while DATOFF **Action**: Avoid requesting this function in this environment.
8	xxxx0824	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: The caller invoked the service but was in 24-bit addressing mode. **Action**: Request this function only when you are in 31-bit addressing mode.
8	xxxx0825	**Equate Symbol**: IwmRsnCodeAscModeNotPrimary **Meaning**: The caller invoked the service but was not in primary ASC mode. **Action**: Avoid requesting this function in this environment.
8	xxxx0827	**Equate Symbol**: IwmRsnCodeRsvdNot0 **Meaning**: Reserved field in parameter list was non-zero. **Action**: Check for use of keywords that are not supported by the MVS release on which the program is running.
8	xxxx0828	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: Version number in parameter list is not valid. **Action**: Check for possible storage overlay of the parameter list.

Table 93. Return and Reason Codes for the IWM4QIN Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx083A	**Equate Symbol**: IwmRsnCodeBadEnclave **Meaning**: Enclave token in parameter list is not valid. **Action**: Check the specification of the **ETOKEN** parameter.
8	xxxx083F	**Equate Symbol**: IwmRsnCodePrimaryNotOwnConn **Meaning**: Primary address space does not own the passed connect token. **Action**: 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.
8	xxxx0840	**Equate Symbol**: IwmRsnCodeServiceNotEnabled **Meaning**: The caller's space connection is not enabled for this service **Action**: Make sure that `Queue_Manager=Yes`, or `Server_Manager=Yes` with `Server_Type=Queue` is specified on the IWM4CON request to enable this service.
8	xxxx0842	**Equate Symbol**: IwmRsnCodeNoWLMConnect **Meaning**: The caller's space disconnected from WLM during processing of the insert request. **Action**: None.
8	xxxx0845	**Equate Symbol**: IwmRsnCodeWrongEnclave **Meaning**: 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. **Action**: Check that the correct enclave token was specified. Check that the server task is invoking the IWM4SSL and IWM4SSM macros in the correct sequence.
8	xxxx089C	**Equate Symbol**: IwmRsnCodeDupAENameInsert **Meaning**: The caller tried to insert to an application environment with a duplicate name within the same node concurrently. **Action**: Make sure not to insert requests to a dynamic and static application environment within the same node concurrently.
C	—	**Equate Symbol**: IwmRetCodeEnvError **Meaning**: Environmental error.
C	xxxx0C01	**Equate Symbol**: IwmRsnCodeNoStg **Meaning**: Storage is not available for the request. **Action**: Contact your system programmer. There is a common storage shortage.

Table 93. Return and Reason Codes for the IWM4QIN Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
C	xxxx0C16	**Equate Symbol**: IwmRsnCodeServerUnavail **Meaning**: A server cannot be started to process the work request. This is probably caused by one of the following: 1. An error in the JCL procedure used to start the server address space. 2. Repeated, unexpected terminations of the server address space. In either of these cases, workload management stops the application environment. A **`DISPLAY WLM`** command shows this state as `INTERNALLY STOPPED`. **Action**: Look at the system log to determine what caused the error: 1. If it is a JCL error, correct the error in the procedure. 2. If it is repeated terminations of the server address space, correct the application error causing the termination. 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. **Note:** 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.
C	xxxx0C1A	**Equate Symbol**: IwmRsnCodeApplNotDefined **Meaning**: The application environment name is not defined in the active WLM policy. **Action**: 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.
C	xxxx0C1B	**Equate Symbol**: IwmRsnCodeApplNotSST **Meaning**: The application environment name is defined for use by a different subsystem type in the active WLM policy. **Action**: 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.
C	xxxx0C1D	**Equate Symbol**: IwmRsnCodeQMgrNotActive **Meaning**: The required Queue Manager is not active. **Action**: 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.
C	xxxx0C22	**Equate Symbol**: IwmRsnCodeApplEnvQuiesced **Meaning**: For server applications connecting to WLM with subsystem type IWEB only: The application environment has been quiesced. The work reqeust is not inserted to the WLM work queue. **Action**: Resume the application environment.

Table 93. Return and Reason Codes for the IWM4QIN Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
C	xxxx0C40	**Equate Symbol**: IwmRsnCodeNoSafCheckPossible **Meaning**: MLS is active but a security check could not be performed probably for one of the following reasons: • • 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,... • A resource or class name is not defined to RACF or the class has not been RAClisted. • The class was RAClisted, but the data space cannot be accessed due to an ALESERV failure. • The class was RAClisted, but the data space has been deleted. • 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. **Action**: Contact your RACF Security Administrator. Check if RACF is properly installed, configured and tuned. Correct the eventual problems.
C	xxxx0C41	**Equate Symbol**: IwmRsnCodeSafCheckFailed **Meaning**: MLS is active. Queue Manager and Server Manager are not authorized to communicate. **Action**: Normally none. If QM and SM really must communicate, conta your RACF Security Administrator. Set the appropriate Security Labels.
C	xxxx0C42	**Equate Symbol**: IwmRsnCodeAletError **Meaning**: Error while accessing access list with ALESERV probably because of one of the following 1. 2. The current access list cannot be expanded. There are no free access list entries and the maximum size has been reached. 3. ALESERV could not obtain storage for an expanded access list. **Action**: 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.
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error. **Action**: Contact your system programmer.

Example

To insert a work request onto the WLM queue manager queues, specify the following:

```
         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
```

IWM4RPT — Report response time

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. Processor consumption data can also be provided.

The secondary purpose of 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 of 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 renders the information that is associated with the monitoring environment unpredictable. To associate a work request with the monitoring environment after use of Report, first use Initialize Mode(Reset) or Relate/Transfer.

Note: This service was previously called IWMRPT for 31 bit addressing only (see "IWMRPT — Report on work request completion" on page 1028).

Environment

The requirements for the caller are:

Minimum authorization:	Supervisor state. PSW key must either be 0 or match the value that is supplied on IWM4CON. PSW key must either be 0 or match the value that is supplied on IWM4MCRE when a monitoring token is passed. PSW key must be 0-7. See "Restrictions" on page 709.
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31- or 64-bit. If in 64-bit addressing mode, code `SYSSTATE AMODE64=YES` before invoking this macro.
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	LOCAL lock is held
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. The high-order halfword of register 0, and the reason code variable when specified, might be non-zero and represents diagnostic data that is NOT part of the external interface. The high-order halfword should be excluded from comparison with the reason code values described above. The constant, IWMRSNCODE_MASK_CONST defined in IWMYCON, might be used for this purpose.
5. All character data, unless otherwise specified, is assumed to be left-aligned and padded with blanks on the right, as needed, to occupy the specified number of bytes.

Restrictions

1. Caller is responsible for error recovery
2. Though the caller is required to be enabled, this requirement is not checked. Violation of this restriction might 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 that is represented by the monitoring token.
   - The monitoring environment must contain the information that is saved by IWM4MINI, not IWM4MRLT.
   - If the key specified on IWM4MCRE was a system key (0-7), then the current PSW key must be 0 or match the key that is 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 can be used only on z/OS V2R1 or higher.

Input register information

Before issuing the IWM4RPT macro, the caller must ensure that the following general-purpose registers (GPRs) contain the specified information:

Register	Contents
13	The address of a 216 byte standard save area in the primary address space.

Before issuing the IWM4RPT 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:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system.

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.

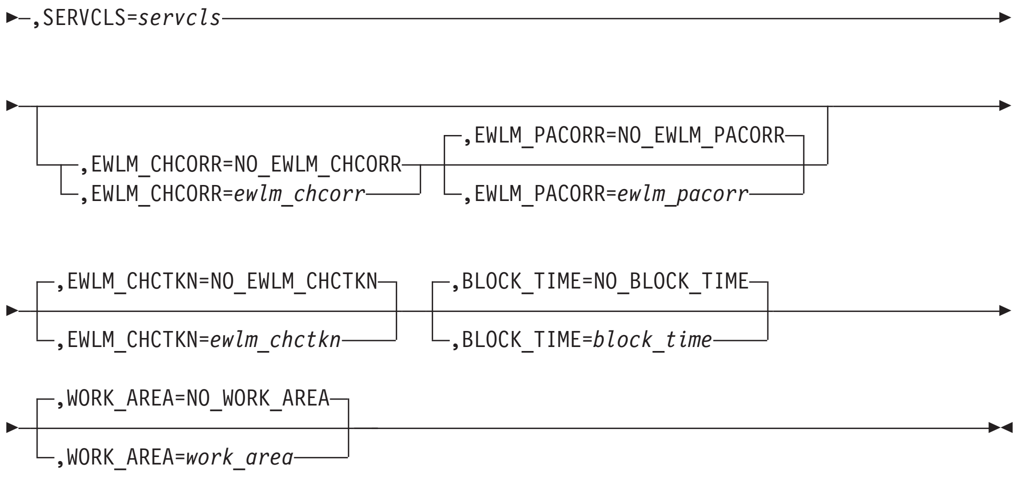
Syntax

main diagram

```
                                    ┌PSWKEYP=CURRENT──────────────┐
►►──┬──────┬─b──IWM4RPT──b──────────┤                             ├──────────────────────────────►
    └name──┘                        └PSWKEYP=VALUE─,PSWKEY=pswkey─┘

                                                                ┌,EWLM=NO─┐
►──┬,MONTKNI=YES──┬,MONTKN=montkn──────┬───┬,CONNTKN=conntkn────┴─────────┴────────────────────►
   │              └,MONTKN64=montkn64──┘   │
   └,MONTKNI=NO─┤ parameters-1 ├───────────┘

   ┌,ENDTIME=CURRENT─┐
►──┤                 ├────────────────────────────────────────────────────────────────────────►
   └,ENDTIME=endtime─┘

►──┬──────────────────────────────────────────────────────────────────┬──────────────────────►
   └,CPUTIME=cputime──,TIMEONCP=timeoncp──,OFFLOADONCP=offloadoncp────┘

     ┌,STATUS=NORMAL─┐  ┌,WORK_COMPCD=NO_WORK_COMPCD─┐
►──┬─┴───────────────┴──┤                            ├────────────────────────┬─────────────►
   │                    └,WORK_COMPCD=work_compcd────┘                        │
   │                  ┌,WORK_COMPCD=NO_WORK_COMPCD─┐                          │
   ├,STATUS=ABNORMAL──┤                            ├──────────────────────────┤
   │                  └,WORK_COMPCD=work_compcd────┘                          │
   ├,STATUS=NORMAL_LE_VAL─,WORK_COMPCD=work_compcd─,OK_THRESHOLD=ok_threshold─┤
   └,STATUS=NORMAL_GE_VAL─,WORK_COMPCD=work_compcd─,OK_THRESHOLD=ok_threshold─┘

   ┌,WORKREQ_STA=IWMEWLMARMSTATUSNONE─┐
►──┤                                  ├──┬──────────────────┬──┬──────────────────┬──────────►
   └,WORKREQ_STA=workreq_sta──────────┘  └,RETCODE=retcode──┘  └,RSNCODE=rsncode──┘

   ┌,PLISTVER=IMPLIED_VERSION─┐  ┌,MF=S──────────────────────────────────┐
►──┼──────────────────────────┼──┤                                       ├──────────────────►◄
   ├,PLISTVER=MAX─────────────┤  │                        ┌,0D───┐       │
   ├,PLISTVER=0───────────────┤  ├,MF=(L─,list addr───────┤      ├─)─────┤
   └,PLISTVER=1───────────────┘  │                        └,attr─┘       │
                                 │                        ┌,COMPLETE─┐   │
                                 └,MF=(E─,list addr───────┴──────────┴─)─┘
```

parameters-1

```
►►──,ARRIVALTIME=arrivaltime──┬,EXSTARTTIMEP=NO──────────────────────────────────┬───────────►
                              └,EXSTARTTIMEP=YES─,EXSTARTTIME=exstarttime────────┘
```



Parameters

The parameters are explained as follows:

name
: An optional symbol, starting in column 1, that is the name on the IWM4RPT 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, which 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 by using a connect token on IWM4MCRE, then the latter connect token is expected to be the same as that specified for IWM4RPT.

: **To code:** Specify the RS-type address, or address in register (2)-(12), of a 32-bit field.

,CPUTIME=*cputime*
: An optional input parameter that contains the total CPU time, in STCK format, for the current work request.

: **To code:** Specify the RS-type address, or address in register (2)-(12), of a 64-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, which 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=NO

An optional parameter, which indicates whether 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.

,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 that is 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 is not reported to EWLM.

Parameter EWLM_CHCORR and EWLM_CHCTKN are mutually exclusive.

The default is NO_EWLM_CHCORR, which 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 that is associated with the work request. Parameter EWLM_CHCORR and EWLM_CHCTKN are mutually exclusive. The default is NO_EWLM_CHCTKN, which 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 that is associated with the work request. The default is NO_EWLM_PACORR, which 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. This should be used only when IWM4MNTF 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. This should be used only when IWM4MNTF 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*,**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 can 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 that is 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 that 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.

,MONTKN64=*montkn64*
When MONTKNI=YES is specified, a required input parameter that contains the long delay monitoring token.

To code: Specify the RS-type address, or address in register (2)-(12), of a 64-bit field.

| **,OFFLOADONCP=***offloadoncp*
| When CPUTIME is specified, a required input parameter that contains the CPU
| time on standard CP that was offload eligible, in STCK format, for the current
| work request.

| **To code:** Specify the RS-type address, or address in register (2)-(12), of a 64-bit
| field.

,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, or specify a literal decimal value.

,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, or specify a literal decimal value.

,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 that are 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 that are 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 parameters referenced in the description of higher versions.
- **1**, which supports the parameters that are supported by 0, as well as CPUTIME, TIMEONCP, and OFFLOADONCP.

To code: Specify one of the following:
- IMPLIED_VERSION
- MAX
- A decimal value of 0 or 1.

,PSWKEY=*pswkey*
When PSWKEYP=VALUE is 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
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. If you specify 15, GPR15, REG15, or R15 (within or without parentheses), the value is left in GPR 15.

To code: Specify the RS-type address of a fullword field, or register (2)-(12) or (15), (GPR15), (REG15), or (R15).

,RSNCODE=*rsncode*
An optional output parameter into which the reason code is to be copied from GPR 0. If you specify 0, 00, GPR0, GPR00, REG0, REG00, or R0 (within or without parentheses), the value is left in GPR 0.

To code: Specify the RS-type address of a fullword field, or register (0) or (2)-(12), (00), (GPR0), (GPR00), REG0), (REG00), or (R0).

,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 that is associated with the Report call has completed normally or not. The default is STATUS=NORMAL.

,STATUS=NORMAL
indicates that work request execution that is associated with the Report call has completed normally.

,STATUS=ABNORMAL
indicates that work request execution that is associated with the Report call has completed abnormally.

,STATUS=NORMAL_LE_VAL
indicates that work request execution that is associated with the Report call has completed normally PROVIDED the work completion code is below or at (<=) the threshold value that is given by OK_THRESHOLD.

,STATUS=NORMAL_GE_VAL
indicates that work request execution that is associated with the Report call has completed normally PROVIDED the work completion code is above or at (>=) the threshold value that is given by OK_THRESHOLD.

| **,TIMEONCP=***timeoncp*
| When CPUTIME is specified, a required input parameter that contains the CPU
| time on standard CP, in STCK request format, for the current work.

| **To code:** Specify the RS-type address, or address in register (2)-(12), of a 64-bit
| field.

,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 IWM4RPT 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, which indicates that no work area is passed.

To code: Specify the RS-type address, or address in register (2)-(12), of a 512-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 that is associated with the Report call. The default is NO_WORK_COMPCD, which 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, or specify a literal decimal value.

,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 that is 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, or specify a literal decimal value.

,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 that is associated with the Report call.

To code: Specify the RS-type address, or address in register (2)-(12), of a fullword field, or specify a literal decimal value.

,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 that is associated with the Report call.

To code: Specify the RS-type address, or address in register (2)-(12), of a fullword field, or specify a literal decimal value.

,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, which 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, or specify a literal decimal value.

ABEND codes

None.

Return codes and reason codes

When the IWM4RPT 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 that is associated with each reason code. IBM support personnel might request the entire reason code, including the **xxxx** value.

Table 94. Return and Reason Codes for the IWM4RPT Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.

Table 94. Return and Reason Codes for the IWM4RPT Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.
4	xxxx0405	**Equate Symbol**: IwmRsnCodeGoalNoMonEnv **Meaning**: System is in goal mode but the input monitoring token indicates that no monitoring environment was established, hence MVS did not receive the information. **Action**: None required.
4	xxxx0409	**Equate Symbol**: IwmRsnCodeNoConn **Meaning**: Connect token does not reflect a successful Connect. The system did not receive the information. **Action**: None required.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx080C	**Equate Symbol**: IwmRsnCodeMonEnvLacksData **Meaning**: Input monitoring environment does not contain the necessary information. **Action**: Ensure that the monitoring environment was established with the necessary information.
8	xxxx080E	**Equate Symbol**: IwmRsnCodeArrTimeGTEndTime **Meaning**: Input arrival time later than end time. **Action**: Check for possible storage overlay of the parameter list or variable.
8	xxxx0820	**Equate Symbol**: IwmRsnCodeBadMonEnv **Meaning**: Input monitoring environment does not pass short form validity checking. **Action**: Check for possible storage overlay.
8	xxxx0821	**Equate Symbol**: IwmRsnCodeBadConn **Meaning**: Input connect token does not pass validity checking. **Action**: Check for possible storage overlay.
8	xxxx082D	**Equate Symbol**: IwmRsnCodeExStTimeGTEndTime **Meaning**: Execution start time greater than execution end time **Action**: Check for possible storage overlay of the parameter list or variable.
8	xxxx0840	**Equate Symbol**: IwmRsnCodeServiceNotEnabled **Meaning**: Caller's space connection is not enabled for this service. **Action**: Avoid requesting this function under the input connection. IWM4CON options must be specified previously to enable this service.

Table 94. Return and Reason Codes for the IWM4RPT Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx087E	**Equate Symbol**: IwmRsnCodeRoMonEnv **Meaning**: Input monitoring environment is report-only. checking. **Action**: Avoid calling this function for report-only monitoring environments.
8	xxxx0894	**Equate Symbol**: IwmRsnCodeInvalidEWLMCorr **Meaning**: 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 gretater than 512 ('0200'x). **Action**: Check the specification of the correlator information.
8	xxxx0895	**Equate Symbol**: IwmRsnCodeEWLMServNotEnabled **Meaning**: Service is not enabled because caller invoked the IWMCONN service with EWLM=NO. **Action**: Specify the parameter **EWLM_CHCORR**, **EWLM_PACORR**, **EWLM_CHCTKN**, or **WORKREQ_STA** only when connected with EWLM=YES.
8	xxxx0897	**Equate Symbol**: IwmRsnCodeTranStatusInvalid **Meaning**: Passed work request completion status is not valid. . **Action**: Check the EWLM ARM interface specification for valid completion status values.
C	—	**Equate Symbol**: IwmRetCodeEnvError **Meaning**: Environmental error.
C	xxxx0C05	**Equate Symbol**: IwmRsnCodeRptNoWorkElt **Meaning**: Report routine was invoked, but no work element was available to save the input information. **Action**: Invoke the function when the conditions are alleviated. This condition may be due to a common storage shortage condition.
C	xxxx0C06	**Equate Symbol**: IwmRsnCodeNoEndTime **Meaning**: No end time was supplied to the service and STCK gave a nonzero condition code. **Action**: No action required.

Example

None.

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. The user can decide if an additional server will be started before previously started servers have connected to WLM, or if WLM needs to wait until all previously started servers have connected before an additional server will be started. 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.

Note: This service was previously called IWMSLIM for 31-bit addressing only (see "IWMSLIM — Application environment limit service" on page 1040).

Environment

The requirements for the caller are:

Minimum authorization:	Problem state. Any PSW key
Dispatchable unit mode:	Task
Cross memory mode:	PASN=HASN=SASN
AMODE:	31- or 64-bit. If in 64-bit addressing mode, code `SYSSTATE AMODE64=YES` before invoking this macro.
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	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.

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

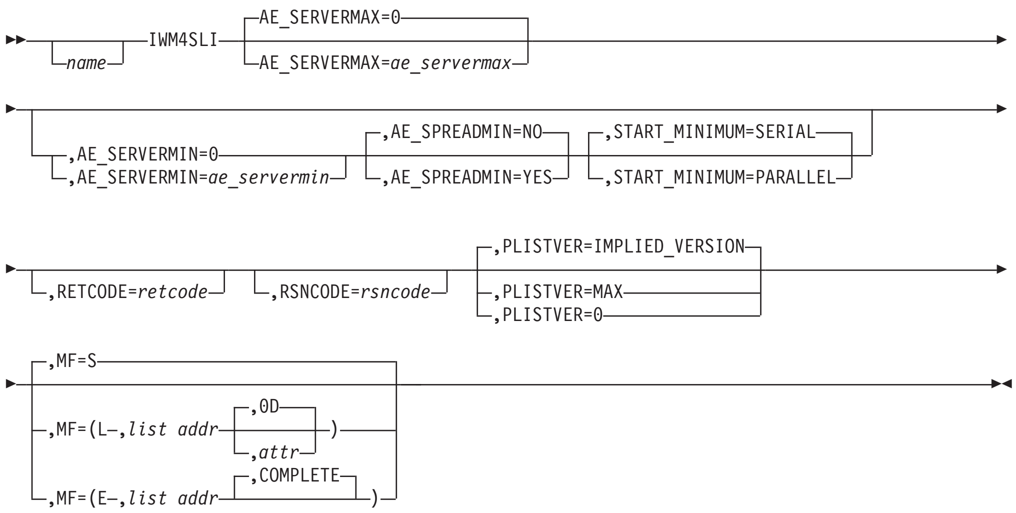
Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

Performance implications

None.

Syntax

The syntax of the IWM4SLI macro is as follows:



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, which indicates that no limit has been specified.

To code: Specify the RS-type address of a halfword field.

,AE_SPREADMIN=NO
,AE_SPREADMIN=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_SPREADMIN=NO.

,AE_SPREADMIN=NO

The server tasks specified in AE_SERVERMIN will be distributed to service classes as needed in order to meet goals.

,AE_SPREADMIN=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:

- `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).

,START_MINIMUM=SERIAL
,START_MINIMUM=PARALLEL

When AE_SERVERMIN=*ae_servermin* is specified, an optional parameter, which indicates whether WLM will start the minimum number of servers one by one or in parallel. The default is START_MINIMUM=SERIAL.

,START_MINIMUM=SERIAL

The server tasks specified in AE_SERVERMIN will be started one by one. This means the next server will only be started if the previous server has connected to WLM.

,START_MINIMUM=PARALLEL

The server tasks specified in AE_SERVERMIN will be started in parallel. This means WLM will start additional servers even when the previous servers have not connected to WLM.

ABEND codes

None.

Return codes 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 95. Return and Reason Codes for the IWM4SLI Macro

Return Code	Reason Code	Equate Symbol, Meaning, and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0801	**Equate Symbol**: IwmRsnCodeSrbMode **Meaning**: The caller is in SRB mode. **Action**: Avoid requesting this function while in SRB mode.
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: The caller is locked. **Action**: Avoid requesting this function while locked.
8	xxxx080B	**Equate Symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list. **Action**: Check for possible storage overlay.
8	xxxx0824	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: The caller invoked the service but was in 24-bit addressing mode. **Action**: Request this function only when you are in 31-bit addressing mode.
8	xxxx0825	**Equate Symbol**: IwmRsnCodeAscModeNotPrimary **Meaning**: The caller invoked the service but was not in primary ASC mode. **Action**: Avoid requesting this function in this environment.
8	xxxx0827	**Equate Symbol**: IwmRsnCodeRsvdNot0 **Meaning**: Reserved field in parameter list was not zero. **Action**: Check for use of keywords that are not supported by the MVS release on which the program is running.

Table 95. Return and Reason Codes for the IWM4SLI Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx0828	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: The Version number in the parameter list is not valid. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx0840	**Equate Symbol**: IwmRsnCodeServiceNotEnabled **Meaning**: The caller's space connection is not enabled for this service **Action**: Make sure that `SERVER_MANAGER=YES` and `SERVER_TYPE=QUEUE` is specified on the IWM4CON request to enable this service.
8	xxxx0841	**Equate Symbol**: IwmRsnCodeXmemMode **Meaning**: The caller is in cross-memory mode. **Action**: Request this function only when you are not in cross-memory mode.
8	xxxx0842	**Equate Symbol**: IwmRsnCodeNoWLMConnect **Meaning**: The caller's space is not connected to WLM. **Action**: Invoke the IWM4CON macro before invoking this macro.
8	xxxx087D	**Equate Symbol**: IwmRsnCodeBadNumAESrvMax **Meaning**: The server maximum value is incorrect. **Action**: Make sure that the maximum value is greater than the minimum value and greater than the **`parallel_eu`** value.
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error. **Action**: Contact your system programmer.

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
```

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.

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*. WLM provides two methods for computing the weight, which can be selected with the optional input parameter **METHOD**. The default method is PROPORTIONAL, the other one is EQUICPU.

- With PROPORTIONAL, the weight is calculated based on six factors: it is a combination of the three processor weights (CPU weight, ZAAP weight, and ZIIP weight) and their respective consumed service units repartition.
- With EQUICPU, WLM computes the weight by trying to simulate a 100% usage of the system capacity, and determining the capacity of a CPU-only system having equivalent resource consumption.

The CPU, ZAAP, and ZIIP weights are each computed based on 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 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 IWM4RPT, this factor is neutral (=1).
- The third 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 IWMSRSRG. If no health indicator was reported, this factor is also neutral.
- The fourth factor is how much other work with lower importance can be displaced, if it receives more work to handle on this system. With the optional **IL_WEIGHTING** parameter, the caller can set the relative balance between the lower and the higher importance levels.

These four factors are combined to create the output processor WEIGHT as a number.

To make it easier for the caller to determine, how far the weights were influenced by the abnormal terminations and health factors, those values can also be output through the optional parameters **ABNORM_COUNT** and **HEALTH**.

The processor weights are returned through the optional **CPUWEIGHT**, **ZAAPWEIGHT** and **ZIIPWEIGHT** parameters. The respective parts of these weights in the WEIGHT are returned through the optional parameters **CPUPROPORTION**, **ZAAPPROPORTION**, and **ZIIPPROPORTION**.

If there are pre-V1R9 systems in the sysplex, the zAAP and zIIP weights and proportions are automatically set to 0, because pre-V1R9 systems do not have such weights and could not be compared to V1R9 systems. For the same reason, if there are pre-V1R11 systems in the sysplex, only `METHOD=PROPORTIONAL` will be used, even if `METHOD=EQUICPU` is specified.

The WEIGHT is equal to the sum of these three proportion fields. As WLM computes the values with higher precision, and rounds them before output, the WEIGHT actually returned is probably greater than the sum of the returned proportion fields by one or two units.

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.
- 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:

Minimum authorization:	Problem state. Any PSW key
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31- or 64-bit. If in 64-bit addressing mode, code `SYSSTATE AMODE64=YES` before invoking this macro.
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks may be held. FRRs may be established.
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, 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 later.
- 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

The syntax of the IWM4SRSC macro is as follows:

```
►►──┬──────┬──IWM4SRSC──WEIGHT=weight──,STOKEN=stoken──┬──────────────────────────┬──────►
    └─name─┘                                            └─,ABNORM_COUNT=abnorm_count─┘

►──┬────────────────┬──┬──────────────────────┬──┬────────────────────────┬──────────────►
   └─,HEALTH=health─┘  └─,CPUWEIGHT=cpuweight─┘  └─,ZAAPWEIGHT=zaapweight─┘

►──┬────────────────────────┬──┬──────────────────────────────┬──────────────────────────►
   └─,ZIIPWEIGHT=ziipweight─┘  └─,CPUPROPORTION=cpuproportion─┘

►──┬────────────────────────────────┬──┬────────────────────────────────┬────────────────►
   └─,ZAAPPROPORTION=zaapproportion─┘  └─,ZIIPPROPORTION=ziipproportion─┘

                                         ┌─,METHOD=PROPORTIONAL─┐
►──┬──────────────────────────────┬──────┼──────────────────────┼────────────────────────►
   └─,MAXSRVUNITS=maxsrvunits─────┘      └─,METHOD=EQUICPU──────┘

   ┌─,COST_ZAAP_ON_CP=1──────────────┐   ┌─,COST_ZIIP_ON_CP=1──────────────┐
►──┼─────────────────────────────────┼───┼─────────────────────────────────┼─────────────►
   └─,COST_ZAAP_ON_CP=cost_zaap_on_cp┘   └─,COST_ZIIP_ON_CP=cost_ziip_on_cp┘

                                        ┌─,IL_WEIGHTING=0────────────┐
►──┬──────────────────────────────────┬─┼────────────────────────────┼─┬──────────────────┬─►
   └─,MAXEQUIVUNITS=maxequivunits─────┘ └─,IL_WEIGHTING=il_weighting─┘ └─,RETCODE=retcode─┘

                         ┌─,PLISTVER=IMPLIED_VERSION─┐
►──┬──────────────────┬──┼───────────────────────────┼───────────────────────────────────►
   └─,RSNCODE=rsncode─┘  ├─,PLISTVER=MAX─────────────┤
                         ├─,PLISTVER=0───────────────┤
                         ├─,PLISTVER=1───────────────┤
                         ├─,PLISTVER=2───────────────┤
                         ├─,PLISTVER=3───────────────┤
                         └─,PLISTVER=4───────────────┘

   ┌─,MF=S──────────────────────────────────────┐
►──┼────────────────────────────────────────────┼────────────────────────────────────►◄
   │                        ┌─,0D───┐           │
   ├─,MF=(L─,list addr──────┼───────┼─)─────────┤
   │                        └─,attr─┘           │
   │                      ┌─,COMPLETE─┐         │
   └─,MF=(E─,list addr────┴───────────┴─)───────┘
```

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.

To code: Specify the RS-type address, or address in register (2)-(12), of a fullword field.

,COST_ZAAP_ON_CP=*cost_zaap_on_cp*
,COST_ZAAP_ON_CP=1

An optional input parameter, which is used in conjunction with METHOD=EQUICPU. It describes how much more it costs to execute zAAP-eligible work on a CPU instead of on a zAAP processor.

To use the full system capacity, independently of the cost, specify COST_ZAAP_ON_CP=1. With high values of this cost parameter, WLM considers that a system having used up its whole free zAAP capacity should offload less work to the CPU, and gives this system a smaller output weight.

This cost parameter must be in the range from 1 to 100. If the passed value is outside of this range, WLM will instead use the nearest valid integer (1 or 100) as cost parameter. The default is 1.

To code: Specify the RS-type address, or address in register (2)-(12), of a fullword field, or specify a literal decimal value.

,COST_ZIIP_ON_CP=*cost_ziip_on_cp*
,COST_ZIIP_ON_CP=1

An optional input parameter, which is used in conjunction with METHOD=EQUICPU. It describes how much more it costs to execute zIIP-eligible work on a CPU instead of on a zIIP processor.

To use the whole system capacity, independently of the cost, specify COST_ZIIP_ON_CP=1. With high values of this cost parameter, WLM considers that a system having used up its whole free zIIP capacity should offload less work to the CPU, and gives this system a smaller output weight.

This cost parameter must be in the range from 1 to 100. If the passed value is out of this range, WLM will use the nearest valid integer (1 or 100) as cost parameter. The default is 1.

To code: Specify the RS-type address, or address in register (2)-(12), of a fullword field, or specify a literal decimal value.

,CPUPROPORTION=*cpuproportion*

An optional output parameter, which contains the CPU weight part in the calculation of the weight.

To code: Specify the RS-type address, or address in register (2)-(12), of a fullword field.

,CPUWEIGHT=*cpuweight*

An optional output parameter, which contains the weight of how well the server is doing on the CPU.

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.

,IL_WEIGHTING=*il_weighting*
,IL_WEIGHTING=0

An optional input parameter, which controls how WLM evaluates available capacity at importance levels (ILs) lower than the currently selected one. The value of IL_WEIGHTING should be in the range from 0 to 3. If the passed value is out of this range, WLM will use the nearest valid integer (0 or 3) as IL_WEIGHTING instead.

When this parameter is set to 0 (the default value), all free capacities used by levels less important than the current one are evaluated the same. This means that the free capacity below current level is considered to be totally free, and this is equivalent to what WLM did prior to V1R11.

When this parameter is set to 1, free capacity at the lowest ILs is evaluated higher than the current IL, with a weighting growing proportionally to the square root of the IL difference + 1. For example, with a selected IL of 1, free capacity at IL 5 is weighted about 2.236 times more than free capacity at IL 1.

When this parameter is set to 2, free capacity at the lowest ILs is evaluated higher than the current IL, with a weighting growing proportionally to the IL difference + 1. For example, with a selected IL of 1, free capacity at IL 5 is weighted 5 times more than free capacity at IL 1.

When this parameter is set to 3, free capacity at the lowest ILs is weighted more than the current IL, with a weighting growing proportionally to the square of the IL difference + 1. For example, with a selected IL of 1, free capacity at IL 5 is weighted 25 times more than free capacity at IL 1. The default is 0.

To code: Specify the RS-type address, or address in register (2)-(12), of a fullword field, or specify a literal decimal value.

,MAXEQUIVUNITS=*maxequivunits*

An optional output parameter, which contains the maximum equivalent service units across all processing resources used to calculate the mixed weight output.

If METHOD=PROPORTIONAL is specified, MAXEQUIVUNITS is automatically set to 0, since WLM does not compute equivalent CPU service units in this case.

If METHOD=EQUICPU is specified, MAXEQUIVUNITS is always a number in the range from 1 to MAXSRVUNITS. MAXEQUIVUNITS then relates to the mixed weight output only, and MAXSRVUNITS to the CPU, ZAAP and ZIIP weight outputs only.

To code: Specify the RS-type address, or address in register (2)-(12), of a fullword field.

,MAXSRVUNITS=*maxsrvunits*

An optional output parameter, which contains the maximum service units across all processing resources used to calculate the weights.

To code: Specify the RS-type address, or address in register (2)-(12), of a fullword field.

,METHOD=PROPORTIONAL
,METHOD=EQUICPU

An optional parameter, which selects the method for computing the output weights. The default is METHOD=PROPORTIONAL.

,METHOD=PROPORTIONAL
Computes the output weights as a proportion of the three processor type weights.

,METHOD=EQUICPU
Computes a CPU equivalent of the systems before computing the output weights.

To specify METHOD=EQUICPU, all systems in the sysplex must run z/OS V1R11, or higher. Otherwise WLM automatically switches back to METHOD=PROPORTIONAL.

,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
,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**, which supports all parameters except those specifically referenced in higher versions.
- **1**, which supports the following parameters and those from version 0:
  ABNORM_COUNT
  HEALTH
- **2**, which supports the following parameters and those from version 0 and 1:
  CPUPROPORTION
  CPUWEIGHT
  ZAAPPROPORTION
  ZAAPWEIGHT
  ZIIPPROPORTION
  ZIIPWEIGHT
- **3**, which supports the following parameter and those from version 0, 1, and 2:
  MAXSRVUNITS
- **4**, which supports the following parameters and those from version 0, 1, 2, and 3:
  COST_ZAAP_ON_CP
  COST_ZIIP_ON_CP
  IL_WEIGHTING
  MAXEQUIVUNITS
  METHOD

To code: Specify one of the following:

- IMPLIED_VERSION
- MAX
- A decimal value of 0, 1, 2, 3, or 4

,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.

,ZAAPPROPORTION=*zaapproportion*
An optional output parameter, which contains the ZAAP weight part in the calculation of the WEIGHT.

To code: Specify the RS-type address, or address in register (2)-(12), of a fullword field.

,ZAAPWEIGHT=*zaapweight*
An optional output parameter, which contains the weight of how well the server is doing on the ZAAP.

To code: Specify the RS-type address, or address in register (2)-(12), of a fullword field.

,ZIIPPROPORTION=*ziipproportion*
An optional output parameter, which contains the ZIIP weight part in the calculation of the WEIGHT.

To code: Specify the RS-type address, or address in register (2)-(12), of a fullword field.

,ZIIPWEIGHT=*ziipweight*
An optional output parameter, which contains the weight of how well the server is doing on the ZIIP.

To code: Specify the RS-type address, or address in register (2)-(12), of a fullword field.

ABEND codes

None.

Return codes 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 96. Return and Reason Codes for the IWM4SRSC Macro

Return Code	Reason Code	Equate Symbol, Meaning, and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.

Table 96. Return and Reason Codes for the IWM4SRSC Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: The caller is locked. **Action**: Avoid requesting this function while locked.
8	xxxx0807	**Equate Symbol**: IwmRsnCodeBadSTOKEN **Meaning**: Bad STOKEN passed. **Action**: Check for possible storage overlay.
8	xxxx080B	**Equate Symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list. **Action**: Check for possible storage overlay.
8	xxxx0823	**Equate Symbol**: IwmRsnCodeDatoff **Meaning**: The caller invoked the service while DATOFF **Action**: Avoid requesting this function in this environment.
8	xxxx0824	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: The caller invoked the service but was in 24-bit addressing mode. **Action**: Request this function only when you are in 31-bit addressing mode.
8	xxxx0825	**Equate Symbol**: IwmRsnCodeAscModeNotPrimary **Meaning**: The caller invoked the service but was not DAT on Primary ASC mode. **Action**: Avoid requesting this function in this environment.
8	xxxx0828	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: Version number in parameter list or version length field is not valid. **Action**: Check for possible storage overlay of the parameter list.
C	—	**Equate Symbol**: IwmRetCodeEnvError **Meaning**: Environmental error.
C	xxxx0C01	**Equate Symbol**: IwmRsnCodeNoStg **Meaning**: Storage is not available for the request. **Action**: There is a storage shortage. The function may work successfully at a later time.

Table 96. Return and Reason Codes for the IWM4SRSC Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error. **Action**: 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
```

IWM4SSL — Select 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.

Note: This service was previously called IWMSSEL for 31-bit addressing only (see "IWMSSEL — Select a request from a caller's work manager queue" on page 1047).

Environment

The requirements for the caller are:

Minimum authorization:	Problem state. Any PSW key
Dispatchable unit mode:	Task
Cross memory mode:	PASN=HASN=SASN
AMODE:	31- or 64-bit. If in 64-bit addressing mode, code `SYSSTATE AMODE64=YES` before invoking this macro.
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks may be held.
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, 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.

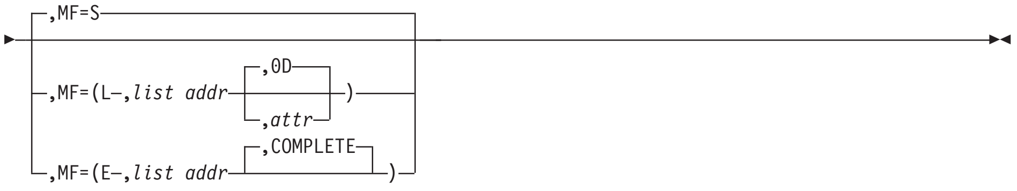
Performance implications

None.

Syntax

The syntax of the IWM4SSL macro is as follows:

```
►►──┬──────┬──IWM4SSL──USERDATA=userdata──,WLMEUTKN=wlmeutkn──────────────────────►
    └─name─┘

►──┬──────────────────────────────┬──┬──────────────────────────────┬──┬────────────────────┬──►
   └─,SERVER_TOKEN=server_token───┘  └─,REGION_TOKEN=region_token───┘  └─,RETCODE=retcode───┘

                               ┌─,PLISTVER=IMPLIED_VERSION─┐
►──┬────────────────────┬──────┼───────────────────────────┼──────────────────────►
   └─,RSNCODE=rsncode───┘      ├─,PLISTVER=MAX─────────────┤
                               └─,PLISTVER=0───────────────┘
```



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 vitual 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 codes 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 97. Return and Reason Codes for the IWM4SSL Macro

Return Code	Reason Code	Equate Symbol, Meaning, and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0801	**Equate Symbol**: IwmRsnCodeSrbMode **Meaning**: The caller is in SRB mode. **Action**: Avoid requesting this function while in SRB mode.
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: Caller is locked. **Action**: Avoid requesting this function while locked.
8	xxxx080B	**Equate Symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list. **Action**: Check for possible storage overlay.
8	xxxx0823	**Equate Symbol**: IwmRsnCodeDatoff **Meaning**: The caller invoked the service while DATOFF **Action**: Avoid requesting this function in this environment.

Table 97. Return and Reason Codes for the IWM4SSL Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx0824	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: The caller invoked the service but was in 24-bit addressing mode. **Action**: Request this function only when you are in 31-bit addressing mode.
8	xxxx0825	**Equate Symbol**: IwmRsnCodeAscModeNotPrimary **Meaning**: The caller invoked the service but was not in primary ASC mode. **Action**: Avoid requesting this function in this environment.
8	xxxx0827	**Equate Symbol**: IwmRsnCodeRsvdNot0 **Meaning**: Reserved field in parameter list was non-zero. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx0828	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: Version number in parameter list is not valid. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx0840	**Equate Symbol**: IwmRsnCodeServiceNotEnabled **Meaning**: The caller's space connection is not enabled for this service **Action**: Make sure that SERVER_MANAGER=YES is specified on the IWM4CON request to enable this service.
8	xxxx0841	**Equate Symbol**: IwmRsnCodeXmemMode **Meaning**: The caller is in cross-memory mode. **Action**: Request this function only when you are not in cross-memory mode.
8	xxxx0842	**Equate Symbol**: IwmRsnCodeNoWLMConnect **Meaning**: The caller's space is not connected to WLM. **Action**: Invoke the IWM4CON macro before invoking this macro.
8	xxxx0854	**Equate Symbol**: IwmRsnCodeTooManySelect **Meaning**: The caller is attempting to select more work units than it has tasks to execute the work. **Action**: Wait until an execution task has issued IWM4STEN before attempting to select more work units.
8	xxxx0864	**Equate Symbol**: IwmRsnCodeSecondaryWorkExists **Meaning**: There are secondary work requests queued to this server task. The caller was expected to process them using IWM4SSM before calling IWM4SSL. **Action**: Select all secondary work requests before issuing IWM4SSL.
C	—	**Equate Symbol**: IwmRetCodeEnvError **Meaning**: Environmental error.

Table 97. Return and Reason Codes for the IWM4SSL Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
C	xxxx0C01	**Equate Symbol**: IwmRsnCodeNoStg **Meaning**: Storage is not available for the request. **Action**: Caller must disconnect by invoking the IWM4DIS macro.
C	xxxx0C14	**Equate Symbol**: IwmRsnCodeNoWorkShutDown **Meaning**: No work selected. Caller is to shutdown. **Action**: The caller must disconnect by invoking the IWM4DIS macro.
C	xxxx0C3B	**Equate Symbol**: IwmRsnCodeStopTask **Meaning**: WLM stopped the server instance. **Action**: Calling task must shutdown, but server address space must remain active.
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error. **Action**: Contact your system programmer.

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
```

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:

1. 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.
2. 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.
3. 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.
4. After the last secondary work request has been processed, the caller invokes the IWM4STEN macro to remove the environment created by IWM4STBG.
5. 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.

Note: This service was previously called IWMSSEM for 31-bit addressing only (see "IWMSSEM — WLM server select secondary service" on page 1055).

Environment

The requirements for the caller are:

Minimum authorization:	Problem state. Any PSW key
Dispatchable unit mode:	Task
Cross memory mode:	PASN=HASN=SASN
AMODE:	31- or 64-bit. If in 64-bit addressing mode, code `SYSSTATE AMODE64=YES` before invoking this macro.
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks may be held.
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, IWMRSNCODE_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

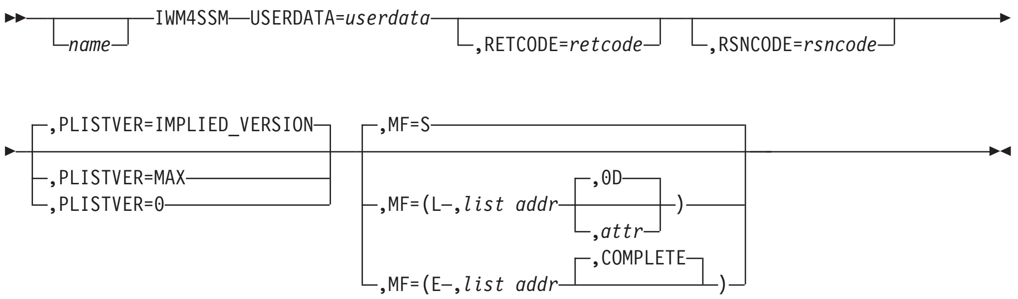
Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

Performance implications

None.

Syntax

The syntax of the IWM4SSM macro is as follows:



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*,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 IWM4QIN.

To code: Specify the RS-type address, or address in register (2)-(12), of a 16-character field.

ABEND codes

None.

Return codes 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 98. Return and Reason Codes for the IWM4SSM Macro

Return Code	Reason Code	Equate Symbol, Meaning, and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0801	**Equate Symbol**: IwmRsnCodeSrbMode **Meaning**: Caller is in SRB mode. **Action**: Avoid requesting this function while in SRB mode.
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: The caller is locked. **Action**: Avoid requesting this function while locked.
8	xxxx080B	**Equate Symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list. **Action**: Check for possible storage overlay.
8	xxxx0824	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: The caller invoked the service but was in 24-bit addressing mode. **Action**: Request this function only when you are in 31-bit addressing mode.
8	xxxx0825	**Equate Symbol**: IwmRsnCodeAscModeNotPrimary **Meaning**: The caller invoked the service but was not in primary ASC mode. **Action**: Avoid requesting this function in this environment.
8	xxxx0827	**Equate Symbol**: IwmRsnCodeRsvdNot0 **Meaning**: Reserved field in parameter list was not zero. **Action**: Check for use of keywords that are not supported by the MVS release on which the program is running.
8	xxxx0828	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: Version number in parameter list is not valid. **Action**: Check for possible storage overlay of the parameter list.

Table 98. Return and Reason Codes for the IWM4SSM Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx0840	**Equate Symbol**: IwmRsnCodeServiceNotEnabled **Meaning**: The caller's space connection is not enabled for this service **Action**: Make sure that SERVER_MANAGER=YES is specified on the IWM4CON request to enable this service.
8	xxxx0841	**Equate Symbol**: IwmRsnCodeXmemMode **Meaning**: The caller is in cross-memory mode. **Action**: Request this function only when you are not in cross-memory mode.
8	xxxx0842	**Equate Symbol**: IwmRsnCodeNoWLMConnect **Meaning**: The caller's space is not connected to WLM. **Action**: Invoke the IWM4CON macro before invoking this macro.
8	xxxx0862	**Equate Symbol**: IwmRsnCodeNoPriorSelect **Meaning**: The caller has not previously selected work using the IWM4SSL macro. **Action**: Invoke the IWM4SSL macro before invoking this macro.
8	xxxx0863	**Equate Symbol**: IwmRsnCodeNoExecEnv **Meaning**: The caller has not established an execution environment using IWM4STBG. **Action**: Invoke the IWM4STBG macro before invoking this macro.
C	—	**Equate Symbol**: IwmRetCodeEnvError **Meaning**: Environmental error.
C	xxxx0C14	**Equate Symbol**: IwmRsnCodeNoWorkShutDown **Meaning**: No work selected. Caller is to shutdown. **Action**: The caller must disconnect by invoking the IW DISC macro.
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error. **Action**: 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
```

IWM4STBG — WLM begin server transaction service

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.

Note: This service was previously called IWMSTBGN for 31-bit addressing only (see “IWMSTBGN — Begin a request from a caller's work manager queue” on page 1062).

Environment

The requirements for the caller are:

Minimum authorization:	Problem state. Any PSW key
Dispatchable unit mode:	Task
Cross memory mode:	PASN=HASN=SASN
AMODE:	31- or 64-bit. If in 64-bit addressing mode, code `SYSSTATE AMODE64=YES` before invoking this macro.
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks may be held.
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, 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:

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

```
►►─┬──────┬─b──IWM4STBG──b──WLMEUTKN=wlmeutkn─┬─────────────────┬──────────►
   └─name─┘                                   └─,ETOKEN=etoken──┘

   ┌─,SUBTASKS=NO──┐
►──┼───────────────┼──┬──────────────────┬──┬──────────────────┬──────────►
   └─,SUBTASKS=YES─┘  └─,RETCODE=retcode─┘  └─,RSNCODE=rsncode─┘
   ┌─,PLISTVER=IMPLIED_VERSION─┐  ┌─,MF=S─────────────────────────────────┐
►──┤                           ├──┼───────────────────────────────────────┼──────►◄
   ├─,PLISTVER=MAX─────────────┤  │                  ┌─,0D───┐            │
   └─,PLISTVER=0───────────────┘  ├─,MF=(L─,list addr┼───────┼─)──────────┤
                                  │                  └─,attr─┘            │
                                  │                  ┌─,COMPLETE─┐        │
                                  └─,MF=(E─,list addr┴───────────┴─)──────┘
```

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 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

,RETCODE=*retcode*

An optional output parameter into which the return code is to be copied from GPR 15. If you specify 15, GPR15, REG15, or R15 (within or without parentheses), the value will be left in GPR 15.

To code: Specify the RS-type address of a fullword field, or register (2)-(12) or (15), (GPR15), (REG15), or (R15).

,RSNCODE=*rsncode*

An optional output parameter into which the reason code is to be copied from GPR 0. If you specify 0, 00, GPR0, GPR00, REG0, REG00, or R0 (within or without parentheses), the value will be left in GPR 0.

To code: Specify the RS-type address of a fullword field, or register (0) or (2)-(12), (00), (GPR0), (GPR00), REG0), (REG00), or (R0).

,SUBTASKS=NO
,SUBTASKS=YES

An optional parameter, which specifies if subtasks of the requesting task are also to be processed. The default is SUBTASKS=NO.

,SUBTASKS=NO

specifies that subtasks of the requesting task are not to be processed.

,SUBTASKS=YES

specifies that subtasks of the requesting task that are not already joined to an enclave are to be joined to the enclave identified by this invocation's ETOKEN parameter. When a currently-dispatched subtask is joined to the enclave, its CPU time for that dispatch is associated with the enclave rather than the address space. When the subtask is removed from the enclave, if it is currently dispatched, its CPU time for that dispatch is associated with the address space rather than the enclave.

If SYSEVENT REQSRMST does not indicate, via bit SRMSTSTS being on, that this function is available, this is treated as SUBTASKS=NO.

When SUBTASKS=YES is in effect, this task's corresponding IWM4STEN will also perform leave processing upon any subtasks that are implicitly associated with the enclave. This includes subtasks that were joined to the enclave due to this task's IWM4STBG processing as well as subtasks that were joined to the enclave by ATTACH processing.

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 codes and reason codes

When the IWM4STBG 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 99. Return and Reason Codes for the IWM4STBG Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.
4	xxxx041F	**Equate Symbol**: IwmRsnCodeExecEnvChanged **Meaning**: The execution environment has changed while the requested function is in progress. **Action**: None required.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0801	**Equate Symbol**: IwmRsnCodeSrbMode **Meaning**: Caller is in SRB mode. **Action**: Avoid requesting this function while in SRB mode.

Table 99. Return and Reason Codes for the IWM4STBG Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: Caller is locked. **Action**: Avoid requesting this function while locked.
8	xxxx080B	**Equate Symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list. **Action**: Check for possible storage overlay.
8	xxxx0810	**Equate Symbol**: IwmRsnCodeEutFrr **Meaning**: Caller has EUT FRR established. **Action**: Avoid requesting this function with an EUT FRR set.
8	xxxx0823	**Equate Symbol**: IwmRsnCodeDatoff **Meaning**: Caller invoked service while DATOFF **Action**: Avoid requesting this function in this environment.
8	xxxx0824	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: Caller invoked service but was in 24 bit addressing mode. **Action**: Request this function only when you are in 31 bit addressing mode.
8	xxxx0825	**Equate Symbol**: IwmRsnCodeAscModeNotPrimary **Meaning**: Caller invoked service but was not DAT on Primary ASC mode. **Action**: Avoid requesting this function in this environment.
8	xxxx0826	**Equate Symbol**: IwmRsnCodeTaskTerm **Meaning**: Caller invoked service while task termination is in progress for the TCB associated with the owner. **Action**: Avoid requesting this function while task termination is in progress.
8	xxxx0828	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: Version number in parameter list is not valid or version length field is not valid. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx083A	**Equate Symbol**: IwmRsnCodeBadEnclave **Meaning**: Enclave token does not pass verification. **Action**: Check for possible storage overlay of the enclave token, or asynchronous events which may have deleted the enclave.

Table 99. Return and Reason Codes for the IWM4STBG Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0840	**Equate Symbol**: IwmRsnCodeServiceNotEnabled **Meaning**: Caller's space connection is not enabled for this service **Action**: Make sure that SERVER_MANAGER=YES is specified on the IWM4CON request to enable this service.
8	xxxx0841	**Equate Symbol**: IwmRsnCodeXmemMode **Meaning**: Caller is in cross-memory mode. **Action**: Request this function only when you are not in cross-memory mode.
8	xxxx0842	**Equate Symbol**: IwmRsnCodeNoWLMConnect **Meaning**: Caller's space is not connected to WLM. **Action**: Invoke the IWM4CON macro before invoking this macro.
8	xxxx0850	**Equate Symbol**: IwmRsnCodeBeginEnvOutstanding **Meaning**: Caller is already operating under an outstanding Begin environment. **Action**: Avoid requesting this function in this environment.
8	xxxx0851	**Equate Symbol**: IwmRsnCodeSecEnvOutstanding **Meaning**: Caller is already operating under an outstanding security environment. **Action**: Avoid requesting this function while there is a TCB level security environment outstanding.
8	xxxx0852	**Equate Symbol**: IwmRsnCodeExecTokenNotCorrect **Meaning**: The execution unit token does not identify a previously selected work unit. **Action**: Verify that you have coded the WLMEUTKN parameter correctly.
8	xxxx0857	**Equate Symbol**: IwmRsnCodeAlreadyInEnclave **Meaning**: Current dispatchable workunit is already in an enclave. **Action**: Avoid requesting this function while the caller is already in an enclave.
8	xxxx085A	**Equate Symbol**: IwmRsnCodeSelectedWorkActive **Meaning**: The selected work element associated with the input execution unit token is already in execution. **Action**: 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	—	**Equate Symbol**: IwmRetCodeEnvError **Meaning**: Environmental error.

Table 99. Return and Reason Codes for the IWM4STBG Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
C	xxxx0C17	**Equate Symbol**: IwmRsnCodeSecEnvCreateFailed **Meaning**: A user security environment cannot be created. **Action**: Verify that the userid is defined to RACF or check the SAF installation exit routine to enable the function.
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error. **Action**: 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:

```
        IWM4STBG WLMEUTKN=WLMEUTKN,RETCODE=RC,RSNCODE=RSN
*
* Storage areas
*
WLMEUTKN DS    CL8              Contains the execution unit
*                               token that was returned by
*                               IWM4SSL
RC       DS    F                Return code
RSN      DS    F                Reason code
```

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.

Note: This service was previously called IWMSTEND for 31-bit addressing only (see "IWMSTEND — End a request from a caller's work manager queue" on page 1070).

Environment

The requirements for the caller are:

Minimum authorization:	Problem state. Any PSW key
Dispatchable unit mode:	Task
Cross memory mode:	PASN=HASN=SASN
AMODE:	31- and 64-bit. If in 64-bit addressing mode, code `SYSSTATE AMODE64=YES` before invoking this macro.
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks may be held.
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, 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

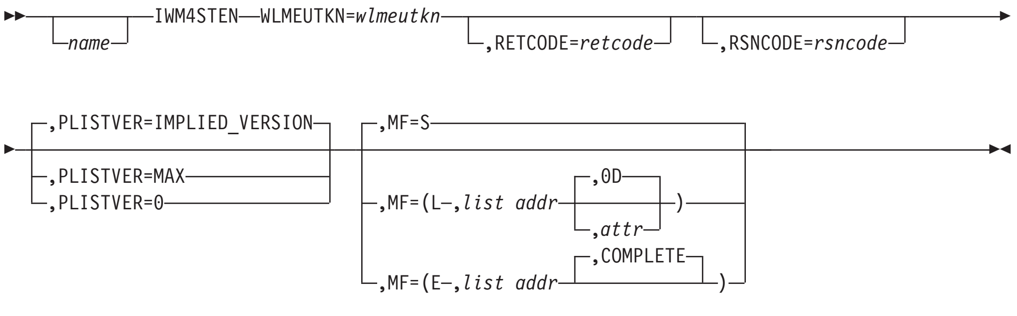
Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

Performance implications

None.

Syntax

The syntax of the IWM4STEN macro is as follows:



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

,RETCODE=*retcode*
An optional output parameter into which the return code is to be copied from GPR 15. If you specify 15, GPR15, REG15, or R15 (with or without parentheses), the value will be left in GPR 15.

To code: Specify the RS-type address of a fullword field, or register (2)-(12) or (15), (GPR15), (REG15), or (R15).

,RSNCODE=*rsncode*
An optional output parameter into which the reason code is to be copied from GPR 0. If you specify 0, 00, GPR0, GPR00, REG0, REG00, or R0 (with or without parentheses), the value will be left in GPR 0.

To code: Specify the RS-type address of a fullword field, or register (0) or (2)-(12), (00), (GPR0), (GPR00), REG0), (REG00), or (R0).

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 codes 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 100. Return and Reason Codes for the IWM4STEN Macro

Return Code	Return Code	Equate Symbol, Meaning, and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.
4	xxxx041C	**Equate Symbol**: IwmRsnCodeNotEnclave **Meaning**: The current dispatchable work unit is not associated with an enclave. **Action**: None required.

Table 100. Return and Reason Codes for the IWM4STEN Macro (continued)

Return Code	Return Code	Equate Symbol, Meaning, and Action
4	xxxx041F	**Equate Symbol**: IwmRsnCodeExecEnvChanged **Meaning**: The execution environment has changed while the requested function is in progress. **Action**: None required.
4	xxxx042F	**Equate Symbol**: IwmRsnCodeSecondaryWorkDeleted **Meaning**: 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. **Action**: Select all secondary work requests before issuing IWM4STEN.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0801	**Equate Symbol**: IwmRsnCodeSrbMode **Meaning**: Caller is in SRB mode. **Action**: Avoid requesting this function while in SRB mode.
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: The caller is locked. **Action**: Avoid requesting this function while locked.
8	xxxx080B	**Equate Symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list. **Action**: Check for possible storage overlay.
8	xxxx0810	**Equate Symbol**: IwmRsnCodeEutFrr **Meaning**: The caller has EUT FRR established. **Action**: Avoid requesting this function with an EUT FRR set.
8	xxxx0823	**Equate Symbol**: IwmRsnCodeDatoff **Meaning**: The caller invoked the service while DATOFF **Action**: Avoid requesting this function in this environment.
8	xxxx0824	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: The caller invoked the service but was in 24-bit addressing mode. **Action**: Request this function only when you are in 31-bit addressing mode.
8	xxxx0825	**Equate Symbol**: IwmRsnCodeAscModeNotPrimary **Meaning**: The caller invoked the service but was not DAT on Primary ASC mode. **Action**: Avoid requesting this function in this environment.

Table 100. Return and Reason Codes for the IWM4STEN Macro (continued)

Return Code	Return Code	Equate Symbol, Meaning, and Action
8	xxxx0828	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: Version number in parameter list is not valid or version length field is not valid. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx0841	**Equate Symbol**: IwmRsnCodeXmemMode **Meaning**: The caller invoked the service but was in cross-memory mode. **Action**: Avoid requesting this function in cross-memory mode.
8	xxxx084F	**Equate Symbol**: IwmRsnCodeWrongExecToken **Meaning**: Current dispatchable work unit is not associated with the input execution unit token. **Action**: Check for possible storage overlay of the execution unit token.
8	xxxx0859	**Equate Symbol**: IwmRsnCodeEnclaveSubTaskExists **Meaning**: The current dispatchable work unit has residual subtasks propagated to the enclave which are still associated with the enclave. The operation (IWM4STBG) that associated this work unit with the enclave did not specify SUBTASKS=YES. **Action**: Avoid requesting this function in this environment.
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error. **Action**: Contact your system programmer.

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
```

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.

Note: This service was previously called IWMTAFF for 31-bit addressing only (see "IWMTAFF — WLM temporal affinity service" on page 1076).

Environment

The requirements for the caller are:

Minimum authorization:	Problem state. Any PSW key
Dispatchable unit mode:	Task
Cross memory mode:	PASN=HASN=SASN
AMODE:	31- and 64-bit. If in 64-bit addressing mode, code `SYSSTATE AMODE64=YES` before invoking this macro.
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	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.

Restrictions

1. This macro may not be used during task/address space termination.
2. 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

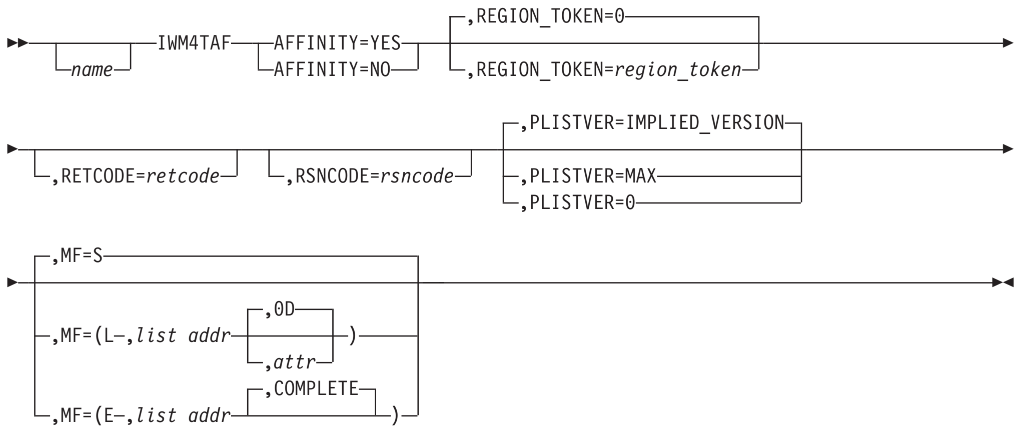
Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

Performance implications

None.

Syntax

The syntax of the IWM4TAF macro is as follows:



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***,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

,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 IWM4CON and 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 codes 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 101. Return and Reason Codes for the IWM4TAF Macro

Return Code	Reason Code	Equate Symbol, Meaning, and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted. **Action**: None required.
4	xxxx0439	**Equate Symbol**: IwmRsnCodeNoAffinityFound **Meaning**: 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. **Action**: 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	**Equate Symbol**: IwmRsnCodeRegionNotFound **Meaning**: The region token does not identify a valid server region. **Action**: Please specify the correct region token.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0801	**Equate Symbol**: IwmRsnCodeSrbMode **Meaning**: Caller is in SRB mode. **Action**: Avoid requesting this function while in SRB mode.
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: Caller is locked. **Action**: Avoid requesting this function while locked.
8	xxxx080B	**Equate Symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list. **Action**: Check for possible storage overlay.

Table 101. Return and Reason Codes for the IWM4TAF Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx0824	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: The caller invoked the service but was in 24-bit addressing mode. **Action**: Request this function only when you are in 31-bit addressing mode.
8	xxxx0825	**Equate Symbol**: IwmRsnCodeAscModeNotPrimary **Meaning**: The caller invoked the service but was not in primary ASC mode. **Action**: Avoid requesting this function in this environment.
8	xxxx0827	**Equate Symbol**: IwmRsnCodeRsvdNot0 **Meaning**: Reserved field in parameter list was not zero. **Action**: Check for use of keywords that are not supported by the MVS release on which the program is running.
8	xxxx0828	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: Version number in parameter list is not valid. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx0840	**Equate Symbol**: IwmRsnCodeServiceNotEnabled **Meaning**: The caller's space connection is not enabled for this service **Action**: Make sure that SERVER_MANAGER=YES and SERVER_TYPE=QUEUE is specified on the IWM4CON request to enable this service.
8	xxxx0841	**Equate Symbol**: IwmRsnCodeXmemMode **Meaning**: Caller is in cross-memory mode. **Action**: Request this function only when you are not in cross-memory mode.
8	xxxx0842	**Equate Symbol**: IwmRsnCodeNoWLMConnect **Meaning**: The caller's space is not connected to WLM. **Action**: Invoke the IWM4CON macro before invoking this macro.
8	xxxx084D	**Equate Symbol**: IwmRsnCodeNotAuthConnect **Meaning**: 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. **Action**: Avoid requesting this function in this environment.
8	xxxx08B1	**Equate Symbol**: IwmRsnCodeTooManyTempAff **Meaning**: No more than 2 GB temporal affinities supported. **Action**: Avoid requesting more than 2 GB temporal affinities to an address space.

Table 101. Return and Reason Codes for the IWM4TAF Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error. **Action**: Contact your system programmer.

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 102 lists the SMF record type 99 action codes and descriptions.

Table 102. SMF record type 99 action codes

Action code number	Equate symbol	Description
1	STA_RECOVERY_RETRY	Retry.
2	STA_RECOVERY_PERC	Percolation.
3	STA_RECOVERY_REDRIVE_SET	Tell WLM to set to same policy again.
10	RA_AUXP_DEC_MPL	Resource adjustment, too much auxiliary storage paging, decrease mpl.
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.
111	RA_UU_NO_RECEIVER	Resource adjustment, underutilized, no action, no receiver found.
112	RA_UU_DISP_NOT_QUEUE_MANAGED	Resource adjustment, underutilized, no action, internal service period not queue-managed.
113	RA_UU_DISP_NOT_ELIGIBLE	Resource adjustment, underutilized, no action, internal service period not eligible.
114	RA_UU_DISP_PENDING	Resource adjustment, underutilized, no action, internal service period not eligible, pending actions.
115	RA_UU_DISP_NOT_BOUND	Resource adjustment, underutilized, no action, internal service period not eligible, no address space bound.
116	RA_UU_DISP_SPACE_NOT_AVAILABLE	Resource adjustment, underutilized, no action, internal service period not eligible, close to limit on the number of address spaces that can be started, or there are not enough address spaces available.
117	RA_UU_DISP_CANT_START_SPACE	Resource adjustment, underutilized, no action, internal service period is not eligible, application environment cannot start servers.
118	RA_UU_DISP_NO_HISTORY	Resource adjustment, underutilized, no action, internal service period is not eligible, there were no queued requests last interval.

Table 102. SMF record type 99 action codes (continued)

Action code number	Equate symbol	Description
119	RA_UU_DISP_INSUFFICIENT_STORAGE	Resource adjustment, underutilized, no action, internal service period is not eligible, there is no discretionary storage for another address space.
120	RA_UU_DISP_SUFFICIENT_SERVERS	Resource adjustment, underutilized, no action, internal service period is not eligible, there are more servers than needed.
121	RA_UU_DISP_OUTREADY_SERVERS	Resource adjustment, underutilized, no action, internal service period is not eligible, there are out ready servers.
122	RA_UU_DISP_CONTENTION	Resource adjustment, underutilized, no action, additional server instances cannot be added without unacceptable increase in contention.
123	RA_UU_DISP_TOO_MANY_IDLE	Resource adjustment, underutilized, no action, the queue already has many idle servers.
125	RA_UU_DISP_NOT_BEST	Resource adjustment, underutilized, no action, this period is not the best.
126	RA_UU_DISP_NOT_BEST_NOT_AVAIL	Resource adjustment, underutilized, no action, short term processor is not available.
127	RA_UU_DISP_NOT_BEST_DELAY_SAMPLES	Resource adjustment, underutilized, no action, projected delay samples are not the best.
130	SWAP_DETECTED_WAIT	Detected wait swap.
140	SWAP_EXCHANGE	Exchange swap.
150	SWAP_LONG_WAIT	Long wait swap.
160	SWAP_UNILATERAL	Unilateral swap.
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.
240	PA_GREC_CAND	Policy adjustment, goal receiver candidate selected.

Table 102. SMF record type 99 action codes (continued)

Action code number	Equate symbol	Description
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 mulli-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.
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_BLKR_MOVED	Policy adjustment, CPU critical, blocker has been moved to a new priority.

Table 102. SMF record type 99 action codes (continued)

Action code number	Equate symbol	Description
529	PA_PCC_BLKR_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.
531	PA_PCC_DON_VIOLTN	Policy adjustment, moving the donor to the receivers priority violates CPU critical rules.
532	PA_PCC_BLKR_IS_DON	Policy adjustment, cannot move the blocker up because it is the donor.
533	PA_PCC_BLKR_IS_REC	Policy adjustment, cannot move the blocker down because it is the receiver.
534	PA_PCC_BLKR_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.
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.

Table 102. SMF record type 99 action codes (continued)

Action code number	Equate symbol	Description
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.
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_BLKR	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_BLKR	Policy adjustment, increase priority for blocker period.
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.

Table 102. SMF record type 99 action codes (continued)

Action code number	Equate symbol	Description
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.
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.
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.

Table 102. SMF record type 99 action codes (continued)

Action code number	Equate symbol	Description
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 untit queueing delay.
1000	PA_TA_EA_PASS_NO	Policy adjustment, tuning alias, efficiency adjustment, pass number.
1900	PA_0C9_suppressed	Policy adjustment.
2010	PA_DEC_PSI_TAR	Policy adjustment, decrease period protective processor storage target for this resource period.
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 thatn 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.
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.

Table 102. SMF record type 99 action codes (continued)

Action code number	Equate symbol	Description
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.
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.
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.

Table 102. SMF record type 99 action codes (continued)

Action code number	Equate symbol	Description
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.
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.
3050	TDH_DEC_CSI_TAR	Time driven housekeeping, decrease common area protective processor storage target.

Table 102. SMF record type 99 action codes (continued)

Action code number	Equate symbol	Description
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.
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 102. SMF record type 99 action codes (continued)

Action code number	Equate symbol	Description
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.
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.

Table 102. SMF record type 99 action codes (continued)

Action code number	Equate symbol	Description
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.
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.
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 reciever 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.

Table 102. SMF record type 99 action codes (continued)

Action code number	Equate symbol	Description
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.
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.
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.

Table 102. SMF record type 99 action codes (continued)

Action code number	Equate symbol	Description
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.
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.
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.

Table 102. SMF record type 99 action codes (continued)

Action code number	Equate symbol	Description
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 increamented 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.
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.
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.

Table 102. SMF record type 99 action codes (continued)

Action code number	Equate symbol	Description
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.
5530	IOV_SUBSYS	Sysevent IoViolat, I/O subsystem trace.
5531	IOV_GREC_SYS	Sysevent IoViolat, goal receiver, sysplex level trace.
5532	IOV_GREC_LOC	Sysevent IoViolat, goal receiver, local system trace.
5533	IOV_GREC_REM	Sysevent IoViolat, goal receiver, remote system trace.
5534	IOV_GREC_NETV_SYS	Sysevent IoViolat, goal receiver, rejected for net value, sysplex level trace.
5535	IOV_GREC_NETV_LOC	Sysevent IoViolat, goal receiver, rejected for net value, local system trace.
5536	IOV_GREC_NETV_REM	Sysevent IoViolat, goal receiver, rejected for net value, remote system trace.
5537	IOV_GDON_NETV_SYS	Sysevent IoViolat, goal donor, rejected for net value, sysplex level trace.
5538	IOV_GDON_NETV_LOC	Sysevent IoViolat, goal donor, rejected for net value, local system trace.
5539	IOV_GDON_NETV_REM	Sysevent IoViolat, goal donor, rejected for net value, remote system trace.
5540	IOV_RREC_NETV	Sysevent IoViolat, resource receiver, rejected for net value.
5541	IOV_RDON_NETV	Sysevent IoViolat, resource donor, rejected for net value.
5542	IOV_GDON_MIMP_SYS	Sysevent IoViolat, goal donor, rejected for net value, sysplex level trace.
5543	IOV_GDON_MIMP_LOC	Sysevent IoViolat, goal donor, most impacted, local system trace.
5544	IOV_GDON_MIMP_REM	Sysevent IoViolat, goal donor, most impacted, remote system trace.
5545	IOV_NORECEIVER	Sysevent IoViolat, no receiver found.
5546	IOV_NODONOR	Sysevent IoViolat, no donor found.
5547	IOV_RC	Sysevent IoViolat, return code.
5548	IOV_IREC_SYS	Sysevent IoViolat, internal receiver, sysplex level trace.
5549	IOV_IREC_LOC	Sysevent IoViolat, internal receiver, local system trace.
5550	IOV_IREC_REM	Sysevent IoViolat, internal receiver, remote system trace.
5551	IOV_IDON_SYS	Sysevent IoViolat, internal donor, sysplex level trace.
5552	IOV_IDON_LOC	Sysevent IoViolat, internal donor, local system trace.
5553	IOV_IDON_REM	Sysevent IoViolat, internal donor, remote system trace.
5554	IOV_DEC_TAR	Sysevent IoViolat, decrease I/O velocity target for subsystem whose target is violated.

Table 102. SMF record type 99 action codes (continued)

Action code number	Equate symbol	Description
5555	IOV_BAD_SUBSYS	Sysevent IoViolat, I/O subsystem trace, bad subsystem index or velocity.
5556	IOV_RDON_MIMP	Sysevent IoViolat, resource donor, most impacted.
5557	IOV_ADD_CHPID	Sysevent IoViolat, add chpid proposal.
5558	IOV_DELETE_CHPID	Sysevent IoViolat, delete chpid proposal.
5559	IOV_AVAILABILITY	Sysevent IoViolat, 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.
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 102. SMF record type 99 action codes (continued)

Action code number	Equate symbol	Description
7590	WSM_END_SWAPIN	Working set management, end a's frames to b swapping interval.
7600	WSM_END_TRYLRU	Working set management, end trylru 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 porcessor 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 trylru 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_DRGROUP_ADD	Dynamic resource group created.
8040	PA_DRGROUP_DELETE	Dynamic resource group deleted.
8050	PA_DRGROUP_MRK_DEL	Dynamic resource group marked for deletion.
8055	PA_DRGROUP_MRK_ALL	All dynamic resource groups marked for deletion.
8060	PA_DRGROUP_EXCHG	Dynamic resource group exchanged.
8070	PA_DRGROUP_MAX_INC	Dynamic resource group maximum service.
8075	PA_DRGROUP_MAX_NI	Dynamic resource group maximum service rate not increased.
8080	PA_DRGROUP_MAX_DEC	Dynamic resource group maximum service rate decreased.
8090	PA_DRGROUP_ADD_INT	Dynamic resource group add initiators.
8095	PA_DRGROUP_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.

Table 102. SMF record type 99 action codes (continued)

Action code number	Equate symbol	Description
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.
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.
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.

Table 102. SMF record type 99 action codes (continued)

Action code number	Equate symbol	Description
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.
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.
9002	PA_BP_DON_PER_VAL	Buffer pool owning period, asess 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.

Table 102. SMF record type 99 action codes (continued)

Action code number	Equate symbol	Description
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.
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 auxiliary 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.
9246	PA_QMPL_NA_NOSYS	No qmpl actions taken because there is no system to start initiators.
9247	PA_QMPL_LIM_GSMAX	Qmpl increase limited because resource group maximum reached.
9250	PA_INC_QMPL_GR	Policy adjustment, increase qmpl for queue servers, goal receiver.

Table 102. SMF record type 99 action codes (continued)

Action code number	Equate symbol	Description
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.
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 102. SMF record type 99 action codes (continued)

Action code number	Equate symbol	Description
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.
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.

Table 102. SMF record type 99 action codes (continued)

Action code number	Equate symbol	Description
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.
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.
9348	PA_LMP_SKIPPED	LPAR weight management was skipped because one or several required conditions were not met.
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.
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.

Table 102. SMF record type 99 action codes (continued)

Action code number	Equate symbol	Description
9998	PA_PAS_RDON_SAMPD	TEMP - resource don using & delay delta.

Appendix B. Application validation reason codes

Table 103 lists the section, reason code, offset, and description for SERVD application validation reason codes.

Table 103. SERVD application 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, SVDCR, 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.

Table 103. SERVD application validation reason codes (continued)

Section	Reason	Offset	Description
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	Subsystem type (SVAEA_SUBSYSTEM_TYPE) is not specified or is incorrect.
	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.

Table 103. SERVD application validation reason codes (continued)

Section	Reason	Offset	Description
SVDEFHDR	0100	SVDEF	The service definition is at a higher level than the WLM code running on this system
	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 (SVDEFCNS) does not match compiled size
	010B	SVDEF	Only checked if releases match - Period entry size (SVDEFCPS) does not match compiled size
	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
	010D	SVDEF	It was tried to install a service definition with a level less than 23 but with more than 999 report classes.
	1C0F	entry	Classification rule was found that has indicators on in the SVDCR reserved for future qualifier bytes. The reserved for future qualifier bytes are in SVDCRRQT_BYTE4.
	1C02	entry	Group value specified (SVDCRRGI = '1'B) for a rule that does not support the group indicator. The qualifier types that do not support groups are: priority, zEnterprise service class.

Table 103. SERVD application validation reason codes (continued)

Section	Reason	Offset	Description
SVDEFHDR (continued)	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), process name (SVDCRRQT_PROCESS_NAME), scheduling environment (SVDCRRQT_SCHEDULING_ENVIRONMENT), zEnterprise Service Class (SVDCRRQT_EWLM_SCLASS), Client UserId (SVDCRRQT_CLIENT_USERID), Client Workstation Name (SVDCRRQT_CLIENT_WORKSTATION_NAME), or Client IP Address (SVDCRRQT_CLIENT_IP_ADDRESS), Package Name (SVDCRRQT_PACK_NAME), Client Transaction Name (SVDCRRQT_CLIENT_TN), or Client Accoutining Information (SVDCRRQT_CLIENT_AI) support substringing for this WLM MVS version.)
	1C13	entry	For the procedure name the substring value is greater than 128, or the substring value plus the number of characters is greater than 128.
	1C2F	entry	Classification rules that use LEVEL029 qualifier types were found and the functionality level in SVDCRLVL is set to less than LEVEL029.
	1C2A	entry	The Client Transaction Name qualifier type was specified greater than 255, or the substring value plus the number of characters is greater than 255.
	1C2B	entry	The Package Name qualifier type was specified greater than 128, or the substring value plus the number of characters is greater than 128.
	1C2C	entry	The Client IP Address qualifier type was specified greater than 39, or the substring value plus the number of characters is greater than 39.
	1C2D	entry	The Client UserID qualifier type specified was greater than 128, or the substring value plus the number of characters is greater than 128.
	1C2E	entry	The Client Workstation Name qualifier type was specified greater than 255, or the substring value plus the number of characters is greater than 255.
	1D05	entry	Group exists that has indicators on in the SVDCR reserved for future qualifier bytes. The reserved for future qualifier bytes are in SVDCRGTY_BYTE4.
	1D06	entry	No groups found. SVDCRGTY and SVDCRGTY_BYTE3+4 is zeroes.
	1D0A	entry	LEVEL029 classification rules found and the functionality level in the SVDCRLVL is not set to LEVEL029.

Table 103. SERVD application validation reason codes (continued)

Section	Reason	Offset	Description
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
SVDEFSCL	8401	offset	Duplicate service class names were found
	8402	section	Number of service classes (SVDEFCN) exceeds 100
	0401	entry	Name field (SVDEFCNM) has leading or imbedded blanks or contains a reserved character: * ? / , . ' ( ) & + - = ; -, =, ;)
	0402	entry	Workload name (SVDEFCWN) not found in the SVDEF
	0403	entry	Name field (SVDEFCNM) 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 (SVDEFGN) exceeds 32
	0601	entry	Name field (SVDEFGNM) 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 (SVDEFRN) exceeds 2047
	0701	entry	Name field (SVDEFRNM) has leading or imbedded blanks or contains a reserved character: * ? / , . ' ( ) & + - = ; -, =, ;)

Table 103. SERVD application validation reason codes (continued)

Section	Reason	Offset	Description
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 (SVDEFCGN) not found in resource group list
	0804	entry	Number of periods (SVDEFCPN) out of bounds (must be from 1 to 8)
	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 99,999,999
	0904	entry	Specified maximum value (SVDEFGMX) exceeds 99,999,999
	0905	entry	Specified minimum value (SVDEFGMN) exceeds maximum value (SVDEFGMX)

Table 103. SERVD application validation reason codes (continued)

Section	Reason	Offset	Description
SVDEFPDA	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, SVDEFAVG), but response time units (SVDEFRTU) not between 1 and 4.
	0507	entry	Percentile, average response time, or velocity goal specified (SVDEFPRC, SVDEFAVG, SVDEFVEL), but importance (SVDEFIMP) is not between 1 and 5.
	0508	entry	Percentile or average response time goal (SVDEFPRC, SVDEFAVG) but response time value (SVDEFVAL) is less than 15 milliseconds.
	0509	entry	Percentile or average response time goal (SVDEFPRC, SVDEFAVG) 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
SVDEFCNS	0A01	section	CPU coefficient (SDVEFCPU) exceeds maximum of 999,000
	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.

Table 103. SERVD application validation reason codes (continued)

Section	Reason	Offset	Description
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).
SVDEFEMS	0C01	entry	Guest platform management provider activation specified but the functionality level in SVDEFLVL is less than LEVEL025
	0C02	entry	Number of excluded host systems specified (SVDEVNSY) but the functionality level in SVDEFLVL is less than LEVEL025
	0C03	entry	The name of an excluded system (SCDEFSYN) is not valid
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 (SVDCRVS) 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 103. SERVD application 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

Table 103. SERVD application validation reason codes (continued)

Section	Reason	Offset	Description
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
	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.

Table 103. SERVD application validation reason codes (continued)

Section	Reason	Offset	Description
SVDCRRUL (continued)	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 pecified 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.
	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 does not support the region management option has a classification rule where the "Manage Region Using Goals Of" field is set to REGION or BOTH.

Table 103. SERVD application validation reason codes (continued)

Section	Reason	Offset	Description
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
	2804	SVNPA	Some section, as governed by offset and length fields, straddles some other section

Table 103. SERVD application validation reason codes (continued)

Section	Reason	Offset	Description
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).

Table 103. SERVD application validation reason codes (continued)

Section	Reason	Offset	Description
SVSEASR	3D20	entry	In the scheduling environment to resource connection, the scheduling environment name (SVSEA_SR_SCHENV_NAME) is not specified.
	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).

Table 103. SERVD application validation reason codes (continued)

Section	Reason	Offset	Description
XML	4001	–	The codepage which has been specified in XML is invalid. Only EDCDIC code pages starting with 'IBM' are accepted by the **`codepage=`** attribute.
	4002	–	The XML does not have the required **`xmlns=`** attribute in the 'ServiceDefinition' tag. The valid name spaces are listed in Appendix C, “Structure of the XML service definition (DTD),” on page 817.
	4003	–	The name space specified in the **`xmlns=`** attribute is invalid. The name space must have the following format: 'http://www.ibm.com/xmlns/prod/zwlm/*yyyy*/*mm*/ServiceDefinition.xsd' The valid name spaces are listed in Appendix C, “Structure of the XML service definition (DTD),” on page 817.
	4004	–	The level of the service definition which has been specified in the level tag does not match the name space specified in the **`xmlns=`** attribute. The valid name spaces and the corresponding levels are listed in Appendix C, “Structure of the XML service definition (DTD),” on page 817.
	4005	unexpected tag	The tag located at the returned offset is not expected at this position in the XML.
	4006	invalid keyword	The keyword located at the returned offset is not known by WLM.
	4007	invalid content	The content located at the returned offset is too long for the tag where it has been specified.
	4008	invalid content	The content located at the returned offset is not valid for the tag where it has been specified.
	4009	–	The XML is not complete. The input buffer does not end with the service definition end tag.

Appendix C. Structure of the XML service definition (DTD)

This section describes the following:

- The structure of the XML output of IWMDEXTR
- The layout of the XML service definition (DTD) that can be passed to IWMDINST

To obtain XML output, specify the `TYPE=XML` parameter for the IWMDEXTR service.

To install such an XML service definition with the IWMDINST service, also specify the `TYPE=XML` parameter.

The following DTD defines the structure of an XML service definition:

```
<!ELEMENT ServiceDefinition ( Name, Description?, CreationDate?, CreationUser?,
  ModificationDate?, ModificationUser?, Level, ReplId?, ProdId?, Notes,
  ResourceGroups, Workloads, ServicePolicies, ReportClasses,
  ClassificationGroups, Classifications, ServiceParameter,
  ApplicationEnvironments?, Resources?, SchedulingEnvironments?, GPMPSettings?,
  Extensions? ) >

<!ATTLIST ServiceDefinition
     xmlns         CDATA  #IMPLIED
     codepage  CDATA   #IMPLIED >

<!ELEMENT Name ( #PCDATA ) >
<!ELEMENT Description ( #PCDATA ) >
<!ELEMENT CreationDate ( #PCDATA ) >
<!ELEMENT CreationUser ( #PCDATA ) >
<!ELEMENT ModificationDate ( #PCDATA ) >
<!ELEMENT ModificationUser ( #PCDATA ) >
<!ELEMENT Level ( #PCDATA ) >
<!ELEMENT ReplId ( #PCDATA ) >
<!ELEMENT ProdId ( #PCDATA ) >

<!ELEMENT Notes ( Note* ) >
<!ELEMENT Note ( #PCDATA ) >

<!ELEMENT ResourceGroups ( ResourceGroup* ) >
<!ELEMENT ResourceGroup ( Name, Description?, CreationDate, CreationUser,
  ModificationDate, ModificationUser, Type?, CapacityMinimum?,
  CapacityMaximum?, MemoryLimit? ) >

<!ELEMENT Type ( #PCDATA ) >
<!ELEMENT CapacityMaximum ( #PCDATA ) >
<!ELEMENT CapacityMinimum ( #PCDATA ) >
<!ELEMENT MemoryLimit ( #PCDATA ) >

<!ELEMENT Workloads ( Workload* ) >
<!ELEMENT Workload ( Name, Description?, CreationDate, CreationUser,
  ModificationDate, ModificationUser, ServiceClasses ) >

<!ELEMENT ServiceClasses ( ServiceClass* ) >
<!ELEMENT ServiceClass ( Name, Description?, CreationDate, CreationUser,
  ModificationDate, ModificationUser, CPUCritical?, IOPriorityGroup?,
  HonorPriority?, ResourceGroupName?, Goal ) >

<!ELEMENT ResourceGroupName ( #PCDATA ) >

<!ELEMENT Goal ( (AverageResponseTime | PercentileResponseTime | Velocity)*,
  Discretionary? ) >
```

```
<!ELEMENT AverageResponseTime ( Importance, Duration?, ResponseTime ) >

<!ELEMENT Importance ( #PCDATA ) >
<!ELEMENT Duration ( #PCDATA ) >
<!ELEMENT ResponseTime ( #PCDATA ) >

<!ELEMENT PercentileResponseTime ( Importance, Duration?, ResponseTime,
  Percentile ) >

<!ELEMENT Percentile ( #PCDATA ) >

<!ELEMENT Velocity ( Importance, Duration?, Level ) >

<!ELEMENT Discretionary EMPTY >

<!ELEMENT ServicePolicies ( ServicePolicy* ) >
<!ELEMENT ServicePolicy ( Name, Description?, CreationDate, CreationUser,
  ModificationDate, ModificationUser, ServiceClassOverrides,
  ResourceGroupOverrides ) >

<!ELEMENT ServiceClassOverrides ( ServiceClassOverride* ) >
<!ELEMENT ServiceClassOverride ( ServiceClassName, CPUCritical?,
  IOPriorityGroup?, HonorPriority?, ResourceGroupName?, Goal ) >

<!ELEMENT ServiceClassName ( #PCDATA ) >
<!ELEMENT CPUCritical ( #PCDATA ) >
<!ELEMENT IOPriorityGroup ( #PCDATA ) >
<!ELEMENT HonorPriority ( #PCDATA )  >

<!ELEMENT ResourceGroupOverrides ( ResourceGroupOverride* ) >
<!ELEMENT ResourceGroupOverride ( ResourceGroupName, Type?,
  CapacityMinimum?, CapacityMaximum?, MemoryLimit? ) >

<!ELEMENT ReportClasses ( ReportClass* ) >
<!ELEMENT ReportClass ( Name, Description?, CreationDate,
  CreationUser, ModificationDate, ModificationUser ) >

<!ELEMENT ClassificationGroups ( ClassificationGroup* ) >
<!ELEMENT ClassificationGroup ( Name, Description?, CreationDate,
  CreationUser, ModificationDate, ModificationUser, QualifierType,
  QualifierNames ) >

<!ELEMENT QualifierType ( #PCDATA ) >

<!ELEMENT QualifierNames ( QualifierName* ) >
<!ELEMENT QualifierName ( Name, Description?, Start? ) >

<!ELEMENT Classifications ( Classification* ) >
<!ELEMENT Classification ( SubsystemType, Description?, CreationDate,
  CreationUser, ModificationDate, ModificationUser,
  DefaultServiceClassName?, DefaultReportClassName?, EWLMClassification?,
  ClassificationRules? ) >

<!ELEMENT SubsystemType ( #PCDATA ) >
<!ELEMENT DefaultServiceClassName ( #PCDATA ) >
<!ELEMENT DefaultReportClassName ( #PCDATA ) >

<!ELEMENT ClassificationRules ( ClassificationRule* ) >
<!ELEMENT ClassificationRule ( Description?, QualifierType, QualifierValue,
  Start?, ServiceClassName?, ReportClassName?, StorageCritical?, RegionGoal?,
  ReportingAttribute?, ClassificationRule* ) >

<!ELEMENT QualifierValue ( #PCDATA ) >
<!ELEMENT Start ( #PCDATA ) >
<!ELEMENT ReportClassName ( #PCDATA ) >
<!ELEMENT RegionGoal ( #PCDATA ) >
```

```
<!ELEMENT StorageCritical ( #PCDATA ) >
<!ELEMENT ReportingAttribute ( #PCDATA ) >
<!ELEMENT ServiceParameter ( ServiceCoefficients, ServiceOptions? ) >

<!ELEMENT ServiceCoefficients ( CPU, IOC, MSO, SRB )? >

<!ELEMENT CPU ( #PCDATA ) >
<!ELEMENT IOC ( #PCDATA ) >
<!ELEMENT MSO ( #PCDATA ) >
<!ELEMENT SRB ( #PCDATA ) >

<!ELEMENT EWLMClassification ( #PCDATA ) >

<!ELEMENT ServiceOptions ( IOPriorityManagement, DynamicAliasManagement?,
  IOPriorityGroupsEnabled? ) >

<!ELEMENT IOPriorityManagement ( #PCDATA ) >
<!ELEMENT DynamicAliasManagement ( #PCDATA ) >
<!ELEMENT IOPriorityGroupsEnabled ( #PCDATA ) >

<!ELEMENT ApplicationEnvironments ( ApplicationEnvironment* ) >
<!ELEMENT ApplicationEnvironment (  Name, Description?, SubsystemType, Limit,
  ProcedureName?, StartParameter? ) >

<!ELEMENT StartParameter ( #PCDATA ) >
<!ELEMENT Limit ( #PCDATA ) >
<!ELEMENT ProcedureName ( #PCDATA ) >

<!ELEMENT Resources ( Resource* ) >
<!ELEMENT Resource ( Name, Description? ) >

<!ELEMENT SchedulingEnvironments ( SchedulingEnvironment* ) >
<!ELEMENT SchedulingEnvironment ( Name, Description?, ResourceNames ) >

<!ELEMENT ResourceNames ( ResourceName* ) >
<!ELEMENT ResourceName ( Name, RequiredState ) >

<!ELEMENT RequiredState ( #PCDATA ) >

<!ELEMENT GPMPSettings  ( Activation, ExcludedHostSystems? ) >
<!ELEMENT Activation ( #PCDATA  ) >
<!ELEMENT ExcludedHostSystems  ( ExcludedHostSystem*  ) >
<!ELEMENT ExcludedHostSystem ( Name  ) >

<!ELEMENT Extensions ( ServiceDefinitionExtensions?,
  ResourceGroupExtensions?, ResourceGroupAttributeExtensions?,
  WorkloadExtensions?, ServiceClassExtensions?,
  ServiceClassAttributeExtensions?, ServicePolicyExtensions?,
  ReportClassExtensions?, ClassificationExtensions?,
  ApplicationEnvironmentExtensions?, ResourceExtensions?,
  SchedulingEnvironmentHeaderExtensions?,
  SchedulingEnvironmentExtensions?,
  SchedulingEnvironmentResourceExtensions? ) >

<!ELEMENT ServiceDefinitionExtensions ( ServiceDefinitionExtension* ) >
<!ELEMENT ServiceDefinitionExtension (VendorId?, RelatedObject,
  ExtensionData?) >

<!ELEMENT VendorId ( #PCDATA ) >
<!ELEMENT RelatedObject ( #PCDATA ) >
<!ELEMENT ExtensionData ( #PCDATA ) >

<!ELEMENT ResourceGroupExtensions ( ResourceGroupExtension* ) >
<!ELEMENT ResourceGroupExtension (VendorId?, RelatedObject, ServicePolicyName?,
  ExtensionData?) >

<!ELEMENT ServicePolicyName ( #PCDATA ) >
```

```
<!ELEMENT ResourceGroupAttributeExtensions ( ResourceGroupAttributeExtension* ) >
<!ELEMENT ResourceGroupAttributeExtension (VendorId?, RelatedObject,
  ServicePolicyName?, ExtensionData?) >

<!ELEMENT WorkloadExtensions ( WorkloadExtension* ) >
<!ELEMENT WorkloadExtension (VendorId?, RelatedObject, ServicePolicyName?,
  ExtensionData?) >

<!ELEMENT ServiceClassExtensions ( ServiceClassExtension* ) >
<!ELEMENT ServiceClassExtension (VendorId?, RelatedObject, ServicePolicyName?,
  ExtensionData?) >

<!ELEMENT ServiceClassAttributeExtensions ( ServiceClassAttributeExtension* ) >
<!ELEMENT ServiceClassAttributeExtension (VendorId?, RelatedObject,
  ServicePolicyName?, ExtensionData?) >

<!ELEMENT ServicePolicyExtensions ( ServicePolicyExtension* ) >
<!ELEMENT ServicePolicyExtension (VendorId?, RelatedObject, ServicePolicyName?,
  ExtensionData?) >

<!ELEMENT ReportClassExtensions ( ReportClassExtension* ) >
<!ELEMENT ReportClassExtension (VendorId?, RelatedObject, ServicePolicyName?,
  ExtensionData?) >

<!ELEMENT ClassificationExtensions ( ClassificationExtension* ) >
<!ELEMENT ClassificationExtension (VendorId?, RelatedObject, ExtensionData?) >

<!ELEMENT ApplicationEnvironmentExtensions ( ApplicationEnvironmentExtension* ) >
<!ELEMENT ApplicationEnvironmentExtension (VendorId?, RelatedObject,
  ExtensionData?) >

<!ELEMENT ResourceExtensions ( ResourceExtension* ) >
<!ELEMENT ResourceExtension (VendorId?, RelatedObjectName?, ExtensionData?) >

<!ELEMENT SchedulingEnvironmentHeaderExtensions
  ( SchedulingEnvironmentHeaderExtension* ) >
<!ELEMENT SchedulingEnvironmentHeaderExtension
  (VendorId?, RelatedObject, ExtensionData?) >

<!ELEMENT SchedulingEnvironmentExtensions
  ( SchedulingEnvironmentExtension* ) >
<!ELEMENT SchedulingEnvironmentExtension
  (VendorId?, RelatedObject, ExtensionData?) >

<!ELEMENT SchedulingEnvironmentResourceExtensions
  ( SchedulingEnvironmentResourceExtension* ) >
<!ELEMENT SchedulingEnvironmentResourceExtension
  (VendorId?, RelatedObject, ExtensionData?) >
```

Table 104 lists the valid name spaces and the corresponding functionality levels:

Table 104. Valid name spaces and corresponding functionality levels

Name space	**Level**
http://www.ibm.com/xmlns/prod/zwlm/1993/09/ServiceDefinition.xsd	001
http://www.ibm.com/xmlns/prod/zwlm/1994/09/ServiceDefinition.xsd	002
http://www.ibm.com/xmlns/prod/zwlm/1997/03/ServiceDefinition.xsd	003
http://www.ibm.com/xmlns/prod/zwlm/1997/09/ServiceDefinition.xsd	004

Table 104. Valid name spaces and corresponding functionality levels (continued)

Name space	Level
http://www.ibm.com/xmlns/prod/zwlm/1998/09/ServiceDefinition.xsd	006
http://www.ibm.com/xmlns/prod/zwlm/1999/03/ServiceDefinition.xsd	007
http://www.ibm.com/xmlns/prod/zwlm/1999/09/ServiceDefinition.xsd	008
http://www.ibm.com/xmlns/prod/zwlm/2000/09/ServiceDefinition.xsd	011
http://www.ibm.com/xmlns/prod/zwlm/2001/09/ServiceDefinition.xsd	013
http://www.ibm.com/xmlns/prod/zwlm/2005/12/ServiceDefinition.xsd	017
http://www.ibm.com/xmlns/prod/zwlm/2006/09/ServiceDefinition.xsd	019
http://www.ibm.com/xmlns/prod/zwlm/2008/09/ServiceDefinition.xsd	021
http://www.ibm.com/xmlns/prod/zwlm/2009/09/ServiceDefinition.xsd	023
http://www.ibm.com/xmlns/prod/zwlm/2010/09/ServiceDefinition.xsd	025
http://www.ibm.com/xmlns/prod/zwlm/2012/09/ServiceDefinition.xsd	029
http://www.ibm.com/xmlns/prod/zwlm/2015/12/ServiceDefinition.xsd	030
http://www.ibm.com/xmlns/prod/zwlm/2016/12/ServiceDefinition.xsd	031

Appendix D. C language interfaces for workload management services

Table 105 shows C language interfaces with their associated workload management services. See *z/OS XL C/C++ Runtime Library Reference* for more information about these and other C language interfaces.

Also see "Interfaces for sysplex routing services" on page 824 for the four C interfaces for accessing the WLM sysplex routing services, and "Interface for querying a virtual server" on page 825 for the C interface for querying a virtual server.

Table 105. C language interfaces

C language interface	Associated WLM service	Reference information
CheckSchEnv	`IWMSEDES`	"IWMSEDES — Scheduling environments determine execution service" on page 344
ConnectExportImport	`IWMCONN WORK_MANAGER=NO ROUTER=NO QUEUE_MANAGER=NO SERVER_MANAGER=NO EXPTIMPT=YES`	"IWMCONN — Connect to workload management" on page 853
ConnectServer	`IWMCONN WORK_MANAGER=NO ROUTER=NO QUEUE_MANAGER=YES SERVER_MANAGER=YES`	"IWMCONN — Connect to workload management" on page 853
ConnectWorkMgr	`IWMCONN WORK_MANAGER=YES ROUTER=NO QUEUE_MANAGER=YES SERVER_MANAGER=NO EXPTIMPT=YES`	"IWMCONN — Connect to workload management" on page 853
ContinueWorkUnit	`IWMCREA TYPE=DEPENDENT`	"IWMECREA — Create an enclave" on page 885
CreateWorkUnit	`IWMCREA TYPE=INDEPENDENT`	"IWMECREA — Create an enclave" on page 885
DeleteWorkUnit	`IWMEDELE`	"IWMEDELE — Delete an enclave" on page 896
DisconnectServer	`IWMDISC`	"IWMDISC — Disconnect from workload management" on page 872
ExportWorkUnit	`IWMEXPT`	"IWMEXPT — Export a WLM enclave" on page 263
ExtractWorkUnit	`IWMESQRY`	"IWMESQRY — Query enclave state" on page 236
ImportWorkUnit	`IWMIMPT`	"IWMIMPT — Import an enclave" on page 273
JoinWorkUnit	`IWMJOIN`	"IWMEJOIN — Join WLM enclave" on page 200
LeaveWorkUnit	`IWMELEAV`	"IWMELEAV — Leave WLM enclave" on page 208

Table 105. C language interfaces (continued)

C language interface	Associated WLM service	Reference information
QueryMetrics	IWMWSYSQ	"IWMWSYSQ — Query system information" on page 446
QuerySchEnv	IWMSEQRY	"IWMSEQRY — Scheduling environments query service" on page 350
QueryWorkUnitClassification	IWM4EQRY	"IWM4EQRY — Query an enclave" on page 528
UnDoExportWorkUnit	IWMUEXPT	"IWMUEXPT — WLM undo export" on page 416
UnDoImportWorkUnit	IWMUIMPT	"IWMUIMPT — WLM undo import" on page 422
_server_classify	IWM4CLSY	"IWM4CLSY — Classify work" on page 464
_server_classify_create	IWM4CLSY	"IWM4CLSY — Classify work" on page 464
_server_classify_init	IWMCONN, IWMDISC	• "IWMCONN — Connect to workload management" on page 853 • "IWMDISC — Disconnect from workload management" on page 872
_server_classify_pwu	IWMQINS,IWMSSEL, IWMSTEND,IWMESQRY, IWMSREF,IWMSTBGN, IWMEDELE,IWMECREA, IWM4CLSY,IWMDISC	• "IWMQINS — Insert a request onto the queue for an execution address space" on page 1016 • "IWMSSEL — Select a request from a caller's work manager queue" on page 1047 • "IWMSTEND — End a request from a caller's work manager queue" on page 1070 • "IWMESQRY — Query enclave state" on page 236 • "IWMSTBGN — Begin a request from a caller's work manager queue" on page 1062 • "IWMEDELE — Delete an enclave" on page 896 • "IWMECREA — Create an enclave" on page 885 • "IWM4CLSY — Classify work" on page 464 • "IWMDISC — Disconnect from workload management" on page 872
_server_thread_query	IWMSINF	"IWMSINF — WLM server manager inform service" on page 368

Interfaces for sysplex routing services

Table 106 on page 825 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 106. C language interfaces for WLM sysplex routing services

C language interface	Associated WLM sysplex routing service	Reference information
IWMDNREG	IWMSRSRG	"IWMSRSRG — Register a server for sysplex routing" on page 396
IWMDNDRG	IWMSRDRS	"IWMSRDRS — Deregister a server for sysplex routing" on page 382
IWMDNGRP	IWMSRDNS	"IWMSRDNS — Get sysplex routing location list" on page 376
IWMDNSRV	IWMSRSRS	"IWMSRSRS — Sysplex routing information" on page 405

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 E. WLM services supporting 31-bit addressing only

The following WLM services support 31-bit addressing only.

The equivalent form of these services that support both 31-bit and 64-bit addressing are described in Chapter 12, “Workload management services,” on page 143.

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 equivalent service, IWM4AEDF, which also supports 64-bit addressing. For more information, see "IWM4AEDF — WLM define dynamic application environments" on page 454.

Environment

The requirements for the caller are:

Minimum authorization:	Supervisor state or program key mask (PKM) allowing keys 0-7.
Dispatchable unit mode:	Task or SRB
Cross memory mode:	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).
AMODE:	31-bit
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks may be held.
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, 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 supports multiple versions. Some keywords are unique to certain versions. Refer to the description of the PLISTVER parameter for further information.

Input register information

Before issuing the IWMAEDEF 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

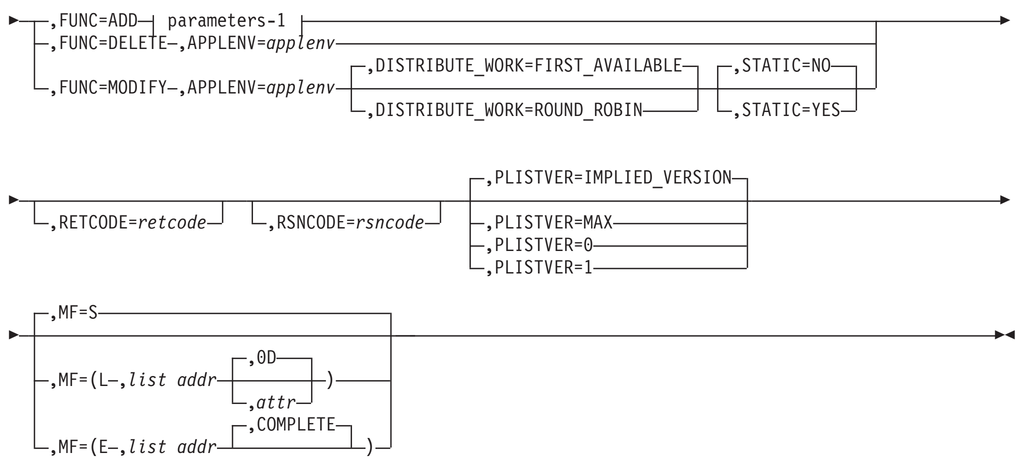
None.

Syntax

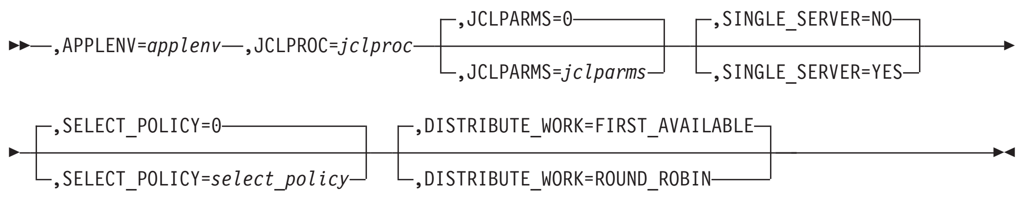
The syntax of the IWMAEDEF macro is as follows:

main diagram

```
►►──┬──────┬──IWMAEDEF──CONNTKN=conntkn──────────────────────────────────────►
    └─name─┘
```



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***,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**, which supports all parameters except those specifically referenced in higher versions.

1, which supports 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 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 codes 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 107. Return and Reason Codes for the IWMAEDEF Macro

Return Code	Reason Code	Equate Symbol, Meaning, and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: Caller is locked. **Action**: Avoid requesting this function while locked.
8	xxxx080B	**Equate Symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list. **Action**: Check for possible storage overlay.
8	xxxx0821	**Equate Symbol**: IwmRsnCodeBadConn **Meaning**: Input connect token does not pass validity checking. **Action**: 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	**Equate Symbol**: IwmRsnCodeDatoff **Meaning**: The caller invoked the service while DATOFF **Action**: Avoid requesting this function in this environment.
8	xxxx0824	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: Caller invoked the service but was in 24-bit addressing mode. **Action**: Request this function only when you are in 31-bit addressing mode.
8	xxxx0825	**Equate Symbol**: IwmRsnCodeAscModeNotPrimary **Meaning**: Caller invoked service but was not in primary ASC mode. **Action**: Avoid requesting this function in this environment.

Table 107. Return and Reason Codes for the IWMAEDEF Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx0827	**Equate Symbol**: IwmRsnCodeRsvdNot0 **Meaning**: Reserved field in parameter list was non-zero. **Action**: Check for use of keywords that are not supported by the MVS release on which the program is running.
8	xxxx0828	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: Version number in parameter list is not valid. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx083F	**Equate Symbol**: IwmRsnCodePrimaryNotOwnConn **Meaning**: Primary address space does not own the passed connect token. **Action**: 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.
8	xxxx0840	**Equate Symbol**: IwmRsnCodeServiceNotEnabled **Meaning**: Caller's space connection is not enabled for this service **Action**: Make sure that Queue_Manager=Yes, or Server_Manager=Yes with Server_Type=Queue is specified on the IWM4CON request to enable this service.
8	xxxx0842	**Equate Symbol**: IwmRsnCodeNoWLMConnect **Meaning**: Caller's space disconnected from WLM during processing of the request. **Action**: None.
8	xxxx0890	**Equate Symbol**: IwmRsnCodeApplEnvExists **Meaning**: The caller tried to add an application environment that has already been defined. subsystem type. **Action**: Check whether the correct application environment name is being used. Make sure that a unique application environment name is used when adding application environments.
8	xxxx0891	**Equate Symbol**: IwmRsnCodeApplEnvNotFound **Meaning**: The caller tried to delete or modify an application environment that does not exist. **Action**: Check whether the correct application environment name is being used.
C	—	**Equate Symbol**: IwmRetCodeEnvError **Meaning**: Environmental error.
C	xxxx0C01	**Equate Symbol**: IwmRsnCodeNoStg **Meaning**: Storage is not available for the request. **Action**: Contact your system programmer. There is a common storage shortage.

Table 107. Return and Reason Codes for the IWMAEDEF Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error. **Action**: Contact your system programmer.

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
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
```

IWMCLSFY — Classify work request

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:

Minimum authorization:	Either problem state or supervisor state. PSW key must either be 0 or match the value supplied on IWM4CON.
Dispatchable unit mode:	Task or SRB
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:	Unlocked or locked.
Control parameters:	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

- This macro may only be used on MVS/ESA (version 3 or higher), NOT versions 1 or 2 of MVS.
- FRRs are allowed.
- This macro may not be used during task/address space termination for the connect owner.
- 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.
- Only limited checking is done of the connect token obtained from IWM4CON.
- SOURCELU is mutually exclusive with NETID/LUNAME.
- This macro supports multiple versions. Some keywords are unique to certain versions. See the PLISTVER parameter description.

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

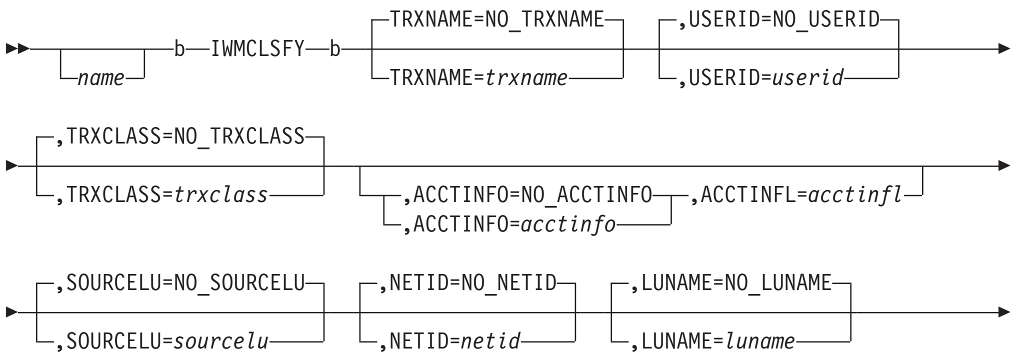
Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

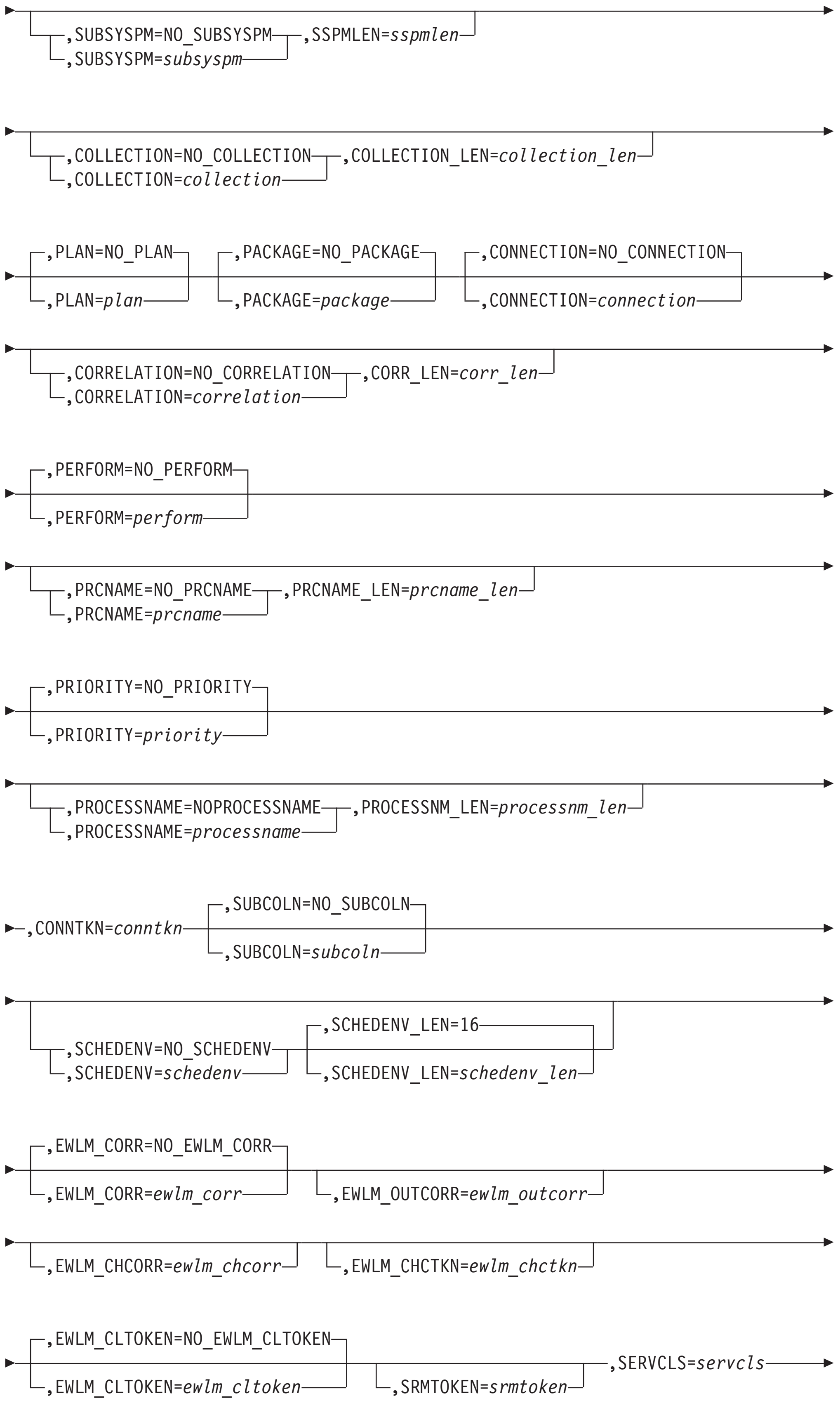
Performance implications

None.

Syntax

main diagram




,SUBSYSPM=NO_SUBSYSPM
,SUBSYSPM=*subsyspm*
,SSPMLEN=*sspmlen*
,COLLECTION=NO_COLLECTION
,COLLECTION=*collection*
,COLLECTION_LEN=*collection_len*
,PLAN=NO_PLAN
,PLAN=*plan*
,PACKAGE=NO_PACKAGE
,PACKAGE=*package*
,CONNECTION=NO_CONNECTION
,CONNECTION=*connection*
,CORRELATION=NO_CORRELATION
,CORRELATION=*correlation*
,CORR_LEN=*corr_len*
,PERFORM=NO_PERFORM
,PERFORM=*perform*
,PRCNAME=NO_PRCNAME
,PRCNAME=*prcname*
,PRCNAME_LEN=*prcname_len*
,PRIORITY=NO_PRIORITY
,PRIORITY=*priority*
,PROCESSNAME=NOPROCESSNAME
,PROCESSNAME=*processname*
,PROCESSNM_LEN=*processnm_len*
,CONNTKN=*conntkn*
,SUBCOLN=NO_SUBCOLN
,SUBCOLN=*subcoln*
,SCHEDENV=NO_SCHEDENV
,SCHEDENV=*schedenv*
,SCHEDENV_LEN=16
,SCHEDENV_LEN=*schedenv_len*
,EWLM_CORR=NO_EWLM_CORR
,EWLM_CORR=*ewlm_corr*
,EWLM_OUTCORR=*ewlm_outcorr*
,EWLM_CHCORR=*ewlm_chcorr*
,EWLM_CHCTKN=*ewlm_chctkn*
,EWLM_CLTOKEN=NO_EWLM_CLTOKEN
,EWLM_CLTOKEN=*ewlm_cltoken*
,SRMTOKEN=*srmtoken*
,SERVCLS=*servcls*

```
►──┬───────────────────────┬──┬───────────────────────┬──┬─────────────────────────────┬──►
   └─,SRVCLSNM=srvclsnm────┘  └─,RPTCLSNM=rptclsnm────┘  └─,TTRACETOKEN=ttracetoken────┘

                                                ┌─,PLISTVER=IMPLIED_VERSION─┐
►──┬─────────────────────┬──┬─────────────────────┬──┼───────────────────────────┼──►
   └─,RETCODE=retcode────┘  └─,RSNCODE=rsncode────┘  ├─,PLISTVER=MAX─────────────┤
                                                ├─,PLISTVER=0───────────────┤
                                                ├─,PLISTVER=1───────────────┤
                                                ├─,PLISTVER=2───────────────┤
                                                ├─,PLISTVER=3───────────────┤
                                                ├─,PLISTVER=4───────────────┤
                                                ├─,PLISTVER=5───────────────┤
                                                ├─,PLISTVER=6───────────────┤
                                                ├─,PLISTVER=7───────────────┤
                                                └─,PLISTVER=8───────────────┘

   ┌─,MF=S──────────────────────────────────────┐
►──┼────────────────────────────────────────────┼──────────────────────────────►◄
   │                        ┌─,0D────┐          │
   ├─,MF=(L─,list addr──────┼────────┼─)────────┤
   │                        └─,attr──┘          │
   │                        ┌─,COMPLETE─┐       │
   ├─,MF=(E─,list addr──────┼───────────┼─)─────┤
   │                        └─,NOCHECK──┘       │
   │                        ┌─,COMPLETE─┐       │
   └─,MF=(M─,list addr──────┼───────────┼─)─────┘
                            └─,NOCHECK──┘
```

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, or specify a literal decimal value.

,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. 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 (getmained).

To code: Specify the RS-type address, or address in register (2)-(12), of a fullword field, or specify a literal decimal value.

,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 IWMCONN 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, or specify a literal decimal value.

,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 specification of NO_CORRELATION The default is NO_CORRELATION. 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. Specification of this parameter indicates that a sub work request will be created.

Notes:

- Currently (z/OS V1R6) only uses the first 64 Bytes of the EWLM_CHCORR field.
- Parameters EWLM_OUTCORR, EWLM_CHCORR and EWLM_CHCTKN are all 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. Specification of this parameter indicates that a sub work request will be created. A EWLM child correlator token must not be passed outside of the EWLM Management Domain.

Parameters EWLM_OUTCORR, EWLM_CHCORR and EWLM_CHCTKN are all mutually exclusive.

To code: Specify the RS-type address, or address in register (2)-(12), of a 64-character field.

,EWLM_CLTOKEN=*ewlm_cltoken*
,EWLM_CLTOKEN=NO_EWLM_CLTOKEN

An optional input parameter, which contains internal EWLM classification information to be passed from EWLM to WLM. This parameter is internally used by WLM and must not be used by application programs. The default is NO_EWLM_CLTOKEN. indicates that no EWLM classification information was passed.

To code: Specify the RS-type address, or address in register (2)-(12), of a 40-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. Moreover the EWLM correlator serves as the input correlator for the EWLM_CHCORR, EWLM_CHTKN, EWLM_OUTCORR parameters.

Note, that 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).

For environments where the EWLM correlator may be available on some, but not all flows, provision of a data area with the first four bytes set to binary zeroes is equivalent to the specification of NO_EWLM_CORR.

The default is NO_EWLM_CORR. 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_OUTCORR=*ewlm_outcorr*

An optional output parameter, which will receive a validated EWLM correlator on return. The execution form of IWMCLSFY will validate the passed correlator in EWLM_CORR and provide a valid EWLM correlator in EWLM_OUTCORR as follows:

- If the EWLM_CORR parameter is specified and the the correlator in EWLM_CORR is a valid ARM correlator in EWLM format, it will be copied to EWLM_OUTCORR.
- 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:

- The specification of EWLM_OUTCORR (unlike EWLM_CHCORR or EWLM_CHCTKN) does not indicate the beginning of an sub work request.
- Currently (z/OS V1R6) only uses the first 64 Bytes of the EWLM_OUTCORR field.
- The application may specify the same parameter for both EWLM_CORR and EWLM_OUTCORR, which means that the EWLM correlator can be validated/replaced in place.
- Parameters EWLM_OUTCORR, EWLM_CHCORR and EWLM_CHCTKN are all 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
,PLISTVER=8
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:

PERFORM	PRCNAME	PRCNAME_LEN

- **2**, which supports both the following parameters and those from version 0 and 1:

PRIORITY

- **3**, which supports both the following parameters and those from version 0,1 and 2:

PROCESSNAME	PROCESSNM_LEN

- **4**, which supports both the following parameters and 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	SRMTOKEN
SCHEDENV_LEN	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

- **8**, which supports both the following parameters and those from version 0,1,2,3,4,5,6 and 7:

EWLM_CLTOKEN

To code: Specify one of the following:

- IMPLIED_VERSION
- MAX
- A decimal value of 0, 1, 2, 3, 4, 5, 6, 7, or 8

,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. 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, or specify a literal decimal value.

,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. indicates that no PRIORITY value is passed.

To code: Specify the RS-type address, or address in register (2)-(12), of a fullword field, or specify a literal decimal value.

,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, or specify a literal decimal value.

,RETCODE=*retcode*
An optional output parameter into which the return code is to be copied from GPR 15. If you specify 15, GPR15, REG15, or R15 (within or without parentheses), the value will be left in GPR 15.

To code: Specify the RS-type address of a fullword field, or register (2)-(12) or (15), (GPR15), (REG15), or (R15).

,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. If you specify 0, 00, GPR0, GPR00, REG0, REG00, or R0 (within or without parentheses), the value will be left in GPR 0.

To code: Specify the RS-type address of a fullword field, or register (0) or (2)-(12), (00), (GPR0), (GPR00), REG0), (REG00), or (R0).

,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, or specify a literal decimal value.

,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 will determine 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, or specify a literal decimal value.

,SUBCOLN=*subcoln*
,SUBCOLN=NO_SUBCOLN

An optional input parameter, which contains the subsystem collection name associated with the work request. The default is NO_SUBCOLN. indicates 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 codes and reason codes

When the IWMCLSFY 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 108. Return and Reason Codes for the IWMCLSFY Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.

Table 108. Return and Reason Codes for the IWMCLSFY Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.
4	xxxx0409	**Equate Symbol**: IwmRsnCodeNoConn **Meaning**: Connect token does not reflect a successful Connect. **Action**: None required.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0821	**Equate Symbol**: IwmRsnCodeBadConn **Meaning**: Input connect token does not pass validity checking. **Action**: Check for possible storage overlay.
8	xxxx0840	**Equate Symbol**: IwmRsnCodeServiceNotEnabled **Meaning**: Caller's space connection is not enabled for this service. **Action**: Avoid requesting this function under the input connection. IWM4CON options must be specified previously to enable this service.
8	xxxx0894	**Equate Symbol**: IwmRsnCodeInvalidEWLMCorr **Meaning**: 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). **Action**: Check the specification of the EWLM correlator in the classification information.
8	xxxx0895	**Equate Symbol**: IwmRsnCodeEWLMServNotEnabled **Meaning**: Service is not enabled because caller invoked the IWMCONN service with EWLM=NO. **Action**: Specify the parameter EWLM_CORR only when connected with EWLM=YES.
C	—	**Equate Symbol**: IwmRetCodeEnvError **Meaning**: Environmental error.
C	xxxx0C01	**Equate Symbol**: IwmRsnCodeNoStg **Meaning**: Storage is not available for the request. **Action**: There is a storage shortage. The function may work successfully at a later time.
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error. **Action**: No action required. The function may be successful if invoked again.

Example

Suppose the transactions processed by a subsystem work manager have the following qualifiers:

- User ID
- Transaction name
- Transaction class

To get the service class associated with an incoming work request, specify:

```
IWMCLSFY USERID=AUSERID,TRXCLASS=ATRXCLS,TRXNAME=ATRXNM,
        CONNTKN=(R7),SERVCLS=(R9),
        RETCODE=RETCODE,RSNCODE=RSNCODE
```

Where the following are declared:

```
AUSERID  DS  CL8
ATRXCLS  DS  CL8
ATRXNM   DS  CL8
```

IWMCONN — Connect 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 equivalent service, IWM4CON, which also supports 64-bit addressing. For more information, see "IWM4CON — Connect to workload management" on page 480.

Environment

The requirements for the caller are:

Minimum authorization:	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.
Dispatchable unit mode:	Task
Cross memory mode:	Non-XMEM when input key is a user key or SERVER_MANAGER = YES, otherwise XMEM, any P,S,H.
AMODE:	31-bit
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	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 not be used during task/address space termination.
2. Only a single connection is allowed to be active for a given address space at any given time.
3. Specification of both Queue_Manager=Yes, and Server_Manager=Yes requires that Server_Type=Queue. Specification of Server_Type=Routing is rejected.
4. Specification of both Router=Yes, and Server_Manager=Yes requires that Server_Type=Routing. Specification of Server_Type=Queue is rejected.
5. 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.
6. 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).
7. 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.

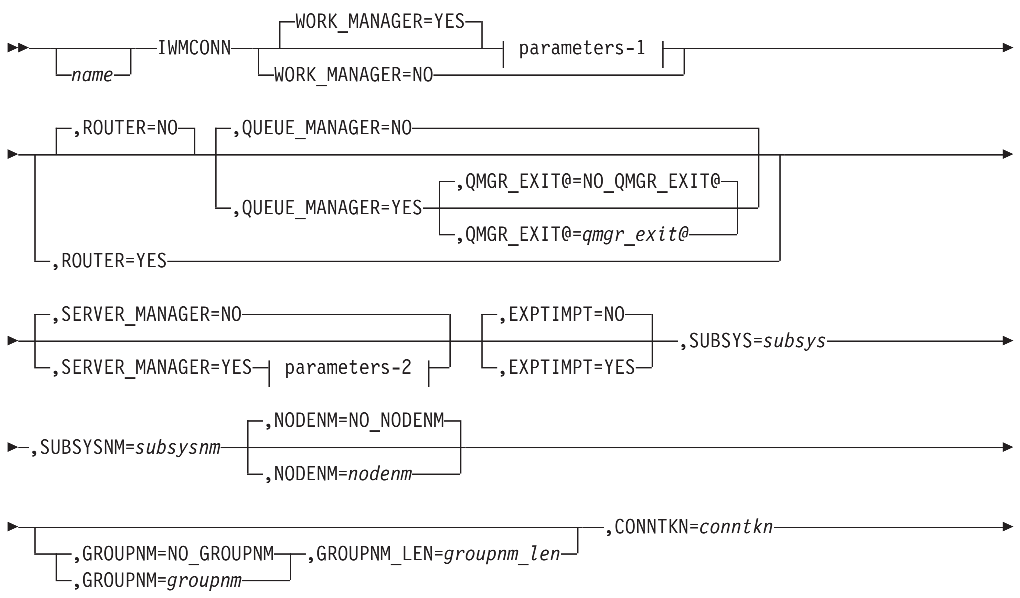
Performance implications

None.

Syntax

The syntax of the IWMCONN macro is as follows:

main diagram



```
                                            ┌─,PLISTVER=IMPLIED_VERSION─┐
►──┬──────────────────┬──┬──────────────────┬──┼───────────────────────────┼──────────────►
   └─,RETCODE=retcode─┘  └─,RSNCODE=rsncode─┘  ├─,PLISTVER=MAX─────────────┤
                                            ├─,PLISTVER=0───────────────┤
                                            ├─,PLISTVER=1───────────────┤
                                            ├─,PLISTVER=2───────────────┤
                                            ├─,PLISTVER=3───────────────┤
                                            └─,PLISTVER=4───────────────┘

   ┌─,MF=S──────────────────────────────────┐
►──┼────────────────────────────────────────┼──────────────────────────────────────────►◄
   │                        ┌─,0D─────┐     │
   ├─,MF=(L─,list addr──────┼─────────┼─)───┤
   │                        └─,attr───┘     │
   │                        ┌─,COMPLETE─┐   │
   └─,MF=(E─,list addr──────┴───────────┴─)─┘
```

parameters-1

```
                                                               ┌─,EWLM=NO──┐
►►──┬──────────────────────────────────────────────────────┬──┼───────────┼─────────────►
    └─┬─,TOPOLOGY=NO_TOPOLOGY─┬─,NUMBERASCB=numberascb─────┘  └─,EWLM=YES─┘
      └─,TOPOLOGY=topology────┘

►──┬─,CONNTKNKEYP=VALUE─,CONNTKNKEY=conntknkey─┬──────────────────────────────────────►◄
   └─,CONNTKNKEYP=PSWKEY───────────────────────┘
```

parameters-2

```
                        ┌─,DYNAMIC=NO──┐
►►──,APPLENV=applenv────┼──────────────┼──,PARALLEL_EU=parallel_eu──────────────────────►
                        └─,DYNAMIC=YES─┘

     ┌─,SERVER_TYPE=QUEUE─┐  ┌─,MANAGE_TASKS=NO──────────────────────────────────┐
►──┬─┴────────────────────┴──┼───────────────────────────────────────────────────┼──┬──►◄
   │                         │                     ┌─,SERVER_LIMIT=1000─────────┐ │  │
   │                         └─,MANAGE_TASKS=YES───┼────────────────────────────┼─┘  │
   │                                               └─,SERVER_LIMIT=server_limit─┘    │
   └─,SERVER_TYPE=ROUTING─,SERVER_DATA=server_data─,SRV_MGR_EXIT@=srv_mgr_exit@──────┘
```

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 IWMCONN must have PSW update authority. These other services include Classify (IWMCLSFY), Report (IWMRPT), Notify (IWMMNTFY). Create (IWMMCREA) 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.

Note however that there are other services that use the connect token, for which the CONNTKNKEY does not relate to PSW update authority but instead must be a system key (0-7) rather than a user key (8-15).

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 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.

,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.

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**, which supports all parameters except those specifically referenced in higher versions.
- **1**, which supports the following parameters and those 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	GROUPNM_LEN

- **3**, which supports the following parameters and those from version 0, 1, and 2:

DYNAMIC	NODENM

- **4**, which supports 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 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 (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 choses 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 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 IWMMINIT 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 (IWMECREA)
- 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 codes 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.

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 109. Return and Reason Codes for the IWMCONN Macro

Return Code	Reason Code	Equate Symbol, Meaning, and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.

Table 109. Return and Reason Codes for the IWMCONN Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0801	**Equate Symbol**: IwmRsnCodeSrbMode **Meaning**: Caller is in SRB mode. **Action**: Avoid requesting this function while in SRB mode.
8	xxxx0802	**Equate Symbol**: IwmRsnCodeXmemUserKeyTkn **Meaning**: Caller is in cross-memory mode while the token was requested in user key. **Action**: Avoid requesting this function while in cross-memory mode.
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: Caller is locked. **Action**: Avoid requesting this function while locked.
8	xxxx080B	**Equate Symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list. **Action**: 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	**Equate Symbol**: IwmRsnCodeEutFrr **Meaning**: Caller has EUT FRR established. **Action**: Avoid requesting this function with an EUT FRR set.
8	xxxx0812	**Equate Symbol**: IwmRsnCodeBadAscb **Meaning**: Bad ASCB address passed. **Action**: Check for possible storage overlay.
8	xxxx0823	**Equate Symbol**: IwmRsnCodeDatoff **Meaning**: Caller invoked service while DATOFF **Action**: Avoid requesting this function in this environment.
8	xxxx0824	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: Caller invoked service but was in 24 bit addressing mode. **Action**: Request this function only when you are in 31 bit addressing mode.

Table 109. Return and Reason Codes for the IWMCONN Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx0825	**Equate Symbol**: IwmRsnCodeAscModeNotPrimary **Meaning**: Caller invoked service but was not DAT on Primary ASC mode. **Action**: Avoid requesting this function in this environment.
8	xxxx0826	**Equate Symbol**: IwmRsnCodeTaskTerm **Meaning**: Caller invoked service while task termination is in progress for the TCB associated with the owner. **Action**: Avoid requesting this function in this environment.
8	xxxx0827	**Equate Symbol**: IwmRsnCodeRsvdNot0 **Meaning**: Reserved field in parameter list was non-zero. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx0828	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: Version number in parameter list is not valid. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx0829	**Equate Symbol**: IwmRsnCodeBadOptions **Meaning**: Parameter list omits required parameters or supplies mutually exclusive parameters or provides data associated with options not selected. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx082C	**Equate Symbol**: IwmRsnCodeBadNumberAscb **Meaning**: NUMBERASCB variable is not a positive value. **Action**: Check for possible storage overlay of the parameter list or variable.
8	xxxx082E	**Equate Symbol**: IwmRsnCodeConnectExists **Meaning**: Connect has already been established for the current home address space. **Action**: Avoid requesting this function when a connection already exists.
8	xxxx0840	**Equate Symbol**: IwmRsnCodeServiceNotEnabled **Meaning**: 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. **Action**: Run the program as a started task.
8	xxxx0841	**Equate Symbol**: IwmRsnCodeXMemMode **Meaning**: Caller is in cross memory mode. **Action**: Invoke the function in non-cross memory mode.

Table 109. Return and Reason Codes for the IWMCONN Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx0847	**Equate Symbol**: IwmRsnCodeOtherSpaceConnected **Meaning**: 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. **Action**: Avoid requesting this function with duplicate values.
8	xxxx0849	**Equate Symbol**: IwmRsnCodeWLMServBadAPPL **Meaning**: The application environment name (APPLENV=) specified is not the same as the one used by WLM to start the server. **Action**: 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.
8	xxxx084A	**Equate Symbol**: IwmRsnCodeWLMServBadSSN **Meaning**: The subsystem name (SUBSYSNM=) specified is not the same as the one used by WLM to start the server. **Action**: 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.
8	xxxx084B	**Equate Symbol**: IwmRsnCodeWLMServBadSST **Meaning**: The subsystem type (SUBSYS=) specified is not the same as the one used by WLM to start the server. **Action**: 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.
8	xxxx084D	**Equate Symbol**: IwmRsnCodeNotAuthConnect **Meaning**: The caller must be supervisor state or have PSW key mask 0-7 authority to connect to the requested WLM services. **Action**: Avoid requesting this function in this environment.
8	xxxx084E	**Equate Symbol**: IwmRsnCodeWlmServBadType **Meaning**: For WLM started servers, the SERVER_TYPE= is not the one used to start the server. **Action**: Specify the correct SERVER_TYPE.
8	xxxx0853	**Equate Symbol**: IwmRsnCodeWlmQmBadType **Meaning**: There is a queue manager or router environment of the specified subsystem name, but of a different type than that specified by the caller. **Action**: 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.

Table 109. Return and Reason Codes for the IWMCONN Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx0855	**Equate Symbol**: IwmRsnCodeBadNumEUMax **Meaning**: PARALLEL_EU variable is greater than the maximum of 1000. **Action**: Specify a value between 1 and 1000.
8	xxxx0856	**Equate Symbol**: IwmRsnCodeBadNumEUMin **Meaning**: PARALLEL_EU variable is less than the minimum of 1. **Action**: Specify a value between 1 and 1000.
8	xxxx085C	**Equate Symbol**: IwmRsnCodeWrongNumEU **Meaning**: Caller invoked service with a PARALLEL_EU value which is different from the PARALLEL_EU of existing servers in the application environment **Action**: Ensure that all servers in the application environment specify the same PARALLEL_EU value.
8	xxxx0873	**Equate Symbol**: IwmRsnCodeWrongSrvLmt **Meaning**: Caller invoked service with a SERVER_LIMIT parameter setting which is different from the SERVER_LIMIT of existing servers in the application environment **Action**: Ensure that all servers in the application environment specify the same SERVER_LIMIT value.
8	xxxx0874	**Equate Symbol**: IwmRsnCodeWrongMngTsk **Meaning**: Caller invoked service with a MANAGE_TASKS parameter setting which is different from the MANAGE_TASKS of existing servers in the application environment **Action**: Ensure that all servers in the application environment specify the same MANAGE_TASKS value.
8	xxxx0878	**Equate Symbol**: IwmRsnCodeBadNumLimitMax **Meaning**: 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. **Action**: Correct number or do not specify SERVER_LIMIT parameter in order to use the default.
8	xxxx0879	**Equate Symbol**: IwmRsnCodeBadNumLimitMin **Meaning**: Caller invoked service with a SERVER_LIMIT parameter setting which is less than what has been defined on the PARALLEL_EU parameter. **Action**: Ensure that SERVER_LIMIT is always greater or equal to PARALLEL_EU.
8	xxxx087A	**Equate Symbol**: IwmRsnCodeNoQServer **Meaning**: The MANAGE_TASKS parameter is not allowed when QUEUE_SERVER=YES has been specified. **Action**: Ensure to use the parameters correctly.

Table 109. Return and Reason Codes for the IWMCONN Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx088E	**Equate Symbol**: IwmRsnCodeWlmServBadSSND **Meaning**: For WLM started servers, the NODENM= is not the one used to start the server. **Action**: Specify the correct NODENM.
8	xxxx088F	**Equate Symbol**: IwmRsnCodeApplNotSSN **Meaning**: The application environment name is defined for use by a different subsystem node. **Action**: Check whether the correct application environment name is being used.
8	xxxx089E	**Equate Symbol**: IwmRsnCodeServiceAModeMismatch **Meaning**: The caller is in 64-bit address mode and tried to invoke a service macro that is only enabled for a 31-bit environment. **Action**: Use the 64-bit enabled service macro (IWM4CON) or change the address mode of the caller to 31-bit.
C	—	**Equate Symbol**: IwmRetCodeEnvError **Meaning**: Environmental error.
C	xxxx0C01	**Equate Symbol**: IwmRsnCodeNoStg **Meaning**: Storage is not available for the request. **Action**: There is a storage shortage. The function may work successfully at a later time.
C	xxxx0C09	**Equate Symbol**: IwmRsnCodeNoResmgr **Meaning**: Resource manager could not be established. **Action**: No action required. This condition may be due to a storage shortage condition.
C	xxxx0C14	**Equate Symbol**: IwmRsnCodeNoWorkShutDown **Meaning**: No work selected. Caller is to shutdown. **Action**: The server should shut down (terminate).
C	xxxx0C19	**Equate Symbol**: IwmRsnCodeNotSecAuthConnect **Meaning**: The caller is not authorized by SAF to connect to WLM with SERVER_MANAGER=YES. **Action**: The security administrator must grant access to the appropriate resource.
C	xxxx0C1A	**Equate Symbol**: IwmRsnCodeApplNotDefined **Meaning**: The application environment name is not defined in the active WLM policy. **Action**: 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.

Table 109. Return and Reason Codes for the IWMCONN Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
C	xxxx0C1B	**Equate Symbol**: IwmRsnCodeApplNotSST **Meaning**: The application environment name is defined for use by a different subsystem type in the active WLM policy. **Action**: 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.
C	xxxx0C1F	**Equate Symbol**: IwmRsnCodeServerExists **Meaning**: A server exists for the specified application environment which only allows 1 such server in the sysplex. **Action**: 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.
C	xxxx0C22	**Equate Symbol**: IwmRsnCodeApplEnvQuiesced **Meaning**: The specified application environment has been quiesced, server cannot be started for the request. **Action**: Restart the application environment and then retry the request.
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error. **Action**: No action required. The function may be successful if invoked again.

Example

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 — Disconnect 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 equivalent service, IWM4DIS, which also supports 64-bit addressing. For more information, see "IWM4DIS — Disconnect from workload management" on page 499.

Environment

The requirements for the caller are:

Minimum authorization:	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.
Dispatchable unit mode:	Task or SRB When the corresponding Connect (IWMCONN) invocation specified SERVER_MANAGER=YES, task mode.
Cross memory mode:	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.
AMODE:	31-bit
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks may be held. When the corresponding Connect (IWMCONN) 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, IWMRSNCODE_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

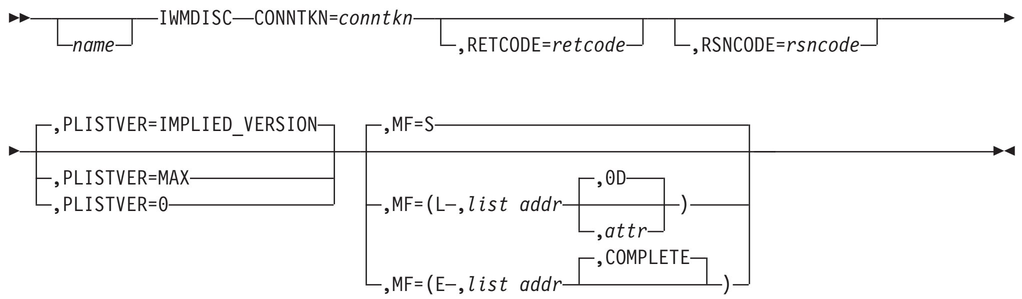
Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

Performance implications

None.

Syntax

The syntax of the IWMDISC macro is as follows:



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***,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 codes 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 110. Return and Reason Codes for the IWMDISC Macro

Return Code	Reason Code	Equate Symbol, Meaning, and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.
4	xxxx0409	**Equate Symbol**: IwmRsnCodeNoConn **Meaning**: Input connection token does not reflect an active connection to WLM. **Action**: None required.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0801	**Equate Symbol**: IwmRsnCodeSrbMode **Meaning**: The caller is in SRB mode. **Action**: Avoid requesting this function while in SRB mode.
8	xxxx0802	**Equate Symbol**: IwmRsnCodeXmemUserKeyTkn **Meaning**: The caller is in cross-memory mode while the token was obtained in a user key. **Action**: Avoid requesting this function while in cross-memory mode.
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: The caller is locked. **Action**: Avoid requesting this function while locked.

Table 110. Return and Reason Codes for the IWMDISC Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx0809	**Equate Symbol**: IwmRsnCodeSrbUserKeyTkn **Meaning**: The caller is in SRB mode, while the token was obtained in a user key (8-F). **Action**: Avoid requesting this function in SRB mode for tokens associated with user key.
8	xxxx080A	**Equate Symbol**: IwmRsnCodeTcbNotOwnerUserKeyTkn **Meaning**: Current TCB is not the owner, while the token was obtained in a user key (8-F). **Action**: Avoid requesting this function under a TCB other than the owner for a token associated with user key.
8	xxxx080B	**Equate Symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list. **Action**: 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	xxxx0810	**Equate Symbol**: IwmRsnCodeEutFrr **Meaning**: The caller has EUT FRR established. **Action**: Avoid requesting this function with an EUT FRR set.
8	xxxx0823	**Equate Symbol**: IwmRsnCodeDatoff **Meaning**: The caller invoked the service while DATOFF **Action**: Avoid requesting this function in this environment.
8	xxxx0824	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: The caller invoked the service but was in 24-bit addressing mode. **Action**: Request this function only when you are in 31-bit addressing mode.
8	xxxx0825	**Equate Symbol**: IwmRsnCodeAscModeNotPrimary **Meaning**: Caller invoked service but was not DAT on Primary ASC mode. **Action**: Avoid requesting this function in this environment.
8	xxxx0826	**Equate Symbol**: IwmRsnCodeTaskTerm **Meaning**: The caller invoked the service while task termination is in progress for the TCB associated with the owner. **Action**: Avoid requesting this function in this environment.
8	xxxx0827	**Equate Symbol**: IwmRsnCodeRsvdNot0 **Meaning**: Reserved field in parameter list was non-zero. **Action**: Check for possible storage overlay of the parameter list.

Table 110. Return and Reason Codes for the IWMDISC Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx0828	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: Version number in parameter list is not valid. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx082F	**Equate Symbol**: IwmRsnCodeWrongHome **Meaning**: The caller invoked the service from the wrong home address space. **Action**: Invoke the function with the correct home address space.
8	xxxx0841	**Equate Symbol**: IwmRsnCodeXMemMode **Meaning**: Caller is in cross-memory mode. **Action**: Invoke the function in non-cross memory mode.
8	xxxx084D	**Equate Symbol**: IwmRsnCodeNotAuthConnect **Meaning**: The caller must be supervisor state or have PSW key mask 0-7 authority to disconnect from the requested WLM services. **Action**: Avoid requesting this function in this environment.
8	xxxx089E	**Equate Symbol**: IwmRsnCodeServiceAModeMismatch **Meaning**: The caller is in 64-bit address mode and tried to invoke a service macro that is only enabled for a 31-bit environment. **Action**: Use the 64-bit enabled service macro (IWM4DIS) or change the address mode of the caller to 31-bit.
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error. **Action**: No action required. The function may be successful if invoked again.

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.

Note: It is recommended to use the enhanced service, IWM4EQRY, which also supports 64-bit addressing. For more information, see "IWM4EQRY — Query an enclave" on page 528.

Environment

The requirements for the caller are:

Minimum authorization:	Supervisor state or program key mask (PKM) allowing keys 0-7.
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31-bit
ASC mode:	Primary or access register (AR) Any P,S,H. If in Access Register ASC mode, specify SYSSTATE ASCENV=AR before invoking this macro.
Interrupt status:	Enabled for I/O and external interrupts.
Locks:	No locks may be held.
Control parameters:	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. Reason code and return code constants are defined within 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. 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.

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:

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

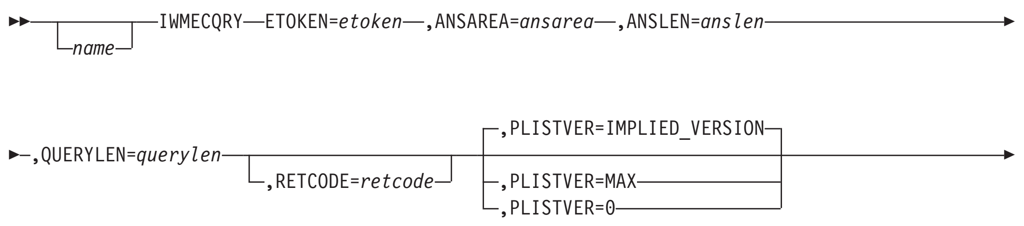
Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

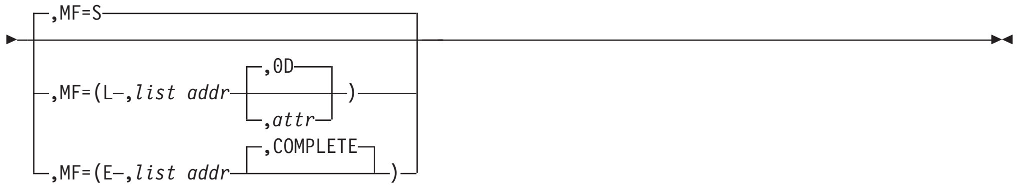
Performance implications

None.

Syntax

The syntax of the IWMECQRY macro is as follows:





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*,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 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 codes 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 111. Return and Reason Codes for the IWMECQRY Macro

Return Code	Reason Code	Equate Symbol, Meaning, and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.
4	xxxx040A	**Equate Symbol**: IwmRsnCodeOutputAreaTooSmall **Meaning**: The output area supplied is too small to receive all the available information. **Action**: None required.
4	xxxx043C	**Equate Symbol**: IwmRsnCodeIsReset **Meaning**: 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. **Action**: None required.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: The caller is locked. **Action**: Avoid requesting this function while locked.
8	xxxx080B	**Equate Symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list. **Action**: Check for possible storage overlay.

Table 111. Return and Reason Codes for the IWMECQRY Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx0824	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: The caller invoked the service but was in 24-bit addressing mode. **Action**: Request this function only when you are in 31-bit addressing mode.
8	xxxx0830	**Equate Symbol**: IwmRsnCodeBadAlet **Meaning**: The caller has passed an invalid ALET. **Action**: Check for possible storage overlay of the parameter list or variable.
8	xxxx083A	**Equate Symbol**: IwmRsnCodeBadEnclave **Meaning**: Enclave token is invalid. **Action**: Check the specification of the ETOKEN parameter.
10	—	**Equate Symbol**: IwmRetCodeCompError: **Meaning**: Component error **Action**: No action required. The function may be successful if invoked again.

IWMECREA — Create an enclave

The purpose of this service is to create an enclave where possibly multiple SRBs and/or TCBsmay 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 46.

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 equivalent service, IWM4ECRE, which also supports 64-bit addressing. For more information, see "IWM4ECRE — Create an enclave" on page 506.

Environment

The requirements for the caller are:

Minimum authorization:	Supervisor state or program key mask (PKM) allowing keys 0-7.
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31-bit
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks may be held.
Control parameters:	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, 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. 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

The syntax of the IWMECREA macro is as follows:

main diagram

```
                                  ┌─TYPE=INDEPENDENT─┐
►►──┬──────┬──IWMECREA──┬──────────────────┴─┤ parameters-1 ├─┬───────────►
    └─name─┘            ├─TYPE=DEPENDENT────────────────────────┤
                        └─TYPE=MONENV─,MONTKN=montkn─┬─,ACCESS=PRIMARY─┬┘
                                                     └─,ACCESS=HOME────┘

►─,ETOKEN=etoken──┬──────────────────┬──┬──────────────────┬──────────────►
                  └─,RETCODE=retcode─┘  └─,RSNCODE=rsncode─┘

  ┌─,PLISTVER=IMPLIED_VERSION─┐  ┌─,MF=S──────────────────────────────┐
►─┴───────────────────────────┴──┼────────────────────────────────────┼──►◄
  ├─,PLISTVER=MAX─────────────┤  │                    ┌─,0D───┐       │
  ├─,PLISTVER=0───────────────┤  ├─,MF=(L─,list addr──┴───────┴─)─────┤
  ├─,PLISTVER=1───────────────┤  │                    └─,attr─┘       │
  ├─,PLISTVER=2───────────────┤  │                    ┌─,COMPLETE─┐   │
  └─,PLISTVER=3───────────────┘  └─,MF=(E─,list addr──┴───────────┴─)─┘
```

parameters-1

```
►►──,CLSFY=clsfy──,ARRIVALTIME=arrivaltime──,FUNCTION_NAME=function_name──────►

  ┌─,EXSTARTDEFER=NO──┐   ┌─,ESTRT=IMPLIED─┐
►─┴───────────────────┴───┼────────────────┴─┬────────────────────────────┬┬──►◄
  └─,EXSTARTDEFER=YES─┘   │                  └─,WORKREQ_HDL=workreq_hdl─┘│
                          ├─,ESTRT=EXPLICIT───────────────────────────────┤
                          ├─,ESTRT=EXPLICIT_SINGLE────────────────────────┤
                          └─,ESTRT=NEVER──────────────────────────────────┘
```

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.

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 IWMCONN 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 EWLMMODE=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=YES
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*,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 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**, which supports all parameters except those specifically referenced in higher versions.
- **1**, which supports the following parameters and those from version 0 :

ACCESS	MONTKN	TYPE

- **2**, which supports the following parameter and those from version 0 and 1:

  EXSTARTDEFER

- **3**, which supports 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 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 codes 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 112. Return and Reason Codes for the IWMECREA Macro

Return Code	Reason Code	Equate Symbol, Meaning, and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: Caller is locked. **Action**: Avoid requesting this function while locked.
8	xxxx080B	**Equate Symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list. **Action**: 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	**Equate Symbol**: IwmRsnCodeBadMonEnv **Meaning**: Monitoring environment does not pass short form verification. **Action**: Check for possible storage overlay.
8	xxxx0821	**Equate Symbol**: IwmRsnCodeBadConn **Meaning**: Connect token from the input classify parameter list does not pass validity checking. **Action**: Check for possible storage overlay.
8	xxxx0824	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: Caller invoked service but was in 24 bit addressing mode. **Action**: Request this function only when you are in 31 bit addressing mode.
8	xxxx0828	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: Version number in parameter list is not valid. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx0836	**Equate Symbol**: IwmRsnCodeMaxEnclave **Meaning**: Enclave could not be created because the enclave limit has been reached. **Action**: Check for possible problems wherein enclaves are not being deleted as expected or excessive numbers of enclaves are being created in a loop.

Table 112. Return and Reason Codes for the IWMECREA Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx0837	**Equate Symbol**: IwmRsnCodeUserKeyConntkn **Meaning**: Connect token from the input classify parameter list is associated with a user key. **Action**: Invoke the function with a token associated with a system key.
8	xxxx0838	**Equate Symbol**: IwmRsnCodeClsfyAreaTooBig **Meaning**: Input area associated with classification information is larger than supported. **Action**: Invoke the function with an area of the proper size. Check for possible storage overlay.
8	xxxx0839	**Equate Symbol**: IwmRsnCodeClsfyPlTooSmall **Meaning**: Input Classify parameter list is too small. **Action**: Invoke the function with an area of the proper size. Check for possible storage overlay.
8	xxxx083B	**Equate Symbol**: IwmRsnCodeHomeNotOwnConn **Meaning**: Home address space does not own the connect token from the input classify parameter list. **Action**: Invoke the function with the correct home address space.
8	xxxx0840	**Equate Symbol**: IwmRsnCodeServiceNotEnabled **Meaning**: Caller's space connection is not enabled for this service. **Action**: Avoid requesting this function under the input connection. IWMCONN options must be specified previously to enable this service.
8	xxxx085D	**Equate Symbol**: IwmRsnCodeMonenvNotHome **Meaning**: The input monitoring environment is related to an address space other than home. **Action**: None required.
8	xxxx0892	**Equate Symbol**: IwmRsnCodeEWLMCorrNotAllowed **Meaning**: Passed classification information must not contain an EWLM correlator (EWLM_CORR). **Action**: Do not pass an EWLM correlator within the classification information when invoking the service with ESTRT=EXPLICIT.
8	xxxx0894	**Equate Symbol**: IwmRsnCodeInvalidEWLMCorr **Meaning**: 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'). **Action**: Check the specification of the EWLM correlator in the classification information.

Table 112. Return and Reason Codes for the IWMECREA Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx0895	**Equate Symbol**: IwmRsnCodeEWLMServNotEnabled **Meaning**: Service is not enabled because caller invoked the IWMCONN service with EWLM=NO. **Action**: Specify the parameter WORKREQ_HDL only when connected with EWLM=YES.
8	xxxx089E	**Equate Symbol**: IwmRsnCodeServiceAModeMismatch **Meaning**: The caller is in 64-bit address mode and tried to invoke a service macro that is only enabled for a 31-bit environment. **Action**: Use the 64-bit enabled service macro (IWM4ECRE) or change the address mode of the caller to 31-bit.
C	—	**Equate Symbol**: IwmRetCodeEnvError **Meaning**: Environmental error.
C	xxxx0C01	**Equate Symbol**: IwmRsnCodeNoStg **Meaning**: Storage is not available for the request. **Action**: There is a storage shortage. The function may work successfully at a later time.
C	xxxx0C0C	**Equate Symbol**: IwmRsnCodeClassifyFail **Meaning**: Received a non-zero return code from the classification service, IWMCLSFY. **Action**: No action required. Reinvoking the function later may succeed.
C	xxxx0C0D	**Equate Symbol**: IwmRsnCodeBadClsfy **Meaning**: Classification apparently can not access the current policy, possibly due to a policy switch in progress. **Action**: Invoke the function when the conditions are alleviated.
C	xxxx0C20	**Equate Symbol**: IwmRsnCodeDepClassifyFail **Meaning**: Unable to obtain classification attributes for a dependent enclave. **Action**: None required.
C	xxxx0C21	**Equate Symbol**: IwmRsnCodeNoMonEnvErr **Meaning**: Input monitoring token indicates no monitoring environment was established. **Action**: None required.
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error. **Action**: No action required. The function may be successful if invoked again.

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 equivalent service, IWM4EDEL, which also supports 64-bit addressing. For more information, see "IWM4EDEL — Delete an enclave" on page 520.

Environment

The requirements for the caller are:

Minimum authorization:	Supervisor state or program key mask (PKM) allowing keys 0-7.
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31-bit
ASC mode:	Primary or access register (AR)
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks may be held. FRR environments may be established.
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, 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 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

The syntax of the IWM4EDEL macro is as follows:

```
►►──┬──────┬──IWM4EDEL──ETOKEN=etoken──┬──────────────────────────┬──────────────────────►
    └─name─┘                           └─,CPUSERVICE=cpuservice─┘

►──┬────────────────────────────────┬──┬──────────────────┬──────────────────────────────►
   └─,SYSPLEXCPUSRV=sysplexcpusrv─┘  └─,CPUTIME=cputime─┘

►──┬──────────────────────────────────┬──┬────────────────────────────┬──────────────────►
   └─,RESPTIME_RATIO=resptime_ratio─┘  └─,ZAAPSERVICE=zaapservice─┘

►──┬────────────────────┬──┬──────────────────────────┬──┬────────────────────────────┬──►
   └─,ZAAPTIME=zaaptime─┘  └─,ZAAPNFACTOR=zaapnfactor─┘  └─,ZIIPSERVICE=ziipservice─┘

►──┬────────────────────┬──┬──────────────────┬──┬──────────────────┬────────────────────►
   └─,ZIIPTIME=ziiptime─┘  └─,RETCODE=retcode─┘  └─,RSNCODE=rsncode─┘

   ┌─,PLISTVER=IMPLIED_VERSION─┐  ┌─,MF=S──────────────────────────────────┐
►──┼───────────────────────────┼──┼────────────────────────────────────────┼──────────►◄
   ├─,PLISTVER=MAX─────────────┤  │                        ┌─,0D───┐       │
   ├─,PLISTVER=0───────────────┤  ├─,MF=(L─,list addr──────┼───────┼─)─────┤
   ├─,PLISTVER=1───────────────┤  │                        └─,attr─┘       │
   └─,PLISTVER=2───────────────┘  │                        ┌─,COMPLETE─┐   │
                                  └─,MF=(E─,list addr──────┴───────────┴─)─┘
```

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.

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 the following parameter and those from version 0:

  RESPTIME_RATIO
- **2**, which supports 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: act.resp.time / goal * 100 if the enclave has a response time goal (limited to: 1<=RESPTIME_RATIO<=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 incorrect 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 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 codes 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 113. Return and Reason Codes for the IWM4EDEL Macro

Return Code	Reason Code	Equate Symbol, Meaning, and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.
4	xxxx0411	**Equate Symbol**: IwmRsnCodeEnclActive **Meaning**: Input enclave had 1 or more SRBs scheduled or running, or 1 or more TCBs joined to the enclave. **Action**: None required.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: Caller is locked. **Action**: Avoid requesting this function while locked.
8	xxxx080B	**Equate Symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list. **Action**: Check for possible storage overlay.
8	xxxx0824	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: Caller invoked service but was in 24 bit addressing mode. **Action**: Request this function only when you are in 31 bit addressing mode.
8	xxxx0828	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: Version number in parameter list is not valid. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx083A	**Equate Symbol**: IwmRsnCodeBadEnclave **Meaning**: Enclave token does not pass verification. **Action**: Check for possible storage overlay of the enclave token, or asynchronous events which may have deleted the enclave.
8	xxxx0872	**Equate Symbol**: IwmRsnCodeForeignEnclave **Meaning**: The enclave is foreign. **Action**: Use the IWMUIMPT macro to delete a foreign enclave.

Table 113. Return and Reason Codes for the IWM4EDEL Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error. **Action**: 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
```

IWMEQRY — Enclave query

This service extends service IWMECQRY, it offers three functions:
1. query the classification attributes of an enclave,
2. query WLM performance management information of an enclave,
3. both of the above.

The output of this service is mapped by macro IWMECDX.

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.

Environment

The requirements for the caller are:

Minimum authorization:	Supervisor state or program key mask (PKM) allowing keys 0-7.
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31-bit
ASC mode:	Primary or access register (AR) If in Access Register ASC mode, specify SYSSTATE ASCENV=AR before invoking this macro.
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks may be held.
Control parameters:	Control parameters must be in the primary address space.

Programming requirements

1. This macro may only be used on z/OS V1.R10 (HBB7750) or higher with APAR OA35822 applied.
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. This macro may not be used prior to the completion of WLM address space initialization

2. The caller must provide storage for an answer area mapped by macro IWMECDX. 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 IWMEQRY 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.

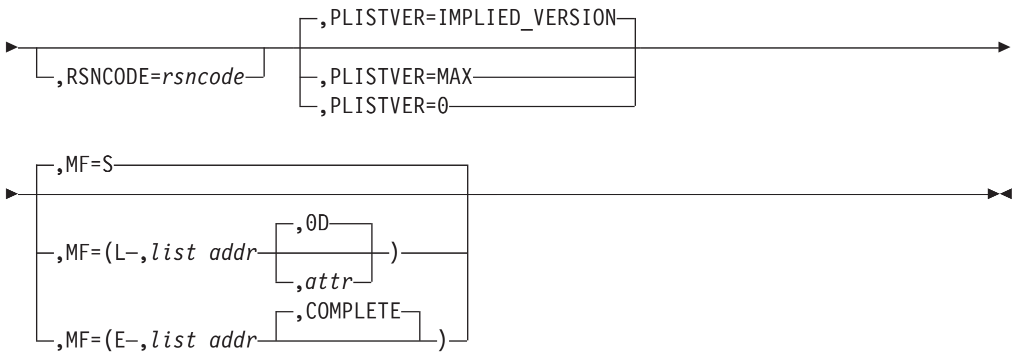
Performance implications

None.

Syntax

main diagram

```
►►─┬──────┬─b─IWMEQRY─b─ETOKEN=etoken─,ANSAREA=ansarea─,ANSLEN=anslen───►
   └─name─┘

►─,QUERYLEN=querylen─┬─,FUNCTION=CLASSINFO─┬─┬──────────────────┬──────►
                     ├─,FUNCTION=PERFINFO──┤ └─,RETCODE=retcode─┘
                     └─,FUNCTION=ALL───────┘
```



Parameters

The parameters are explained as follows:

name

An optional symbol, starting in column 1, that is the name on the IWMEQRY macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

,ANSAREA=*ansarea*

A required output parameter, which is to receive the data being returned. The layout of this area is defined by macro IWMECDX.

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 IWM4EQRY.

To code: Specify the RS-type address, or address in register (2)-(12), of a fullword field, or specify a literal decimal value.

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.

,FUNCTION=CLASSINFO
,FUNCTION=PERFINFO
,FUNCTION=ALL

A required parameter that indicates the query function to be executed.

,FUNCTION=CLASSINFO

Use this function to query the classification attributes of an enclave. This is the same information as is returned by service IWMECQRY.

,FUNCTION=PERFINFO

Use this function to query the WLM performance management information of an enclave. This data is based on the classification attributes and the active WLM Policy.

,FUNCTION=ALL

Use this function to query both, the classification attributes and the WLM performance management information of an enclave.

,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

,QUERYLEN=*querylen*
A required output parameter, variable is to receive the number of bytes needed to contain the data being returned by IWMEQRY. The length of the area needed to contain the data is dependent on the Function being used. If the ANSLEN is less than the QUERYLEN, then no data is returned in the output area specified by ANSAREA and a return code of 4 is issued.

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. If you specify 15, GPR15, REG15, or R15 (within or without parentheses), the value will be left in GPR 15.

To code: Specify the RS-type address of a fullword field, or register (2)-(12) or (15), (GPR15), (REG15), or (R15).

,RSNCODE=*rsncode*
An optional output parameter into which the reason code is to be copied from GPR 0. If you specify 0, 00, GPR0, GPR00, REG0, REG00, or R0 (within or without parentheses), the value will be left in GPR 0.

To code: Specify the RS-type address of a fullword field, or register (0) or (2)-(12), (00), (GPR0), (GPR00), REG0), (REG00), or (R0).

ABEND codes

None.

Return codes and reason codes

When the IWMEQRY 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 114. Return and Reason Codes for the IWMEQRY Macro

Return Code	Reason Code	Equate Symbol Meaning and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.

Table 114. Return and Reason Codes for the IWMEQRY Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
4	xxxx040A	**Equate Symbol**: IwmRsnCodeOutputAreaTooSmall **Meaning**: The output area supplied is too small to receive all the available information. **Action**: Reinvoke the service with an output area of sufficient size to receive all information.
4	xxxx043C	**Equate Symbol**: IwmRsnCodeIsReset **Meaning**: 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. Information returned. **Action**: None required.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: Caller is locked. **Action**: Avoid requesting this function while locked.
8	xxxx080B	**Equate Symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list. **Action**: Check for possible storage overlay.
8	xxxx0824	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: Caller invoked service but was in 24 bit addressing mode. **Action**: Request this function only when you are in 31-bit or 64-bit addressing mode.
8	xxxx0828	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: Caller invoked service with an invalid value for PLISTVER. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx0829	**Equate Symbol**: IwmRsnCodeBadOptions **Meaning**: Parameter list omits required parameters or supplies mutually exclusive parameters or provides data associated with options not selected. **Action**: Check for possible invalid input data in the parameter list.

Table 114. Return and Reason Codes for the IWMEQRY Macro (continued)

Return Code	Reason Code	Equate Symbol Meaning and Action
8	xxxx0830	**Equate Symbol**: IwmRsnCodeBadAlet **Meaning**: Caller has an invalid ALET. The ALET is used to address the output area specified in parameter ANSAREA. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx083A	**Equate Symbol**: IwmRsnCodeBadEnclave **Meaning**: Enclave token is invalid. **Action**: Check the specification of the ETOKEN parameter.
8	xxxx089E	**Equate Symbol**: IwmRsnCodeServiceAModeMismatch **Meaning**: Caller is in an addressing mode incompatible with the invoked service. **Action**: Check the specification of the caller's allowed AMODE.
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error **Action**: Consider reporting the problem to IBM.

Example

None.

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:

Minimum authorization:	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.
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any HASN, any SASN
AMODE:	31-bit
ASC mode:	Primary mode
Interrupt status:	Enabled
Locks:	Unlocked
Control parameters:	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:

```
►►──┬──────┬──IWMMABNL──ABNORMAL=abnormal──,MONTKN=montkn──────────────────────►
    └─name─┘

►─,MONENVKEYP=─┬─VALUE───,MONENVKEY=monenvkey──┬─,MONENV=─┬─NOSWITCH──┬────────►
               ├─PSWKEY────────────────────────┤          └─SECONDARY─┘
               └─UNKNOWN───────────────────────┘

►──┬──────────────────┬──┬──────────────────┬──────────────────────────────────►◄
   └─,RETCODE=retcode─┘  └─,RSNCODE=rsncode─┘
```

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 IWMMABNL, for example - IWMMABNL_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.

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 codes 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		**Meaning**: Successful completion. **Action**: None.
04	0402	**Meaning**: Warning. Input monitoring token indicates no monitoring environment was established. **Action**: None.
08	0820	**Meaning**: 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`.

IWMMCHST — Monitor 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 equivalent service, IWM4MCHS, which also supports 64-bit addressing. For more information, see "IWM4MCHS — Change the state of a work request" on page 548.

Environment

The requirements for the caller are:

Minimum authorization:	Problem state. PSW key must either be 0 or match the value supplied on IWMMCREA.
Dispatchable unit mode:	Task or SRB
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:	Suspend locks are allowed.
Control parameters:	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 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 IWMMCHST 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

The syntax of the IWMMCHST macro is as follows:

```
►►──┬──────┬──IWMMCHST──┬─STATE=FREE──────────────────────────────────┬─,MONTKN=montkn──────►
    └─name─┘            ├─STATE=ACTIVE────────────────────────────────┤
                        ├─STATE=ACTIVE_APPL───────────────────────────┤
                        ├─STATE=READY─────────────────────────────────┤
                        ├─STATE=IDLE──────────────────────────────────┤
                        └─STATE=WAITING──┬─,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──────────┘

   ┌─,RESTKN=NORESTKN─┐  ┌─,RUNTIME_VER=SHORT_FORM─┐  ┌─,COMPCODE=YES─┐
►──┼──────────────────┼──┼─────────────────────────┼──┼───────────────┼─────────────────────►
   └─,RESTKN=restkn───┘  └─,RUNTIME_VER=MINIMAL────┘  └─,COMPCODE=NO──┘

                                                             ┌─,COMPLETE─┐
►──┬─────────────────────┬──┬─────────────────────┬─,MF=(M─,list addr─┼───────────┼─)──────────►◄
   └─,RETCODE=retcode────┘  └─,RSNCODE=rsncode────┘                  └─,NOCHECK──┘
```

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 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.

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 codes 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 115. Return and Reason Codes for the IWM4MCHS Macro

Return Code	Reason Code	Equate Symbol, Meaning, and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.
4	xxxx0402	**Equate Symbol**: IwmRsnCodeNoMonEnv **Meaning**: Input monitoring token indicates no monitoring environment was established. **Action**: None required.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0820	**Equate Symbol**: IwmRsnCodeBadMonEnv **Meaning**: Monitoring environment does not pass short form verification. **Action**: 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 equivalent service, IWM4MCRE, which also supports 64-bit addressing. For more information, see "IWM4MCRE — Create delay monitoring environment" on page 559.

Environment

The requirements for the caller are:

Minimum authorization:	Supervisor state or program key mask (PKM) allowing keys 0-7.
Dispatchable unit mode:	Task
Cross memory mode:	Non-XMEM or XMEM. Any P.S.H.
AMODE:	31-bit
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts No (EUT) FRR established.
Locks:	No locks may be held.
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, 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 IWMMCREA 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 IWMMCREA, 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 IWMMCREA. This condition may be satisfied by ensuring that the then current primary matches primary at the time that IWMMCREA 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 resouces 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

The syntax of the IWMMCREA macro is as follows:

main diagram

```
►►──┬──────┬──IWMMCREA──────────────────────────────────────────────────────────►
    └─name─┘

    ┌─REQTYPE=SINGLE──┐
►───┴─────────────────┴─,MONTKN=montkn──────────────────────────────────────────►
    └─REQTYPE=MULTIPLE─,AMOUNT=amount─,MONTKN_LIST=montkn_list─,MONTKN_LISTLEN=montkn_listlen─┘

    ┌─,REPORTONLY=NO─┐ ┌─,BPMGMTONLY=NO──┐
►───┼────────────────┴─┼─────────────────┼──┬─,SUBSYSP=CONNECT─,CONNTKN=conntkn──────────►
    │                  └─,BPMGMTONLY=YES─┘  │ └─,SUBSYSP=VALUE─┤ parameters-1 ├─┘
    └─,REPORTONLY=YES───────────────────────┘

►──┬─,MONTKNKEYP=VALUE─,MONTKNKEY=montknkey─┬──┬──────────────────┬──┬──────────────────┬──►
   └─,MONTKNKEYP=PSWKEY─────────────────────┘  └─,RETCODE=retcode─┘  └─,RSNCODE=rsncode─┘

    ┌─,PLISTVER=IMPLIED_VERSION─┐  ┌─,MF=S──────────────────────────────┐
►───┼───────────────────────────┼──┼────────────────────────────────────┼───────────►◄
    ├─,PLISTVER=MAX─────────────┤  │                    ┌─,0D───┐       │
    ├─,PLISTVER=0───────────────┤  ├─,MF=(L─,list addr──┼───────┼─)─────┤
    └─,PLISTVER=1───────────────┘  │                    └─,attr─┘       │
                                   │                    ┌─,COMPLETE─┐   │
                                   └─,MF=(E─,list addr──┴───────────┴─)─┘
```

parameters-1

```
►►──,SUBSYS=subsys──,SUBSYSNM=subsysnm──────────────────────────────────────────►

   ┌─,EWLM=NO─────────────────────────────────────────────────────────────┐
►──┼──────────────────────────────────────────────────────────────────────┼──────►◄
   └─,EWLM=YES──┬─────────────────────────────────────────────────────────┤
                └─┬─,GROUPNM=NO_GROUPNM─┬─,GROUPNM_LEN=groupnm_len───────┘
                  └─,GROUPNM=groupnm────┘
```

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.

,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 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 codes and reason codes

When the IWMMCREA 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 116. Return and Reason Codes for the IWMMCREA Macro

Return Code	Reason Code	Equate Symbol, Meaning, and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.
4	xxxx0409	**Equate Symbol**: IwmRsnCodeNoConn **Meaning**: 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. **Action**: None required.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0801	**Equate Symbol**: IwmRsnCodeSrbMode **Meaning**: Caller is in SRB mode. **Action**: Avoid requesting this function while in SRB mode.
8	xxxx0802	**Equate Symbol**: IwmRsnCodeXmemUserKeyTkn **Meaning**: Caller is in cross-memory mode while the token was requested in user key. **Action**: Avoid requesting this function while in cross-memory mode.

Table 116. Return and Reason Codes for the IWMMCREA Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: Caller is locked. **Action**: Avoid requesting this function while locked.
8	xxxx080B	**Equate Symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list. **Action**: Check for possible storage overlay.
8	xxxx0810	**Equate Symbol**: IwmRsnCodeEutFrr **Meaning**: Caller has EUT FRR established. **Action**: Avoid requesting this function with an EUT FRR set.
8	xxxx0821	**Equate Symbol**: IwmRsnCodeBadConn **Meaning**: Input connect token does not pass validity checking. **Action**: Check for possible storage overlay.
8	xxxx0823	**Equate Symbol**: IwmRsnCodeDatoff **Meaning**: Caller invoked service while DATOFF **Action**: Avoid requesting this function in this environment.
8	xxxx0824	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: Caller invoked service but was in 24 bit addressing mode. **Action**: Request this function only when you are in 31 bit addressing mode.
8	xxxx0825	**Equate Symbol**: IwmRsnCodeAscModeNotPrimary **Meaning**: Caller invoked service but was not DAT on Primary ASC mode. **Action**: Avoid requesting this function in this environment.
8	xxxx0826	**Equate Symbol**: IwmRsnCodeTaskTerm **Meaning**: Caller invoked service while task termination is in progress for the TCB associated with the owner. **Action**: Avoid requesting this function in this environment.
8	xxxx0827	**Equate Symbol**: IwmRsnCodeRsvdNot0 **Meaning**: Reserved field in parameter list was non-zero. **Action**: Check for possible storage overlay of the parameter list.

Table 116. Return and Reason Codes for the IWMMCREA Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx0828	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: Version number in parameter list is not valid. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx0829	**Equate Symbol**: IwmRsnCodeBadOptions **Meaning**: Parameter list omits required parameters or supplies mutually exclusive parameters or provides data associated with options not selected. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx0844	**Equate Symbol**: IwmRsnCodeBadMonTknListLen **Meaning**: The storage area length specified on the MONTKN_LISTLEN parameter is not large enough to contain the data being returned. No data is returned. **Action**: Invoke the function with an output area sufficient to receive the data.
C	—	**Equate Symbol**: IwmRetCodeEnvError **Meaning**: Environmental error.
C	xxxx0C01	**Equate Symbol**: IwmRsnCodeNoStg **Meaning**: Storage is not available for the request. **Action**: There is a storage shortage. The function may work successfully at a later time.
C	xxxx0C09	**Equate Symbol**: IwmRsnCodeNoResmgr **Meaning**: Resource manager could not be established. **Action**: No action required. This condition may be due to a storage shortage condition.
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error. **Action**: No action required. The function may be successful if invoked again.

IWMMDELE — Delete the 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:

Minimum authorization:	Supervisor state, or program key mask (PKM) allowing keys 0 - 7.
Dispatchable unit mode:	Task or SRB
Cross memory mode:	Any PASN, any SASN. HASN must match the HASN when IWM4MCRE was invoked.
AMODE:	31-bit
ASC mode:	Primary
Interrupt status:	Enabled
Locks:	Unlocked, but FRRs may be established.
Control parameters:	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:

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 IWMMDELE macro is as follows:

```
►►─┬──────┬─IWMMDELE──MONTKN=montkn─┬──────────────────┬─┬──────────────────┬─►
   └─name─┘                         └─,RETCODE=retcode─┘ └─,RSNCODE=rsncode─┘

   ┌─,MF=S───────────────────────────────┐
►──┼─────────────────────────────────────┼─────────────────────────────────►◄
   │                  ┌─,0D─────────┐    │
   ├─,MF=(L,─MFCTRL───┼─────────────┼─)──┤
   │                  └─,mfattr─────┘    │
   │                  ┌─,COMPLETE─┐      │
   └─,MF=(E,─MFCTRL───┼───────────┼─)────┘
                      └─,complete─┘
```

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 codes and reason codes

When IWMMDELE 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		**Meaning**: Successful completion.

Hexadecimal Return Code	Hexadecimal Reason Code	Meaning
04	0402	**Meaning**: Warning. Input monitoring token indicates no monitoring environment was established.
04	0403	**Meaning**: Warning. Input monitoring token does not reflect an allocated monitoring environment owned by the current home address space.
08	0802	**Meaning**: Program error. The caller is in cross-memory mode while the token was obtained in user key.
08	0803	**Meaning**: Program error. Caller is disabled.
08	0804	**Meaning**: Program error. The caller is locked.
08	0805	**Meaning**: Program error. Input monitoring token reflects a switch continuation.
08	0806	**Meaning**: Program error. Input monitoring token reflects a continuation to a dependent monitoring environment.
08	0808	**Meaning**: Program error. Input monitoring token reflects a continuation from a parent monitoring environment.
08	0809	**Meaning**: Program error. The caller is in SRB mode, while the token was obtained in a user key (8-F).
08	080A	**Meaning**: Program error. Current task is not the current owner, while the token was obtained in a user key (8-F).
08	080B	**Meaning**: Program error. Error accessing parameter list.
08	0823	**Meaning**: Program error. The caller invoked the service while dynamic address translation was disabled.
08	0824	**Meaning**: Program error. The caller invoked the service but was in 24-bit addressing mode.
08	0825	**Meaning**: Program error. The caller invoked the service but was not in Primary ASC mode.
08	0826	**Meaning**: Program error. The caller invoked the service while task termination is in progress for the task associated with the owner.
08	0827	**Meaning**: Program error. Reserved field in parameter list was non-zero.
08	0828	**Meaning**: Program error. Version number in parameter list is not valid.
08	082A	**Meaning**: 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
```

IWMMEXTR — Monitoring environment extract

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, user ID 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.

The EWLM_S_CURCORR keyword should only be specified, if a work unit has been started by IWMMSTRT.

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:

Minimum authorization:	Either problem state or supervisor state. Any PSW key.
Dispatchable unit mode:	Task or SRB
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:	Suspend locks are allowed.
Control parameters:	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. The 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 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 IWMMEXTR 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

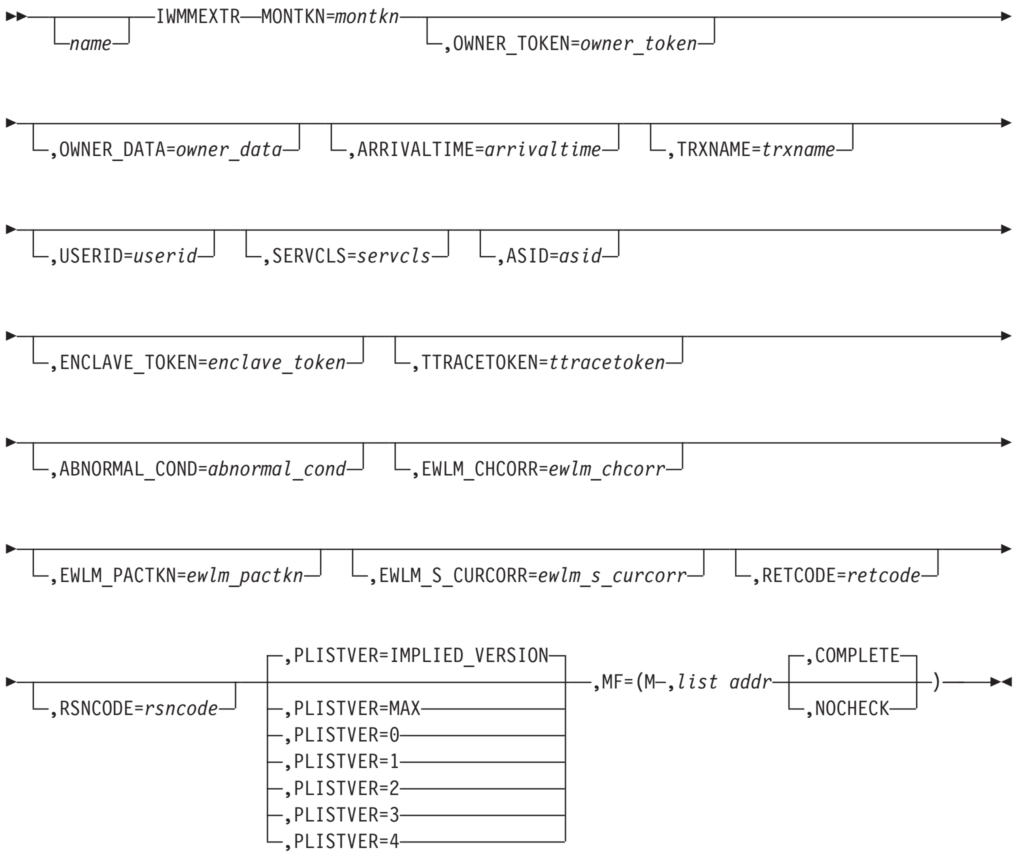
Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

Performance implications

None.

Syntax

The syntax of the IWMMEXTR macro is as follows:



Parameters

The parameters are explained as follows:

name
: An optional symbol, starting in column 1, that is the name on the IWMMEXTR 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 IWMMXFER 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.

,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.

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.

,EWLM_S_CURCORR=*ewlm_s_curcorr*
An optional output parameter, which contains the current correlator of the work unit started by IWMMSTRT. Normally this correlator is different from the child correlator of the work request created by IWM4MINI.

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 the following parameter and those from version 0:

  TTRACETOKEN
- **2**, which supports the following parameters and those from version 0 and 1:

  ASID
  ENCLAVE_TOKEN
- **3**, which supports the following parameters and those from version 0, 1, and 2:

  EWLM_CHCORR
  EWLM_PACTKN
- **4**, which supports the following parameter and those from version 0, 1, 2, and 3:

  EWLM_S_CURCORR

To code: Specify one of the following:

- IMPLIED_VERSION
- MAX
- A decimal value of 0, 1, 2, 3, or 4

,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.

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 user ID 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 codes and reason codes

When the IWMMEXTR 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 117. Return and Reason Codes for the IWMMEXTR Macro

Return code	Reason code	Equates symbol, meaning, and action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.
4	xxxx0402	**Equate Symbol**: IwmRsnCodeNoMonEnv **Meaning**: Monitoring token indicates that no monitoring environment exists. **Action**: None required.
4	xxxx040C	**Equate Symbol**: IwmRsnCodeMonEnvLacksInfo **Meaning**: Input monitoring environment does not contain the necessary information to return the data requested. **Action**: None required.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0820	**Equate Symbol**: IwmRsnCodeBadMonEnv **Meaning**: Monitoring environment does not pass short form verification. **Action**: Check for possible storage overlay.
8	xxxx0895	**Equate Symbol**: IwmRsnCodeEWLMServNotEnabled **Meaning**: Service is not enabled because monitoring environment can not be associated with EWLM work requests. **Action**: 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.
8	xxxx08AC	**Equate Symbol**: IwmRsnCodeTranNotStarted **Meaning**: A work unit has not been started. **Action**: Start a work unit by IWMMSTRT macro, before specifying the EWLM_S_CURCORR parameter.

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 user ID, 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 IWMMINIT 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 equivalent service, IWM4MINI, which also supports 64-bit addressing. For more information, see “IWM4MINI — Monitoring environment initialization” on page 590.

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 IWMMCREA.

Dispatchable unit mode:	Task or SRB
Cross memory mode:	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.
AMODE:	31-bit
ASC mode:	Primary Any P,S,H.
Interrupt status:	Enabled for I/O and external interrupts
Locks:	Locked or unlocked
Control parameters:	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 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.
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.

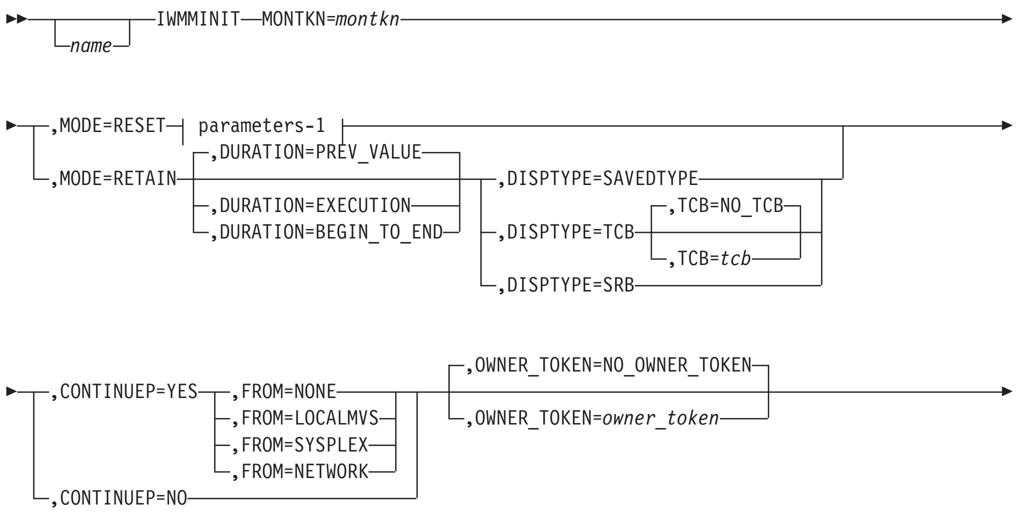
Performance implications

None.

Syntax

The syntax of the IWMMINIT macro is as follows:

main diagram



```
   ┌─,OWNER_DATA=NO_OWNER_DATA─┐
►──┼───────────────────────────┼──────────────────────────────────────────────────────►
   └─,OWNER_DATA=owner_data────┘

     ┌─,REPORTONLY=NO─┐
►────┴────────────────┴──┤ parameters-2 ├─────────────────────────────────────────────►
   └─,REPORTONLY=YES──┬─,ASSOCIATE=ENCLAVE─,ENCLAVETOKEN=enclavetoken──┤
                      └─,ASSOCIATE=ADDRESS_SPACE─,ASID=asid──────────┘

                      ┌─,TRXNAME=NO_TRXNAME─┐   ┌─,USERID=NO_USERID─┐
►──┬─,SCOPE=SHARED─┬──┼─────────────────────┼───┼───────────────────┼──────────────────►
   └─,SCOPE=SINGLE─┘  └─,TRXNAME=trxname────┘   └─,USERID=userid────┘

   ┌─,TRXCLASS=NO_TRXCLASS─┐  ┌─,TTRACETOKEN=NO_TTRACETOKEN─┐  ┌─,SOURCELU=NO_SOURCELU─┐
►──┼───────────────────────┼──┼─────────────────────────────┼──┼───────────────────────┼──►
   └─,TRXCLASS=trxclass────┘  └─,TTRACETOKEN=ttracetoken────┘  └─,SOURCELU=sourcelu────┘

►──┬────────────────────────────────────────────────────────────────────────────┬──────►
   └──┬─,LU62TKN=NO_LU62TKN─┬──┬─,LU62TKN_FMT=LU_NO_CC_27─────────────────────┤
      └─,LU62TKN=lu62tkn────┘  ├─,LU62TKN_FMT=FULL_LU_NO_CC_27────────────────┤
                               ├─,LU62TKN_FMT=FULL_LU_0_CC_28─────────────────┤
                               ├─,LU62TKN_FMT=FULL_LU_CC_36───────────────────┤
                               └─,LU62TKN_FMT=OTHER─,LU62TKN_LEN=lu62tkn_len──┘

                                              ┌─,PLISTVER=IMPLIED_VERSION─┐
►──┬───────────────────┬──┬───────────────────┬──┼───────────────────────────┼─────────►
   └─,RETCODE=retcode──┘  └─,RSNCODE=rsncode──┘  ├─,PLISTVER=MAX─────────────┤
                                              ├─,PLISTVER=0───────────────┤
                                              └─,PLISTVER=1───────────────┘

                           ┌─,COMPLETE─┐
►─,MF=(M─,list addr────────┼───────────┼─)────────────────────────────────────────────►◄
                           └─,NOCHECK──┘
```

parameters-1

```
     ┌─,DURATION=EXECUTION──────┐                    ┌─,TCB=NO_TCB─┐
►►───┼──────────────────────────┼──┬─,DISPTYPE=TCB───┼─────────────┼──┬────────────────►
     └─,DURATION=BEGIN_TO_END───┘  │                 └─,TCB=tcb────┘  │
                                   └─,DISPTYPE=SRB────────────────────┘

                                                      ┌─,EWLM_PACORR=NO_EWLM_PACORR─┐
►──┬─,ARRIVALTIMEP=CURRENT──────────────────────────┬──┼─────────────────────────────┼──►
   └─,ARRIVALTIMEP=YES─,ARRIVALTIME=arrivaltime─────┘  └─,EWLM_PACORR=ewlm_pacorr────┘

   ┌─,EWLM_PACTKN=NO_EWLM_PACTKN─┐
►──┼─────────────────────────────┼────────────────────────────────────────────────────►◄
   └─,EWLM_PACTKN=ewlm_pactkn────┘
```

parameters-2

```
      ┌─,BPMGMTONLY=NO─┐                                             ┌─,SERVCLS=NO_SERVCLS─┐
►►──┬─┴────────────────┴──┬─,EXSTARTTIMEP=NO──────────────────────────┬──┼─────────────────────┼──┬──►◄
    │                     ├─,EXSTARTTIMEP=CURRENT─────────────────────┤  └─,SERVCLS=servcls────┘  │
    │                     └─,EXSTARTTIMEP=YES─,EXSTARTTIME=exstarttime┘                           │
    └─,BPMGMTONLY=YES───,ASSOCIATE=ENCLAVE─,ENCLAVETOKEN=enclavetoken─────────────────────────────┘
```

Parameters

The parameters are explained as follows:

name
: An optional symbol, starting in column 1, that is the name on the IWMMINIT 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=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 80 on page 604 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 80 on page 604 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.

,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, which 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 IWMMCREA.

,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 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 whcih 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, which 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, which 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 user ID 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, which indicates that no user ID 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 codes 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 118. Return and Reason Codes for the IWMMINIT Macro

Return Code	Reason Code	Equate Symbol, Meaning, and Action
0	—	IwmRetCodeOk: Successful completion.
4	—	IwmRetCodeWarning: Successful completion, unusual conditions noted.
4	xxxx0402	**Equate Symbol**: IwmRsnCodeNoMonEnv **Meaning**: Monitoring token indicates that no monitoring environment exists.
8	—	IwmRetCodeInvocError: Invalid invocation environment or parameters
8	xxxx081E	**Equate Symbol**: IwmRsnCodeBadLU62TknLen **Meaning**: The length byte of the LU62 token has an invalid value. Only values 1-36 (decimal) are valid.
8	xxxx0820	**Equate Symbol**: IwmRsnCodeBadMonEnv: **Meaning**: Monitoring environment does not pass short form verification.
8	xxxx0894	**Equate Symbol**: IwmRsnCodeInvalidEWLMCorr **Meaning**: 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). **Action**: Check the specification of the correlator information.

Table 118. Return and Reason Codes for the IWMMINIT Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx0895	**Equate Symbol**: IwmRsnCodeEWLMServNotEnabled **Meaning**: Service is not enabled because the Monitoring Environment was not created with EWLM=YES (either on IWMCONN or IWMMCREA). **Action**: Specify the parameter EWLM_PACORR or EWLM_PACTKN only when the Monitoring Environment was created with EWLM=YES (either on IWMCONN or IWMMCREA).
C	—	IwmRetCodeEnvError: Environmental error
C	xxxx0C07	**Equate Symbol**: IwmRsnCodeNoArrTime: **Meaning**: No arrival time was supplied to the service and STCK gave a non-zero condition code.
C	xxxx0C08	**Equate Symbol**: IwmRsnCodeNoExTime: **Meaning**: No execution start time was supplied to the service and STCK gave a non-zero condition code.

Example

```
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
```

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:

Minimum authorization:	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.
Dispatchable unit mode:	Task or SRB
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:	LOCAL lock held
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, 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).

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:

Register	**Contents**
13	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:

Register	**Contents**
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system.

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

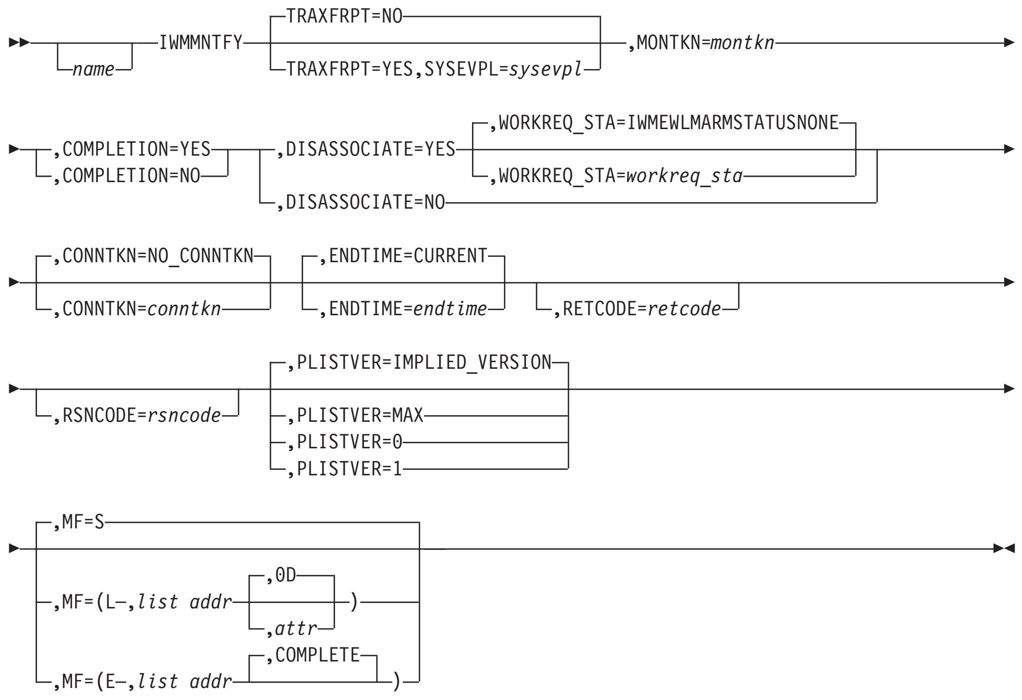
depends, the caller must save them before issuing the service, and restore them after the system returns control.

Performance implications

None.

Syntax

The syntax of the IWMMNTFY macro is as follows:



Parameters

The parameters are explained as follows:

name
: An optional symbol, starting in column 1, that is the name on the IWMMNTFY 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 IWMMNTFY 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 IWMMNTFY Completion(Yes).

The execution time, as given by the difference between the IWMMNTFY 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, which 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, which 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*,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

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 the following parameter and those from version 0:

WORKREQ_STA

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).

,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 irrevelant. 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 irrevelant. 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 the following:

- IwmEwlmArmStatusGood(0)
- IwmEwlmArmStatusAborted(1)
- IwmEwlmArmStatusFailed(2)
- IwmEwlmArmStatusUnknown(3)

These codes correspond to status codes in the OpenGroup ARM 4.0 Standard. For further information about the meaning of the status codes, refer to the ARM 4.0 Standard at http://www.opengroup.org/management/arm. The default is IWMEWLMARMSTATUSNONE. This indicates that the COMPLETION parameter value and internal information in the monitoring environment are examined to determine the status of the work request. If COMPLETION=YES is specified and no abnormal event was recorded for the monitoring environment with the IWMMABNL service, the completion status IwmEwlmArmStatusGood is reported to EWLM. If an abnormal event was reported via IWMMABNL or COMPLETION=NO was specified, the completion status IwmEwlmArmStatusFailed is reported to EWLM.

To code: Specify the RS-type address, or address in register (2)-(12), of a fullword field.

ABEND codes

None.

Return codes and reason codes

When the IWMMNTFY 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 119. Return and Reason Codes for the IWMMNTFY Macro

Return Code	Reason Code	Equate Symbol, Meaning, and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.
4	xxxx0404	**Equate Symbol**: IwmRsnCodeCompatNoSyseventRqd **Meaning**: Reserved.
4	xxxx0405	**Equate Symbol**: IwmRsnCodeGoalNoMonEnv **Meaning**: System is in goal mode but the input monitoring token indicates no monitoring environment was established, hence MVS did not receive the information. **Action**: None required.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx080C	**Equate Symbol**: IwmRsnCodeMonEnvLacksData **Meaning**: Input monitoring environment does not contain the necessary information. **Action**: Ensure that the monitoring environment was established with the necessary information.
8	xxxx080F	**Equate Symbol**: IwmRsnCodeNoUserKeyNtfy **Meaning**: User key routine not allowed to issue Notify. **Action**: Avoid requesting this function in user key.

Table 119. Return and Reason Codes for the IWMMNTFY Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx0820	**Equate Symbol**: IwmRsnCodeBadMonEnv **Meaning**: Input monitoring environment does not pass short form validity checking. **Action**: Check for possible storage overlay.
8	xxxx0821	**Equate Symbol**: IwmRsnCodeBadConn **Meaning**: Input connect token does not pass validity checking. **Action**: Check for possible storage overlay.
8	xxxx082D	**Equate Symbol**: IwmRsnCodeExStTimeGTEndTime **Meaning**: Input execution start time later than end time. **Action**: Check for possible storage overlay of the parameter list or variable.
8	xxxx0840	**Equate Symbol**: IwmRsnCodeServiceNotEnabled **Meaning**: The caller's space connection is not enabled for this service. **Action**: Avoid requesting this function under the input connection. IWM4CON options must be specified previously to enable this service.
8	xxxx087E	**Equate Symbol**: IwmRsnCodeRoMonEnv **Meaning**: Input monitoring environment is Report-Only. **Action**: Avoid requesting this function for Report-Only PBs.
8	xxxx0895	**Equate Symbol**: IwmRsnCodeEWLMServNotEnabled **Meaning**: The service is not enabled because monitoring environment cannot be associated with EWLM work requests. **Action**: 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.
8	xxxx0897	**Equate Symbol**: IwmRsnCodeTranStatusInvalid **Meaning**: Passed work request completion status is not valid. **Action**: Check the EWLM ARM interface specification for valid completion status values.
C	—	**Equate Symbol**: IwmRetCodeEnvError **Meaning**: Environmental error.

Table 119. Return and Reason Codes for the IWMMNTFY Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
C	xxxx0C02	**Equate Symbol**: IwmRsnCodeReportingSusp **Meaning**: Reserved SYSEVENT TRAXFRPT invoked, but reporting is temporarily suspended for one of the following reasons: • RMF workload activity reporting is not active • 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. **Action**: Invoke the function when the conditions are alleviated.
C	xxxx0C03	**Equate Symbol**: IwmRsnCodeSyseventNoWorkElt **Meaning**: SYSEVENT TRAXFRPT invoked, but no work element was available to save the input information. **Action**: Invoke the function when the conditions are alleviated. This condition may be due to a common storage shortage condition.
C	xxxx0C04	**Equate Symbol**: IwmRsnCodeNtfyNoWorkElt **Meaning**: Notify routine invoked, but no work element was available to save the input information. **Action**: Invoke the function when the conditions are alleviated. This condition may be due to a common storage shortage condition.
C	xxxx0C06	**Equate Symbol**: IwmRsnCodeNoEndTime **Meaning**: No end time was supplied to the service and STCK gave a non-zero condition code. **Action**: No action required.

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:

Minimum authorization:	Problem state, or supervisor state. PSW key must either be 0, or match the value specified on IWM4MCRE.
Dispatchable unit mode:	Task or SRB
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:	Unlocked when PARENT=FINDACTIVE is specified, otherwise, no restrictions.
Control parameters:	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

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

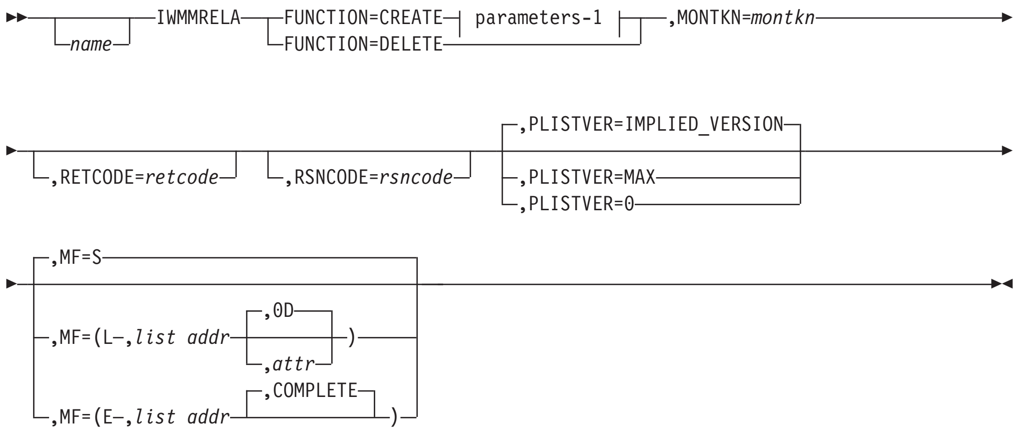
depends, the caller must save them before issuing the service, and restore them after the system returns control.

Performance implications

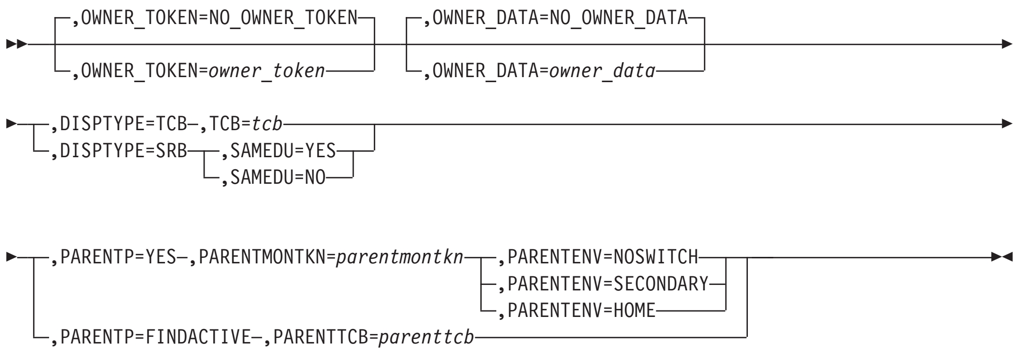
None.

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

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 codes 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.

Table 120. Return and Reason Codes for the IWMMRELA Macro

Return Code	Reason Code	Equate Symbol, Meaning, and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion.
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.
4	xxxx0402	**Equate Symbol**: IwmRsncodeNoMonEnv **Meaning**: Input monitoring token indicates no monitoring environment was established.
4	xxxx0406	**Equate Symbol**: IwmRsncodeNoParEnv **Meaning**: No parent monitoring environment was established. The input dependent monitoring environment is now related to the Home address space.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0802	**Equate Symbol**: IwmRsnCodeXmemUserKeyTkn **Meaning**: The caller is in cross-memory mode while the token was obtained in user key.
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: The caller is locked.
8	xxxx080B	**Equate Symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list.
8	xxxx081A	**Equate Symbol**: IwmRsnCodeCallerNotAuthDepEnv **Meaning**: The caller is not authorized to update the dependent monitoring environment.
8	xxxx0820	**Equate Symbol**: IwmRsnCodeBadMonEnv **Meaning**: Monitoring environment does not pass verification.
8	xxxx0822	**Equate Symbol**: IwmRsnCodeBadParEnv **Meaning**: Parent monitoring environment does not pass verification.
8	xxxx0823	**Equate Symbol**: IwmRsnCodeDatoff **Meaning**: The caller invoked the service while DATOFF.
8	xxxx0824	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: The caller invoked the service but was in 24-bit addressing mode.
8	xxxx0825	**Equate Symbol**: IwmRsnCodeAscModeNotPrimary **Meaning**: The caller invoked the service but was not DAT on Primary ASC mod.

Table 120. Return and Reason Codes for the IWMMRELA Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx0827	**Equate Symbol**: IwmRsnCodeRsvdNot0 **Meaning**: Reserved field in parameter list was non-zero
8	xxxx0828	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: The Version number in the parameter list is not valid.
8	xxxx0829	**Equate Symbol**: IwmRsnCodeBadOptions **Meaning**: Parameter list omits required parameters or supplies mutually exclusive parameters or provides data associated with options not selected.
8	xxxx087E	**Equate Symbol**: IwmRsnCodeRoMonEnv **Meaning**: Monitoring environment is report only.
8	xxxx087F	**Equate Symbol**: IwmRsnCodeRoParEnv **Meaning**: Parent monitoring environment is report only.
8	xxxx08A4	**Equate Symbol**: IwmRsnCodeBPParEnv **Meaning**: Parent monitoring environment is buffer pool management only.
10	—	**Equate Symbol**: IwmRetCodeCompError: **Meaning**: Component error

Example

To relate two monitoring environments where an address space switch is not required, specify:

```
IWMMRELA FUNCTION=CREATE,MONTKN=(R7),PARENTP=YES,
      PARENTMONTKN=(R8),PARENTENV=NOSWITCH,
      DISPTYPE=SRB,SAMEDU=YES,
      RETCODE=RCODE,RSNCODE=RSN
```

IWMMSTOP — Stop a work unit

The IWMMSTOP service allows to stop a work unit which has been started by IWMMSTRT. A work unit started by IWM4MINI is not affected by this service. The work unit is unblocked, if it is blocked at the time you issue this macro.

Environment

The requirements for the caller are:

Minimum authorization:	Supervisor state or program key mask (PKM) allowing keys 0-7.
Dispatchable unit mode:	Task or SRB
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 restrictions.
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 bits 0-31 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 before the completion of WLM address space initialization
- The caller must have issued the IWMMSTRT macro successfully.
- The caller is responsible for error recovery.
- The current PSW key must be 0 or match the key specified on IWM4MCRE provided the latter is a system key (0-7).
- The caller must serialize to prevent any delay monitoring services from being invoked concurrently for the environment represented by the monitoring token.

Input register information

Before issuing the IWMMSTOP 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. The reason code is stored in bits 0-31.
1	Used as a work register by the system.
2 - 13	Unchanged.
14	Used as a work register by the system.
15	Return code is stored in bits 0-31.

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 work registers by the system.

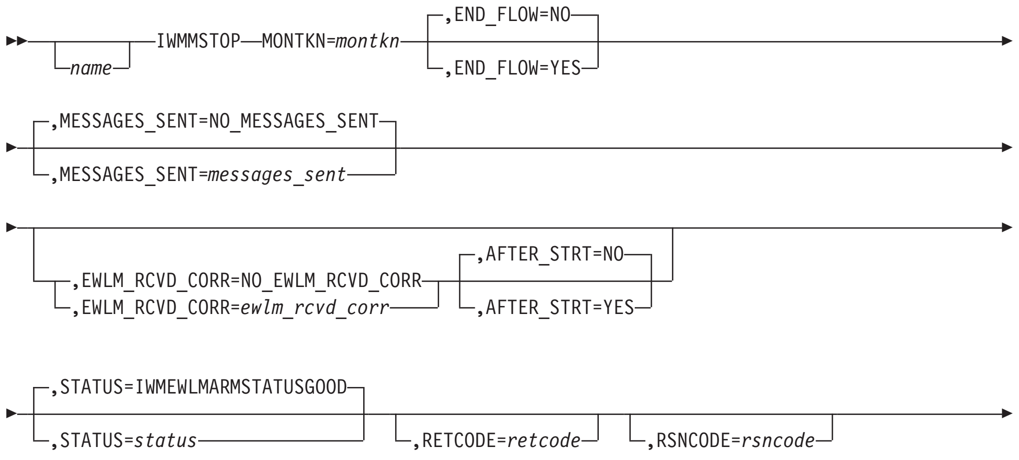
Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

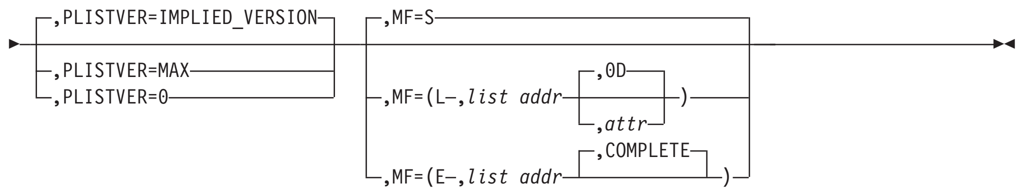
Performance implications

None.

Syntax

The syntax of the IWMMSTOP macro is as follows:


name
IWMMSTOP
MONTKN=montkn
,END_FLOW=NO
,END_FLOW=YES
,MESSAGES_SENT=NO_MESSAGES_SENT
,MESSAGES_SENT=messages_sent
,EWLM_RCVD_CORR=NO_EWLM_RCVD_CORR
,EWLM_RCVD_CORR=ewlm_rcvd_corr
,AFTER_STRT=NO
,AFTER_STRT=YES
,STATUS=IWMEWLMARMSTATUSGOOD
,STATUS=status
,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 IWMMSTOP macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

,AFTER_STRT=NO
,AFTER_STRT=YES

When EWLM_RCVD_CORR=*ewlm_rcvd_corr* is specified, an optional parameter, which indicates the moment the correlator has been received. The default is AFTER_STRT=NO.

,AFTER_STRT=NO

indicates that the correlator has been received before this work unit has been started by IWMMSTRT.

,AFTER_STRT=YES

indicates that the correlator has arrived within the scope of this work unit that means after issuing IWMMSTRT.

,END_FLOW=NO
,END_FLOW=YES

An optional parameter, which indicates the completion of a message flow. The default is END_FLOW=NO.

,END_FLOW=NO

indicates that a message flow has not completed.

,END_FLOW=YES

indicates that a message flow has completed. Specify END_FLOW=YES, if you know that the running work unit is the last one in a work unit flow. This indication cannot be cleared, if it has been set.

,EWLM_RCVD_CORR=*ewlm_rcvd_corr*
,EWLM_RCVD_CORR=NO_EWLM_RCVD_CORR

An optional input parameter, which contains a cross-platform Enterprise Workload Management (EWLM) correlator received from another application. Workflows often have multiple parent work units that must complete before a new work unit can be initiated. You can pass only one parent correlator to the IWMMSTRT macro and one additional parent correlator to the IWMMSTOP macro. You have to issue the IWMMUPD macro, if more than two parent correlators should be assigned to a work unit. This correlator is ignored, if it is an unknown EWLM correlator. The default is NO_EWLM_RCVD_CORR, which indicates that parameter EWLM_RCVD_CORR has not been specified.

To code: Specify the RS-type address, or address in register (2)-(12), of a character field.

,MESSAGES_SENT=*messages_sent*
,MESSAGES_SENT=NO_MESSAGES_SENT

An optional input parameter, which contains the number of messages sent to other applications. This value is added to the total messages_sent value of the work unit. The total messages_sent value should not exceed 32767. The default is NO_MESSAGES_SENT, which indicates that parameter MESSAGES_SEND has not been specified.

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*

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

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).

,STATUS=*status*
,STATUS=IWMEWLMARMSTATUSGOOD

An optional input parameter, which contains the completion status code of the work unit. 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 an explanation of the status codes, refer to the ARM 4.0 Standard at http://www.opengroup.org/management/arm. The default is IWMEWLMARMSTATUSGOOD, which indicates that the work unit completed successfully.

To code: Specify the RS-type address, or address in register (2)-(12), of a fullword field.

ABEND codes

None.

Return codes and reason codes

When the IWMMSTOP 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 121. Return and Reason Codes for the IWMMSTOP Macro

Return Code	Reason Code	Equate Symbol, Meaning, and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.
4	xxxx0402	**Equate Symbol**: IwmRsnCodeNoMonEnv **Meaning**: The input monitoring token indicates no monitoring environment was established. **Action**: Establish a monitoring environment by macro IWM4MCRE.
4	xxxx0441	**Equate Symbol**: IwmRsnCodeTooManyMsgCorrs **Meaning**: The correlator passed to EWLM_RCVD_CORR is ignored, since the maximum number of supported correlators has been reached. **Action**: None required.
4	xxxx0443	**Equate Symbol**: IwmRsnCodeTooManyMsgsSent **Meaning**: The value passed to MESSAGES_SENT is ignored, since the maximum number of messages sent is reached. **Action**: None required.
4	xxxx0444	**Equate Symbol**: IwmRsnCodeTooManyMsgsReceived **Meaning**: The EWLM_RCVD_CORR parameter has been specified too often. The correlated counter is not increased. **Action**: None required.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0820	**Equate Symbol**: IwmRsnCodeBadMonEnv **Meaning**: Input monitoring environment does not pass short form validity checking. **Action**: Check for possible storage overlay.

Table 121. Return and Reason Codes for the IWMMSTOP Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx0897	**Equate Symbol**: IwmRsnCodeTranStatusInvalid **Meaning**: An unsupported value has been passed to the STATUS parameter. **Action**: Specify a supported value.
8	xxxx08AC	**Equate Symbol**: IwmRsnCodeTranNotStarted **Meaning**: No work unit has been started by IWMMSTRT for the specified monitoring environment. **Action**: Start a work unit by IWMMSTRT macro, before issuing this macro.

IWMMSTRT — Indicate the start of a work unit

This service indicates that a work unit is beginning execution. The work unit runs under the specified monitoring environment, but is reported to EWLM completely independently from a potentially running transaction on the same monitoring environment that is defined by IWM4MINI and IWMRPT/IWMMNTFY calls. You can use the set of services IWMMSTRT, IWMMSTOP, IWMMUPD to provide data for "mini work units" running within a long-running transaction. In addition, a work unit started by IWMMSTRT can participate in asynchronous and synchronous work unit flows.

Environment

The requirements for the caller are:

Minimum authorization:	Supervisor state or program key mask (PKM) allowing keys 0-7.
Dispatchable unit mode:	Task or SRB
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 restrictions.
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. You must include the IWMYCON and CVT mapping macros in the calling program.
3. The macro IWMPB must be in the library concatenation, since it is included by IWMYCON.
4. Note that the high-order halfword of bits 0-31 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 before the completion of WLM address space initialization.
2. The caller must have issued the IWM4MCRE macro successfully and the created delay monitoring environment must be enabled for EWLM support. This means the delay monitoring environment must be created by one of the following ways:
   - IWM4CON EWLM=YES ... and IWM4MCRE SUBSYSP=CONNECT ... OR
   - IWM4MCRE SUBSYSP=VALUE EWLM=YES ...
3. The caller is responsible for error recovery.

4. The current PSW key must be 0 or match the key specified on IWM4MCRE provided the latter is a system key (0-7).
5. The caller must serialize to prevent any delay monitoring services from being invoked concurrently for the environment represented by the monitoring token.

Input register information

Before issuing the IWMMSTRT 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. The reason code is stored in bits 0-31.
1	Used as work registers by the system.
2 - 13	Unchanged
14	Used as a work register by the system.
15	Return code stored in bits 0-31

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.

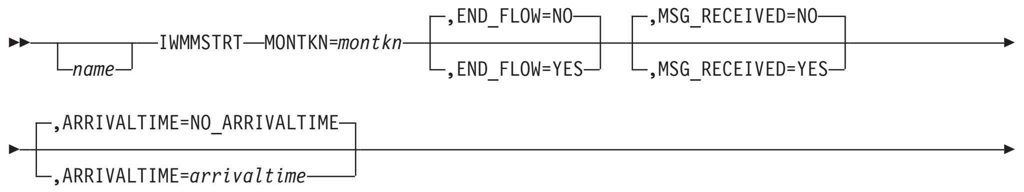
Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

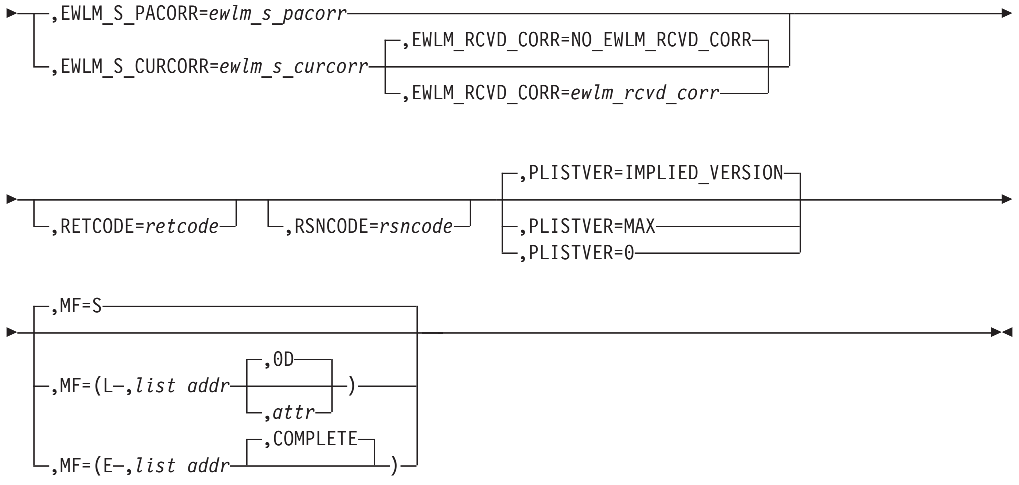
Performance implications

None.

Syntax

The syntax of the IWMMSTRT macro is as follows:




,EWLM_S_PACORR=ewlm_s_pacorr
,EWLM_S_CURCORR=ewlm_s_curcorr
,EWLM_RCVD_CORR=NO_EWLM_RCVD_CORR
,EWLM_RCVD_CORR=ewlm_rcvd_corr
,RETCODE=retcode
,RSNCODE=rsncode
,PLISTVER=IMPLIED_VERSION
,PLISTVER=MAX
,PLISTVER=0
,MF=S
,MF=(L–,list addr
,0D
,attr
)
,MF=(E–,list addr
,COMPLETE
)

Parameters

The parameters are explained as follows:

name

An optional symbol, starting in column 1, that is the name on the IWMMSTRT 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 a timestamp in STCK format. This timestamp is subtracted from the current timestamp and assigned as queued time to the work unit. For example, you may use this parameter, if the work unit is started by receipt of a message from a queue and you know the put time (the timestamp when the message has been put onto the queue). The default is NO_ARRIVALTIME which indicates that parameter ARRIVALTIME has not been specified.

To code: Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

,END_FLOW=NO
,END_FLOW=YES

An optional parameter, which indicates the completion of a message flow. The default is END_FLOW=NO.

,END_FLOW=NO

indicates that a message flow has not completed.

,END_FLOW=YES

indicates that a message flow has completed. Specify END_FLOW=YES, if you know that this work unit is the last one in a work unit flow. This indication cannot be cleared, if it has been set.

,EWLM_RCVD_CORR=*ewlm_rcvd_corr*
,EWLM_RCVD_CORR=NO_EWLM_RCVD_CORR

When EWLM_S_CURCORR=*ewlm_s_curcorr* is specified, an optional input parameter, which contains a cross platform Enterprise Workload Management (EWLM) correlator received from another application. Normally this is a received correlator which has the independent flag and the asynchronous flag set. It should not be passed to the EWLM_S_PACORR or the EWLM_S_CURCORR parameter. If you pass this correlator to one of them then the started work unit is not reclassified and runs under the classification of this correlator. When you receive a correlator with the independent flag set, you should:

1. Reclassify the work unit by issuing IWM4CLSY EWLM_CORR=r_corr EWLM_CHCORR=c_corr. r_corr is the received correlator and c_corr is the correlator created by IWM4CLSY.
2. Start the work unit by issuing IWMMSTRT EWLM_S_CURCORR=c_corr EWLM_RCVD_CORR=r_corr .

The default is NO_EWLM_RCVD_CORR which indicates that parameter EWLM_RCVD_CORR has not been specified.

To code: Specify the RS-type address, or address in register (2)-(12), of a character field.

,EWLM_S_CURCORR=*ewlm_s_curcorr*
A required input parameter which contains a cross-platform Enterprise Workload Management (EWLM) correlator for the current application. The correlator passed to this parameter is used as the current correlator of this work unit. It has usually been created by means of a previous IWM4CLSY call with the EWLM_CHCORR parameter.

To code: Specify the RS-type address, or address in register (2)-(12), of a character field.

,EWLM_S_PACORR=*ewlm_s_pacorr*
A required input parameter which contains a cross-platform Enterprise Workload Management (EWLM) parent correlator received from another application.

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***,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

To code: Specify the RS-type address, or address in register (2)-(12), of a 32-bit field.

,MSG_RECEIVED=NO
,MSG_RECEIVED=YES
An optional parameter, which indicates whether this work unit has been started as a result of a receipt of a message. The default is MSG_RECEIVED=NO.

,MSG_RECEIVED=NO

indicates that this work unit has not been started by receipt of a message.

,MSG_RECEIVED=YES

indicates that this work unit has been started as a result of a receipt of a message.

,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 codes and reason codes

When the IWMMSTRT 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 122. Return and Reason Codes for the IWMMSTRT Macro

Return Code	Reason Code	Equate Symbol, Meaning, and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.
4	xxxx0402	**Equate Symbol**: IwmRsnCodeNoMonEnv **Meaning**: The input monitoring token indicates no monitoring environment was established. **Action**: Establish a monitoring environment by macro IWM4MCRE.
4	xxxx0442	**Equate Symbol**: IwmRsnCodeCorrelatorUnknown **Meaning**: A unknown correlator has been passed to the EWLM_RCVD_CORR parameter. It is ignored. **Action**: None required.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0820	**Equate Symbol**: IwmRsnCodeBadMonEnv **Meaning**: Input monitoring environment does not pass short form validity checking. **Action**: Check for possible storage overlay.

Table 122. Return and Reason Codes for the IWMMSTRT Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx0894	**Equate Symbol**: IwmRsnCodeInvalidEwlmCorr **Meaning**: An unknown EWLM correlator has been passed to the EWLM_S_PACORR or EWLM_S_CURCORR parameter. **Action**: Specify a supported correlator. You can create a supported correlator by macro IWM4CLSY.
8	xxxx0895	**Equate Symbol**: IwmRsnCodeEWLMServNotEnabled **Meaning**: The EWLM service was not enabled for a delay monitoring environment. **Action**: Create a monitoring environment with EWLM=YES (either on macro IWM4CON or macro IWM4MCRE).
8	xxxx08AD	**Equate Symbol**: IwmRsnCodeAlreadyActive **Meaning**: A work unit started by IWMMSTRT is already active. **Action**: Stop the active work unit by macro IWMMSTOP before creating a new one.
8	xxxx08AF	**Equate Symbol**: IwmRsnCodeArrTimeGTStartTime **Meaning**: The arrivaltime passed is greater than the current timestamp. **Action**: Check the format of the passed arrivaltime.

Example

To indicate that the current monitoring environment continues only once elsewhere in the sysplex, specify:

```
IWMMSTRT FUNCTION=CONTINUE,WHERE=SYSPLEX,MONTKN=(R7),
         RETCODE=RCODE,RSNCODE=RSN
```

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:

Minimum authorization:	Problem state, or supervisor state. PSW key must either be 0, or match the value specified on IWM4MCRE.
Dispatchable unit mode:	Task or SRB
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:	Suspend locks are allowed, as are FRRs
Control parameters:	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

```
                                     ┌CONTINUE─┤ parameters-1 ├──────────────┐
►►─┬──────┬─IWMMSWCH──FUNCTION=──────┴RETURN───┬─────────────────────────────┬┴──────────────►
   └name──┘                                    └,RUNTIME_VER=──┬SHORT_FORM─┬┘
                                                               └MINIMAL────┘

►─,MONTKN=montkn─┬─────────────────────┬─┬───────────────────┬─┬───────────────────┬─────────►
                 └,COMPCODE=──┬YES─┬───┘ └,RETCODE=retcode───┘ └,RSNCODE=rsncode───┘
                              └NO──┘

  ┌,MF=S─────────────────────────────────┐
►─┼──────────────────────────────────────┼────────────────────────────────────────────────►◄
  │                   ┌,0D────────┐      │
  ├,MF=(L,─MFCTRL─────┼───────────┼─)────┤
  │                   └,mfattr────┘      │
  │                   ┌,COMPLETE──┐      │
  └,MF=(E,─MFCTRL─────┼───────────┼─)────┘
                      └,complete──┘
```

parameters-1

```
►►─┬──────────────────────────────────┬─,WHERE=─┬─LOCALMVS─┬──────────────────────────►◄
   └─,RUNTIME_VER=─┬─SHORT_FORM─┬─┘         ├─SYSPLEX──┤
                   └─MINIMAL────┘           └─NETWORK──┘
```

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.

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 codes 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		**Meaning**: Successful completion.
04	0402	**Meaning**: Warning. Input monitoring token indicates no monitoring environment was established.
04	0407	**Meaning**: Warning. Switch return was from a monitoring environment with an outstanding continuation.
08	081C	**Meaning**: Program error. Outstanding continuation exists.
08	0820	**Meaning**: 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
```

IWMMUPD — Update data for a work unit

The IWMMUPD service allows to update data about a work unit which has been started by IWMMSTRT. A work unit started by IWM4MINI is not affected by this service.

Environment

The requirements for the caller are:

Minimum authorization:	Supervisor state or program key mask (PKM) allowing keys 0-7.
Dispatchable unit mode:	Task or SRB
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 restrictions.
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 bits 0-31 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 before the completion of WLM address space initialization
2. The caller must have issued the IWMMSTRT macro successfully.
3. The caller is responsible for error recovery.
4. The current PSW key must be 0 or match the key specified on IWM4MCRE provided the latter is a system key (0-7).
5. The caller must serialize to prevent any delay monitoring services from being invoked concurrently for the environment represented by the monitoring token.

Input register information

Before issuing the IWMMUPD 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. The reason code is stored in bits 0-31.

1 Used as a work register by the system.

2 - 13 Unchanged.

14 Used as a work register by the system.

15 Return code is stored in bits 0-31.

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 work registers by the system.

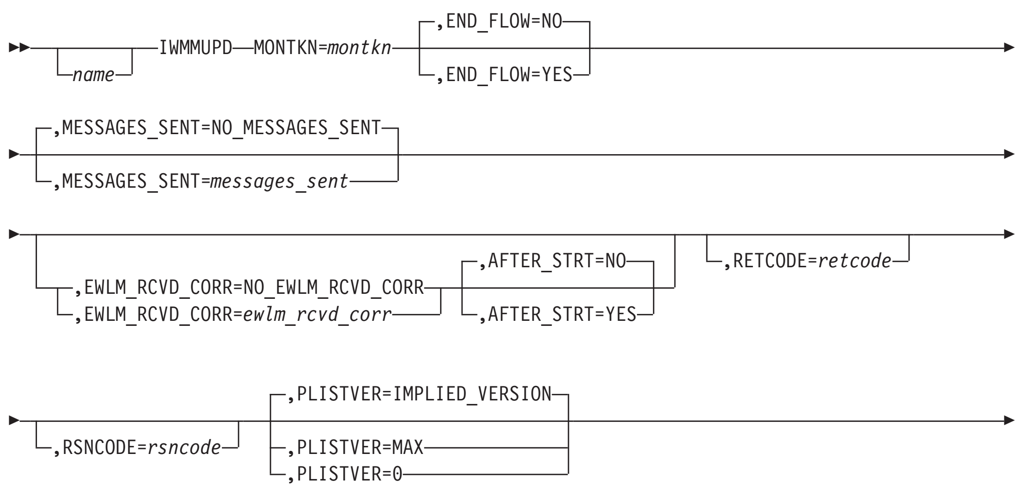
Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

Performance implications

None.

Syntax

The syntax of the IWMMUPD macro is as follows:


name
IWMMUPD
MONTKN=*montkn*
,END_FLOW=NO
,END_FLOW=YES
,MESSAGES_SENT=NO_MESSAGES_SENT
,MESSAGES_SENT=*messages_sent*
,EWLM_RCVD_CORR=NO_EWLM_RCVD_CORR
,EWLM_RCVD_CORR=*ewlm_rcvd_corr*
,AFTER_STRT=NO
,AFTER_STRT=YES
,RETCODE=*retcode*
,RSNCODE=*rsncode*
,PLISTVER=IMPLIED_VERSION
,PLISTVER=MAX
,PLISTVER=0

```
   ┌─,MF=S──────────────────────────────┐
►──┼────────────────────────────────────┼────────────────────────────────────────────►◄
   │                     ┌─,0D──┐       │
   ├─,MF=(L─,list addr───┼──────┼─)─────┤
   │                     └─,attr┘       │
   │                     ┌─,COMPLETE─┐  │
   └─,MF=(E─,list addr───┴───────────┴─)┘
```

Parameters

The parameters are explained as follows:

name

An optional symbol, starting in column 1, that is the name on the IWMMUPD macro invocation. The name must conform to the rules for an ordinary assembler language symbol.

,AFTER_STRT=NO
,AFTER_STRT=YES

When EWLM_RCVD_CORR=*ewlm_rcvd_corr* is specified, an optional parameter, which indicates the moment the correlator has been received. The default is AFTER_STRT=NO.

,AFTER_STRT=NO

indicates that the correlator has been received before this work unit has been started by IWMMSTRT.

,AFTER_STRT=YES

indicates that the correlator has arrived within the scope of this work unit that means after issuing IWMMSTRT.

,END_FLOW=NO
,END_FLOW=YES

An optional parameter, which indicates the completion of a message flow. The default is END_FLOW=NO.

,END_FLOW=NO

indicates that a message flow has not completed.

,END_FLOW=YES

indicates that a message flow has completed. Specify END_FLOW=YES, if you know that the running work unit is the last one in a work unit flow. This indication cannot be cleared, if it has been set.

,EWLM_RCVD_CORR=*ewlm_rcvd_corr*
,EWLM_RCVD_CORR=NO_EWLM_RCVD_CORR

An optional input parameter, which contains a cross-platform Enterprise Workload Management (EWLM) correlator received from another application. Workflows often have multiple parent work units that must complete before a new work unit can be initiated. You can pass only one parent correlator to the IWMMSTRT macro and one additional parent correlator to the IWMMUPD macro. You have to issue the IWMMUPD macro, if more than two parent correlators should be assigned to a work unit. This correlator is ignored, if it is an unknown EWLM correlator. The default is NO_EWLM_RCVD_CORR, which indicates that parameter EWLM_RCVD_CORR has not been specified.

To code: Specify the RS-type address, or address in register (2)-(12), of a character field.

,MESSAGES_SENT=*messages_sent*
,MESSAGES_SENT=NO_MESSAGES_SENT

An optional input parameter, which contains the number of messages sent to other applications. This value is added to the total messages_sent value of the work unit. The total messages_sent value should not exceed 32767. The default is NO_MESSAGES_SENT, which indicates that parameter MESSAGES_SEND has not been specified.

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*

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

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 codes and reason codes

When the IWMMUPD 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 123. Return and Reason Codes for the IWMMUPD Macro

Return Code	Reason Code	Equate Symbol, Meaning, and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.

Table 123. Return and Reason Codes for the IWMMUPD Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.
4	xxxx0402	**Equate Symbol**: IwmRsnCodeNoMonEnv **Meaning**: The input monitoring token indicates no monitoring environment was established. **Action**: Establish a monitoring environment by macro IWM4MCRE.
4	xxxx0441	**Equate Symbol**: IwmRsnCodeTooManyMsgCorrs **Meaning**: The correlator passed to EWLM_RCVD_CORR is ignored, since the maximum number of supported correlators has been reached. **Action**: None required.
4	xxxx0443	**Equate Symbol**: IwmRsnCodeTooManyMsgsSent **Meaning**: The value passed to MESSAGES_SENT is ignored, since the maximum number of messages sent is reached. **Action**: None required.
4	xxxx0444	**Equate Symbol**: IwmRsnCodeTooManyMsgsReceived **Meaning**: The EWLM_RCVD_CORR parameter has been specified too often. The correlated counter is not increased. **Action**: None required.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0820	**Equate Symbol**: IwmRsnCodeBadMonEnv **Meaning**: Input monitoring environment does not pass short form validity checking. **Action**: Check for possible storage overlay.
8	xxxx08AC	**Equate Symbol**: IwmRsnCodeTranNotStarted **Meaning**: No work unit has been started by IWMMSTRT for the specified monitoring environment. **Action**: Start a work unit by IWMMSTRT macro, before issuing this macro.

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.
Dispatchable unit mode:	Task or SRB
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 may be held.
Control parameters:	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. 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(CONTINUE|RETURN) 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

The syntax of the IWMMXFER macro is as follows:

```
►►──┬──────┬──IWMMXFER──────────────────────────────────────────────────────────►
    └─name─┘

              ┌─,RUNTIME_VER=SHORT_FORM─┐
►──┬─FUNCTION=CONTINUE─┼─────────────────────────┼──────────────────────────────────┬──►
   │                   └─,RUNTIME_VER=MINIMAL────┘                                  │
   │                 ┌─,RUNTIME_VER=SHORT_FORM─┐  ┌─,WORKREQ_STA=IWMEWLMARMSTATUSNONE─┐│
   └─FUNCTION=RETURN─┼─────────────────────────┼──┼───────────────────────────────────┼┘
                     └─,RUNTIME_VER=MINIMAL────┘  └─,WORKREQ_STA=workreq_sta──────────┘

►─,MONTKN=montkn──┬─,PARENTKEYP=VALUE─,PARENTKEY=parentkey─┬──┬─,PARENTENV=NOSWITCH──┬──►
                  ├─,PARENTKEYP=PSWKEY─────────────────────┤  └─,PARENTENV=SECONDARY─┘
                  └─,PARENTKEYP=UNKNOWN────────────────────┘

   ┌─,COMPCODE=YES─┐                                        ┌─,PLISTVER=IMPLIED_VERSION─┐
►──┼───────────────┼──┬─────────────────┬──┬─────────────────┬──┼───────────────────────────┼──►
   └─,COMPCODE=NO──┘  └─,RETCODE=retcode─┘  └─,RSNCODE=rsncode─┘  ├─,PLISTVER=MAX─────────────┤
                                                                ├─,PLISTVER=0───────────────┤
                                                                └─,PLISTVER=1───────────────┘

                        ┌─,COMPLETE─┐
►─,MF=(M─,list addr─┼───────────┼─)──────────────────────────────────────────────►◄
                        └─,NOCHECK──┘
```

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 "normal" 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 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

,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 usable 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 usable 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 the following:

- IwmEwlmArmStatusGood(0)
- IwmEwlmArmStatusAborted(1)

- IwmEwlmArmStatusFailed(2)
- IwmEwlmArmStatusUnknown(3)

These codes correspond to status codes in the OpenGroup ARM 4.0 Standard. For further information about the meaning of the status codes, refer to the ARM 4.0 Standard at http://www.opengroup.org/management/arm. The default is IWMEWLMARMSTATUSNONE. This indicates that internal information in the monitoring environment are examined to determine the status of the work request. If no abnormal event was recorded for the monitoring environment with the IWMMABNL service, the completion status IwmEwlmArmStatusGood is reported to EWLM. If an abnormal event was reported with IWMMABNL, the completion status IwmEwlmArmStatusFailed is reported to EWLM.

To code: Specify the RS-type address, or address in register (2)-(12), of a fullword field.

ABEND codes

None.

Return codes and reason codes

When the IWMMXFER 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 124. Return and Reason Codes for the IWMMXFER Macro

Return Code	Reason Code	Equate Symbol, Meaning, and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.
4	xxxx0402	**Equate Symbol**: IwmRsnCodeNoMonEnv **Meaning**: Input monitoring token indicates no monitoring environment was established. **Action**: None required.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx081F	**Equate Symbol**: IwmRsnCodeNoRelate **Meaning**: NO Parent environment exists since Relate Function(Continue) has not been performed or has not been performed subsequent to a Relate Function(Delete). **Action**: Check for possible storage overlay and whether Relate Function(Continue) has been used properly.

Table 124. Return and Reason Codes for the IWMMXFER Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx0820	**Equate Symbol**: IwmRsnCodeBadMonEnv **Meaning**: Monitoring environment does not pass verification.
8	xxxx0822	IwmRsnCodeBadParEnv: Parent monitoring environment does not pass verification. **Action**: Check for possible storage overlay.
8	xxxx0895	**Equate Symbol**: IwmRsnCodeEWLMServNotEnabled **Meaning**: Service is not enabled because monitoring environment can not be associated with EWLM work requests. **Action**: 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.
8	xxxx0897	**Equate Symbol**: IwmRsnCodeTranStatusInvalid **Meaning**: Passed work request completion status is not valid. **Action**: Check the EWLM ARM interface specification for valid completion status values.

IWMQDEL — Delete 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 equivalent service, IWM4QDE, which also supports 64-bit addressing. For more information, see “IWM4QDE — Delete a request from the queue for an execution address space” on page 683.

Environment

The requirements for the caller are:

Minimum authorization:	Supervisor state or program key mask (PKM) allowing keys 0-7.
Dispatchable unit mode:	Task or SRB
Cross memory mode:	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).
AMODE:	31-bit
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks may be held.
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, IWMRSNCODE_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
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

The syntax of the IWMQDEL macro is as follows:

```
►►──┬──────┬──IWMQDEL──CONNTKN=conntkn──,WLMWUTKN=wlmwutkn──┬────────────────────┬──────►
    └─name─┘                                                └─,RETCODE=retcode───┘

                           ┌─,PLISTVER=IMPLIED_VERSION─┐
►──┬────────────────────┬──┼───────────────────────────┼──────────────────────────────►
   └─,RSNCODE=rsncode───┘  ├─,PLISTVER=MAX─────────────┤
                           └─,PLISTVER=0───────────────┘

   ┌─,MF=S──────────────────────────────────┐
►──┼────────────────────────────────────────┼────────────────────────────────────────►◄
   │                        ┌─,0D────┐      │
   ├─,MF=(L─,list addr──────┼────────┼─)────┤
   │                        └─,attr──┘      │
   │                        ┌─,COMPLETE─┐   │
   └─,MF=(E─,list addr──────┴───────────┴─)─┘
```

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).

,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 codes 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 125. Return and Reason Codes for the IWMQDEL Macro

Return Code	Reason Code	Equate Symbol, Meaning, and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.
4	xxxx0408	**Equate Symbol**: IwmRsnCodeWorkNotFound: **Meaning**: No work matching the input search criteria was found. **Action**: None required.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: Caller is locked. **Action**: Avoid requesting this function while locked.
8	xxxx080B	**Equate Symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list. **Action**: 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	xxxx0821	**Equate Symbol**: IwmRsnCodeBadConn **Meaning**: Input connect token does not pass validity checking. **Action**: Make sure to use the connect token returned by the IWMCONN service requesting Queue_Manager=Yes, or Server_Manager=Yes with Server_Type=Queue.
8	xxxx0823	**Equate Symbol**: IwmRsnCodeDatoff **Meaning**: Caller invoked service while DATOFF **Action**: Avoid requesting this function in this environment.
8	xxxx0824	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: Caller invoked service but was in 24 bit addressing mode. **Action**: Request this function only when you are in 31 bit addressing mode.
8	xxxx0825	**Equate Symbol**: IwmRsnCodeAscModeNotPrimary **Meaning**: Caller invoked service but was not DAT on Primary ASC mod **Action**: Avoid requesting this function in this environment.

Table 125. Return and Reason Codes for the IWMQDEL Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx0827	**Equate Symbol**: IwmRsnCodeRsvdNot0 **Meaning**: Reserved field in parameter list was non-zero. **Action**: Check for use of keywords that are not supported by the MVS release on which the program is running.
8	xxxx0828	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: Version number in parameter list is not valid. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx083F	**Equate Symbol**: IwmRsnCodePrimaryNotOwnConn **Meaning**: Primary address space does not own the passed connect token. **Action**: Avoid requesting this function while primary address space does not own the connect token.
8	xxxx0840	**Equate Symbol**: IwmRsnCodeServiceNotEnabled **Meaning**: Caller's space connection is not enabled for this service **Action**: 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	xxxx0848	**Equate Symbol**: IwmRsnCodeBadWorkUnitToken **Meaning**: The work unit token is not valid. **Action**: Check the specification of the WLMWUTKN parameter.
8	xxxx089E	**Equate Symbol**: IwmRsnCodeServiceAModeMismatch **Meaning**: The caller is in 64-bit address mode and tried to invoke a service macro that is only enabled for a 31-bit environment. **Action**: Use the 64-bit enabled service macro (IWM4QDE) or change the address mode of the caller to 31-bit.
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error. **Action**: Contact your system programmer.

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 — Insert 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 equivalent service, IWM4QIN, which also supports 64-bit addressing. For more information, see "IWM4QIN — Insert a request onto the queue for an execution address space" on page 696.

Environment

The requirements for the caller are:

Minimum authorization:	Supervisor state or program key mask (PKM) allowing keys 0-7.
Dispatchable unit mode:	Task or SRB
Cross memory mode:	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).
AMODE:	31-bit
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks may be held.
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, 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

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:

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

The syntax of the IWMQINS macro is as follows:

```
►►──┬──────┬──IWMQINS──CONNTKN=conntkn──,ETOKEN=etoken──,USERDATA=userdata──────────►
    └─name─┘

                   ┌─,DYNAMIC=NO──┐ ┌─,DEPENDENT=NO──┐
►─,APPLENV=applenv─┼──────────────┼─┼────────────────┼──────────────────────────────►
                   └─,DYNAMIC=YES─┘ └─,DEPENDENT=YES─┘
```

```
   ┌─,SECUSER=NO──────────────────┐
►──┼──────────────────────────────┼──┬────────────────────┬──────────────────────────────────►
   └─,SECUSER=YES─,USERID=userid──┘  └─,WLMWUTKN=wlmwutkn─┘

   ┌─,SERVER_TOKEN=0─────────────┐  ┌─,REGION_TOKEN=0─────────────┐
►──┼─────────────────────────────┼──┼─────────────────────────────┼──┬──────────────────┬────►
   └─,SERVER_TOKEN=server_token──┘  └─,REGION_TOKEN=region_token──┘  └─,RETCODE=retcode─┘

                               ┌─,PLISTVER=IMPLIED_VERSION─┐
►──┬────────────────────┬──────┼───────────────────────────┼──────────────────────────────────►
   └─,RSNCODE=rsncode───┘      ├─,PLISTVER=MAX─────────────┤
                               ├─,PLISTVER=0───────────────┤
                               ├─,PLISTVER=1───────────────┤
                               ├─,PLISTVER=2───────────────┤
                               └─,PLISTVER=3───────────────┘

   ┌─,MF=S─────────────────────────────────────┐
►──┼───────────────────────────────────────────┼──────────────────────────────────────────────►◄
   │                        ┌─,0D────┐          │
   ├─,MF=(L─,list addr──────┼────────┼─)────────┤
   │                        └─,attr──┘          │
   │                        ┌─,COMPLETE─┐       │
   └─,MF=(E─,list addr──────┴───────────┴─)─────┘
```

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 IWMCONN 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 defintion. 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 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

,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**, which supports all parameters except those specifically referenced in higher versions.
- **1**, which supports the following parameter and those from version 0:

  SERVER_TOKEN
- **2**, which supports the following parameter and those from version 0 and 1:

  REGION_TOKEN
- **3**, which supports the following parameter 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 IWM4CON and IWMSSEL 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.

,SECUSER=NO
No security environment to be established.

,SECUSER=YES
Use the specified user ID to establish a security environment.

,SERVER_TOKEN=*server_token*
,SERVER_TOKEN=0
An optional input parameter, which contains a server token returned by the IWMSSEL 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 IWMSSEL.
- 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 IWMSSEL 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 user ID.

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 codes 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 126. Return and Reason Codes for the IWMQINS Macro

Return Code	Reason Code	Equate Symbol, Meaning, and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.
4	xxxx042E	**Equate Symbol**: IwmRsnCodeServerNotFound **Meaning**: The server token does not identify an existing server tas The server task may have terminated since the token was obtained. **Action**: If the server task has not terminated, check that the correct token is specified.
4	xxxx043A	**Equate Symbol**: IwmRsnCodeRegionNotFound **Meaning**: The region token does not identify a valid server region. **Action**: Please specify the correct region token.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.

Table 126. Return and Reason Codes for the IWMQINS Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: Caller is locked. **Action**: Avoid requesting this function while locked.
8	xxxx080B	**Equate Symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list. **Action**: 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	xxxx0821	**Equate Symbol**: IwmRsnCodeBadConn **Meaning**: Input connect token does not pass validity checking. **Action**: Make sure to use the connect token returned by the IWMCONN service requesting Queue_Manager=Yes, or Server_Manager=Yes with Server_Type=Queue.
8	xxxx0823	**Equate Symbol**: IwmRsnCodeDatoff **Meaning**: Caller invoked service while DATOFF **Action**: Avoid requesting this function in this environment.
8	xxxx0824	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: Caller invoked service but was in 24 bit addressing mode. **Action**: Request this function only when you are in 31 bit addressing mode.
8	xxxx0825	**Equate Symbol**: IwmRsnCodeAscModeNotPrimary **Meaning**: Caller invoked service but was not in primary ASC mode. **Action**: Avoid requesting this function in this environment.
8	xxxx0827	**Equate Symbol**: IwmRsnCodeRsvdNot0 **Meaning**: Reserved field in parameter list was non-zero. **Action**: Check for use of keywords that are not supported by the MVS release on which the program is running.
8	xxxx0828	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: Version number in parameter list is not valid. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx083A	**Equate Symbol**: IwmRsnCodeBadEnclave **Meaning**: Enclave token in parameter list is not valid. **Action**: Check the specification of the ETOKEN parameter.

Table 126. Return and Reason Codes for the IWMQINS Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx083F	**Equate Symbol**: IwmRsnCodePrimaryNotOwnConn **Meaning**: Primary address space does not own the passed connect token. **Action**: 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	**Equate Symbol**: IwmRsnCodeServiceNotEnabled **Meaning**: Caller's space connection is not enabled for this service **Action**: 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	**Equate Symbol**: IwmRsnCodeNoWLMConnect **Meaning**: Caller's space disconnected from WLM during processing of the insert request. **Action**: None.
8	xxxx0845	**Equate Symbol**: IwmRsnCodeWrongEnclave **Meaning**: 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. **Action**: Check that the correct enclave token was specified. Check that the server task is invoking the IWMSSEL and IWMSSEM macros in the correct sequence.
8	xxxx089E	**Equate Symbol**: IwmRsnCodeServiceAModeMismatch **Meaning**: The caller is in 64-bit address mode and tried to invoke a service macro that is only enabled for a 31-bit environment. **Action**: Use the 64-bit enabled service macro (IWM4QIN) or change the address mode of the caller to 31-bit.
C	—	**Equate Symbol**: IwmRetCodeEnvError **Meaning**: Environmental error.
C	xxxx0C01	**Equate Symbol**: IwmRsnCodeNoStg **Meaning**: Storage is not available for the request. **Action**: Contact your system programmer. There is a common storage shortage.

Table 126. Return and Reason Codes for the IWMQINS Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
C	xxxx0C16	**Equate Symbol**: IwmRsnCodeServerUnavail **Meaning**: A server cannot be started to process the work request. This is probably caused by one of the following: 1. An error in the JCL procedure used to start the server address space. 2. Repeated, unexpected terminations of the server address space. In either of these cases, workload management stops the application environment. A DISPLAY WLM command shows this state as INTERNALLY STOPPED. **Action**: Look at the system log to determine what caused the error: 1. If it is a JCL error, correct the error in the procedure. 2. If it is repeated terminations of the server address space, correct the application error causing the termination. 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. **Note:** 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.
C	xxxx0C1A	**Equate Symbol**: IwmRsnCodeApplNotDefined **Meaning**: The application environment name is not defined in the active WLM policy. **Action**: 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.
C	xxxx0C1B	**Equate Symbol**: IwmRsnCodeApplNotSST **Meaning**: The application environment name is defined for use by a different subsystem type in the active WLM policy. **Action**: 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.
C	xxxx0C1D	**Equate Symbol**: IwmRsnCodeQMgrNotActive **Meaning**: The required Queue Manager is not active. **Action**: 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.
C	xxxx0C22	**Equate Symbol**: IwmRsnCodeApplEnvQuiesced **Meaning**: For server applications connecting to WLM with subsystem type IWEB only: The application environment has been quiesced. The work reqeust is not inserted to the WLM work queue. **Action**: Resume the application environment.

Table 126. Return and Reason Codes for the IWMQINS Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
C	xxxx0C40	**Equate Symbol**: IwmRsnCodeNoSafCheckPossible **Meaning**: MLS is active but a security check could not be performed probably for one of the following reasons: • • 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,... • A resource or class name is not defined to RACF or the class has not been RAClisted. • The class was RAClisted, but the data space cannot be accessed due to an ALESERV failure. • The class was RAClisted, but the data space has been deleted. • 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. **Action**: Contact your RACF Security Administrator. Check if RACF is properly installed, configured and tuned. Correct the eventual problems.
C	xxxx0C41	**Equate Symbol**: IwmRsnCodeSafCheckFailed **Meaning**: MLS is active. Queue Manager and Server Manager are not authorized to communicate. **Action**: Normally none. If QM and SM really must communicate, conta your RACF Security Administrator. Set the appropriate Security Labels.
C	xxxx0C42	**Equate Symbol**: IwmRsnCodeAletError **Meaning**: Error while accessing access list with ALESERV probably because of one of the following 1. 2. The current access list cannot be expanded. There are no free access list entries and the maximum size has been reached. 3. ALESERV could not obtain storage for an expanded access list. **Action**: 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
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error. **Action**: Contact your system programmer.

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
```

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:

Minimum authorization:	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.
Dispatchable unit mode:	Task or SRB
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:	LOCAL lock held
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, 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. The 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. 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 IWM4MRLT.
   - 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:

Register	Contents
13	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:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system.

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

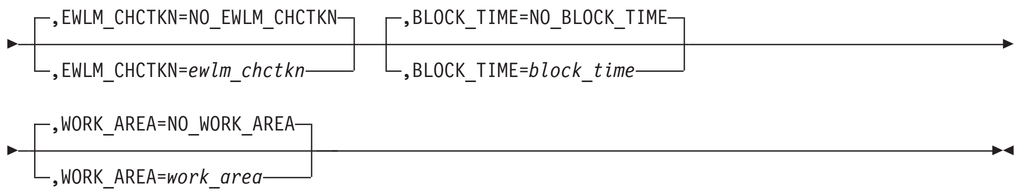
The syntax of the IWMRPT macro is as follows:

main diagram

```
                         ┌─TRAXFRPT=NO─┐ ┌─,PSWKEYP=CURRENT──────────────────┐
►►──┬──────┬──IWMRPT──┬──┴─────────────┴─┼───────────────────────────────────┼─┬──────────────────►
    └─name─┘          │                  └─,PSWKEYP=VALUE─,PSWKEY=pswkey─────┘ │
                      └─TRAXFRPT=YES─,SYSEVPL=sysevpl──────────────────────────┘

                                                                ┌─,ENDTIME=CURRENT─┐
►──┬─,MONTKNI=YES─,MONTKN=montkn──────┬─,CONNTKN=conntkn────────┼──────────────────┼──────────────►
   └─,MONTKNI=NO─┤ parameters-1 ├─────┘                         └─,ENDTIME=endtime─┘

   ┌─,STATUS=NORMAL─┐ ┌─,WORK_COMPCD=NO_WORK_COMPCD─┐
►──┼────────────────┴─┼─────────────────────────────┼──────────────────────────┬─────────────────►
   │                  └─,WORK_COMPCD=work_compcd────┘                          │
   │                    ┌─,WORK_COMPCD=NO_WORK_COMPCD─┐                        │
   ├─,STATUS=ABNORMAL───┼─────────────────────────────┼────────────────────────┤
   │                    └─,WORK_COMPCD=work_compcd────┘                        │
   ├─,STATUS=NORMAL_LE_VAL─,WORK_COMPCD=work_compcd─,OK_THRESHOLD=ok_threshold─┤
   └─,STATUS=NORMAL_GE_VAL─,WORK_COMPCD=work_compcd─,OK_THRESHOLD=ok_threshold─┘

   ┌─,WORKREQ_STA=IWMEWLMARMSTATUSNONE─┐
►──┼───────────────────────────────────┼──┬──────────────────┬──┬──────────────────┬─────────────►
   └─,WORKREQ_STA=workreq_sta──────────┘  └─,RETCODE=retcode─┘  └─,RSNCODE=rsncode─┘

   ┌─,PLISTVER=IMPLIED_VERSION─┐ ┌─,MF=S──────────────────────────────┐
►──┼───────────────────────────┼─┼────────────────────────────────────┼─────────────────────────►◄
   ├─,PLISTVER=MAX─────────────┤ │                      ┌─,0D────┐    │
   ├─,PLISTVER=0───────────────┤ ├─,MF=(L─,list addr──┼────────┼─)──┤
   └─,PLISTVER=1───────────────┘ │                      └─,attr──┘    │
                                 │                      ┌─,COMPLETE─┐ │
                                 └─,MF=(E─,list addr──┴───────────┴─)─┘
```

parameters-1

```
►►──,ARRIVALTIME=arrivaltime──┬─,EXSTARTTIMEP=NO──────────────────────────────┬─────────────────►
                              └─,EXSTARTTIMEP=YES─,EXSTARTTIME=exstarttime────┘

►─,SERVCLS=servcls──┬─────────────────────────────────────────────────────────────────────┬─────►
                    │                                  ┌─,EWLM_PACORR=NO_EWLM_PACORR─┐    │
                    └─┬─,EWLM_CHCORR=NO_EWLM_CHCORR─┬──┼─────────────────────────────┼────┘
                      └─,EWLM_CHCORR=ewlm_chcorr────┘  └─,EWLM_PACORR=ewlm_pacorr────┘
```



Parameters

The parameters are explained as follows:

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, which 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*,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 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 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=*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.

,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, which 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, which 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 the following:

- IwmEwlmArmStatusGood(0)
- IwmEwlmArmStatusAborted(1)
- IwmEwlmArmStatusFailed(2)
- IwmEwlmArmStatusUnknown(3)

These codes correspond to status codes in the OpenGroup ARM 4.0 Standard. For further information about the meaning of the status codes refer to the ARM 4.0 Standard at http://www.opengroup.org/management/arm. The default is IWMEWLMARMSTATUSNONE. This 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 codes 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 127. Return and Reason Codes for the IWMRPT Macro

Return Code	Reason Code	Equate Symbol, Meaning, and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.
4	xxxx0405	**Equate Symbol**: IwmRsnCodeGoalNoMonEnv **Meaning**: System is in goal mode but the input monitoring token indicates no monitoring environment was established, hence MVS did not receive the information. **Action**: None required.
4	xxxx0409	**Equate Symbol**: IwmRsnCodeNoConn **Meaning**: 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. **Action**: None required.

Table 127. Return and Reason Codes for the IWMRPT Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx080C	**Equate Symbol**: IwmRsnCodeMonEnvLacksData **Meaning**: Input monitoring environment does not contain the necessary information. **Action**: Ensure that the monitoring environment was established with the necessary information.
8	xxxx080E	**Equate Symbol**: IwmRsnCodeArrTimeGTEndTime **Meaning**: Input arrival time later than end time. **Action**: Check for possible storage overlay of the parameter list or variable.
8	xxxx0820	**Equate Symbol**: IwmRsnCodeBadMonEnv **Meaning**: Input monitoring environment does not pass short form validity checking. **Action**: Check for possible storage overlay.
8	xxxx0821	**Equate Symbol**: IwmRsnCodeBadConn **Meaning**: Input connect token does not pass validity checking. **Action**: Check for possible storage overlay.
8	xxxx082D	**Equate Symbol**: IwmRsnCodeExStTimeGTEndTime **Meaning**: Execution start time greater than execution end time **Action**: Check for possible storage overlay of the parameter list or variable.
8	xxxx0840	**Equate Symbol**: IwmRsnCodeServiceNotEnabled **Meaning**: Caller's space connection is not enabled for this service. **Action**: Avoid requesting this function under the input connection. IWM4CON options must be specified previously to enable this service.
8	xxxx087E	**Equate Symbol**: IwmRsnCodeRoMonEnv **Meaning**: Input monitoring environment is report-only. **Action**: Avoid calling this function for report-only monitoring environments.
8	xxxx0894	**Equate Symbol**: IwmRsnCodeInvalidEWLMCorr **Meaning**: 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). **Action**: Check the specification of the correlator information.

Table 127. Return and Reason Codes for the IWMRPT Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx0895	**Equate Symbol**: IwmRsnCodeEWLMServNotEnabled **Meaning**: The service is not enabled because the caller invoked the IWMCONN service with EWLM=NO. **Action**: Specify the parameter EWLM_CHCORR, EWLM_PACORR, EWLM_CHCTKN, or WORKREQ_STA only when connected with EWLM=YES.
8	xxxx0897	**Equate Symbol**: IwmRsnCodeTranStatusInvalid **Meaning**: Passed work request completion status is not valid. **Action**: Check the EWLM ARM interface specification for valid completion status values.
C	—	**Equate Symbol**: IwmRetCodeEnvError **Meaning**: Environmental error.
C	xxxx0C02	**Equate Symbol**: IwmRsnCodeReportingSusp **Meaning**: SYSEVENT TRAXFRPT invoked, but reporting is temporarily suspended for one of the following reasons: • RMF workload activity reporting is not active • 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. **Action**: Invoke the function when the conditions are alleviated.
C	xxxx0C03	**Equate Symbol**: IwmRsnCodeSyseventNoWorkElt **Meaning**: SYSEVENT TRAXFRPT invoked, but no work element was available to save the input information. **Action**: Invoke the function when the conditions are alleviated. This condition may be due to a common storage shortage condition.
C	xxxx0C05	**Equate Symbol**: IwmRsnCodeRptNoWorkElt **Meaning**: Report routine invoked, but no work element was available to save the input information. **Action**: Invoke the function when the conditions are alleviated. This condition may be due to a common storage shortage condition.
C	xxxx0C06	**Equate Symbol**: IwmRsnCodeNoEndTime **Meaning**: No end time was supplied to the service and STCK gave a non-zero condition code. **Action**: No action required.

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 equivalent service, IWM4SLI, which also supports 64-bit addressing. For more information, see “IWM4SLI — Application environment limit service” on page 720.

Environment

The requirements for the caller are:

Minimum authorization:	Problem state. Any PSW key
Dispatchable unit mode:	Task
Cross memory mode:	PASN=HASN=SASN
AMODE:	31-bit
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	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.

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 IWMCONN Server_Manager=YES, Server_Type=Queue.
4. 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

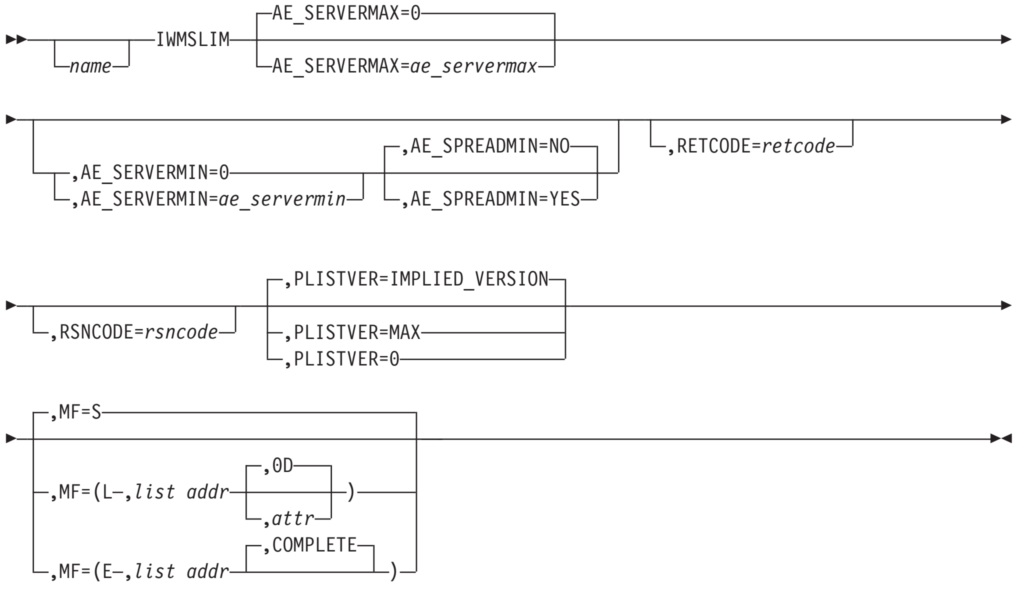
Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

Performance implications

None.

Syntax

The syntax of the IWMSLIM macro is as follows:



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_SPREADMIN=NO
,AE_SPREADMIN=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_SPREADMIN=NO.

,AE_SPREADMIN=NO
The server tasks specified in AE_SERVERMIN will be distributed to service classes as needed in order to meet goals.

,AE_SPREADMIN=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:

- **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 codes 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 128. Return and Reason Codes for the IWMSLIM Macro

Return Code	Reason Code	Equate Symbol, Meaning, and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0801	**Equate Symbol**: IwmRsnCodeSrbMode **Meaning**: Caller is in SRB mode. **Action**: Avoid requesting this function while in SRB mode.
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: Caller is locked. **Action**: Avoid requesting this function while locked.
8	xxxx080B	**Equate Symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list. **Action**: 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	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: Caller invoked service but was in 24 bit addressing mode. **Action**: Request this function only when you are in 31 bit addressing mode.
8	xxxx0825	**Equate Symbol**: IwmRsnCodeAscModeNotPrimary **Meaning**: Caller invoked service but was not in primary ASC mode. **Action**: Avoid requesting this function in this environment.
8	xxxx0828	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: Version number in parameter list is not valid. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx0840	**Equate Symbol**: IwmRsnCodeServiceNotEnabled **Meaning**: Caller's space connection is not enabled for this service **Action**: Make sure that SERVER_MANAGER=YES and SERVER_TYPE=QUEUE is specified on the IWMCONN request to enable this service.

Table 128. Return and Reason Codes for the IWMSLIM Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx0841	**Equate Symbol**: IwmRsnCodeXmemMode **Meaning**: Caller is in cross-memory mode. **Action**: Request this function only when you are not in cross-memory mode.
8	xxxx0842	**Equate Symbol**: IwmRsnCodeNoWLMConnect **Meaning**: Caller's space is not connected to WLM. **Action**: Invoke the IWMCONN macro before invoking this macro.
8	xxxx089E	**Equate Symbol**: IwmRsnCodeServiceAModeMismatch **Meaning**: The caller is in 64-bit address mode and tried to invoke a service macro that is only enabled for a 31-bit environment. **Action**: Use the 64-bit enabled service macro (IWM4SLI) or change the address mode of the caller to 31-bit.
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error. **Action**: Contact your system programmer.

Example

To specify application limits to WLM.

```
  IWMCONN 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
```

IWMSSEL — Select a request from a caller's work manager queue

The IWMSSEL 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 equivalent service, IWM4SSL, which also supports 64-bit addressing. For more information, see "IWM4SSL — Select a request from a caller's work manager queue" on page 739.

Environment

The requirements for the caller are:

Minimum authorization:	Problem state. Any PSW key
Dispatchable unit mode:	Task
Cross memory mode:	PASN=HASN=SASN
AMODE:	31-bit
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks may be held.
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, IWMRSNCODE_MASK_CONST defined in IWMYCON, may be used for this purpose.

Restrictions

None.

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.

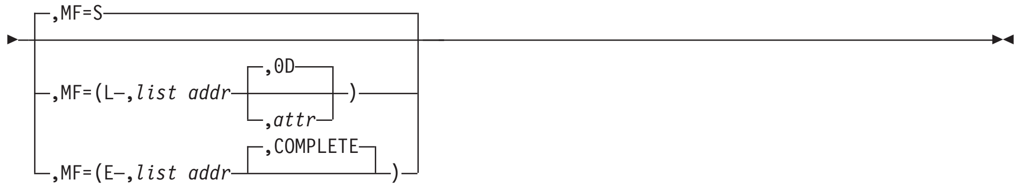
Performance implications

None.

Syntax

The syntax of the IWMSSEL macro is as follows:

```
►►─┬──────┬─IWMSSEL──USERDATA=userdata──,WLMEUTKN=wlmeutkn──────────────────────►
   └─name─┘

►─┬──────────────────────────────┬─┬──────────────────────────────┬─┬────────────────────┬─►
  └─,SERVER_TOKEN=server_token─┘ └─,REGION_TOKEN=region_token─┘ └─,RETCODE=retcode─┘

                         ┌─,PLISTVER=IMPLIED_VERSION─┐
►─┬────────────────────┬─┼───────────────────────────┼──────────────────────────►
  └─,RSNCODE=rsncode─┘   ├─,PLISTVER=MAX─────────────┤
                         ├─,PLISTVER=0───────────────┤
                         └─,PLISTVER=1───────────────┘
```



Parameters

The parameters are explained as follows:

name

An optional symbol, starting in column 1, that is the name on the IWMSSEL 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.
- **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 the following parameter 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 vitual 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 codes and reason codes

When the IWMSSEL 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 129. Return and Reason Codes for the IWMSSEL Macro

Return Code	Reason Code	Equate Symbol, Meaning, and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0801	**Equate Symbol**: IwmRsnCodeSrbMode **Meaning**: Caller is in SRB mode. **Action**: Avoid requesting this function while in SRB mode.
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: Caller is locked. **Action**: Avoid requesting this function while locked.

Table 129. Return and Reason Codes for the IWMSSEL Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx080B	**Equate Symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list. **Action**: 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	**Equate Symbol**: IwmRsnCodeDatoff **Meaning**: Caller invoked service while DATOFF **Action**: Avoid requesting this function in this environment.
8	xxxx0824	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: Caller invoked service but was in 24 bit addressing mode. **Action**: Request this function only when you are in 31 bit addressing mode.
8	xxxx0825	**Equate Symbol**: IwmRsnCodeAscModeNotPrimary **Meaning**: Caller invoked service but was not in primary ASC mode. **Action**: Avoid requesting this function in this environment.
8	xxxx0827	**Equate Symbol**: IwmRsnCodeRsvdNot0 **Meaning**: Reserved field in parameter list was non-zero. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx0828	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: Version number in parameter list is not valid. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx0840	**Equate Symbol**: IwmRsnCodeServiceNotEnabled **Meaning**: Caller's space connection is not enabled for this service **Action**: Make sure that SERVER_MANAGER=YES is specified on the IWMCONN request to enable this service.
8	xxxx0841	**Equate Symbol**: IwmRsnCodeXmemMode **Meaning**: Caller is in cross-memory mode. **Action**: Request this function only when you are not in cross-memory mode.
8	xxxx0842	**Equate Symbol**: IwmRsnCodeNoWLMConnect **Meaning**: Caller's space is not connected to WLM. **Action**: Invoke the IWMCONN macro before invoking this macro.

Table 129. Return and Reason Codes for the IWMSSEL Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx0854	**Equate Symbol**: IwmRsnCodeTooManySelect **Meaning**: The caller is attempting to select more work units than it has tasks to execute the work. **Action**: Wait until an execution task has issued IWMSTEND before attempting to select more work units.
8	xxxx0864	**Equate Symbol**: IwmRsnCodeSecondaryWorkExists **Meaning**: There are secondary work requests queued to this server task. The caller was expected to process them using IWMSSEM before calling IWMSSEL. **Action**: Select all secondary work requests before issuing IWMSSEL.
8	xxxx089E	**Equate Symbol**: IwmRsnCodeServiceAModeMismatch **Meaning**: The caller is in 64-bit address mode and tried to invoke a service macro that is only enabled for a 31-bit environment. **Action**: Use the 64-bit enabled service macro (IWM4SSL) or change the address mode of the caller to 31-bit.
C	—	**Equate Symbol**: IwmRetCodeEnvError **Meaning**: Environmental error.
C	xxxx0C01	**Equate Symbol**: IwmRsnCodeNoStg **Meaning**: Storage is not available for the request. **Action**: Caller must disconnect by invoking the IWMDISC macro.
C	xxxx0C14	**Equate Symbol**: IwmRsnCodeNoWorkShutDown **Meaning**: No work selected. Caller is to shutdown. **Action**: Caller must disconnect by invoking the IWMDISC macro.
C	xxxx0C3B	**Equate Symbol**: IwmRsnCodeStopTask **Meaning**: WLM decided to stop the server instance. **Action**: Calling task must shutdown, but server address space must remain active.
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error. **Action**: Contact your system programmer.

Example

To select a work request from the WLM queue manager queues:

```
         IWMSSEL 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
```

IWMSSEM — WLM server select secondary service

The IWMSSEM 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 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 equivalent service, IWM4SSM, which also supports 64-bit addressing. For more information, see "IWM4SSM — WLM server select secondary service" on page 746.

Environment

The requirements for the caller are:

Minimum authorization:	Problem state. Any PSW key
Dispatchable unit mode:	Task
Cross memory mode:	PASN=HASN=SASN
AMODE:	31-bit
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks may be held.
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, 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

The syntax of the IWMSSEM macro is as follows:

```
►►──┬──────┬──IWMSSEM──USERDATA=userdata──┬─────────────────┬──┬─────────────────┬──────────►
    └─name─┘                              └─,RETCODE=retcode─┘  └─,RSNCODE=rsncode─┘

   ┌─,PLISTVER=IMPLIED_VERSION─┐ ┌─,MF=S───────────────────────────────────┐
►──┼───────────────────────────┼─┼─────────────────────────────────────────┼──────────────►◄
   ├─,PLISTVER=MAX─────────────┤ │                      ┌─,0D───┐          │
   └─,PLISTVER=0───────────────┘ ├─,MF=(L─,list addr──┼───────┼─)───────┤
                                 │                      └─,attr─┘          │
                                 │                      ┌─,COMPLETE─┐      │
                                 └─,MF=(E─,list addr──┴───────────┴─)────┘
```

Parameters

The parameters are explained as follows:

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 codes and reason codes

When the IWMSSEM 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 130. Return and Reason Codes for the IWMSSEM Macro

Return Code	Reason Code	Equate Symbol, Meaning, and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0801	**Equate Symbol**: IwmRsnCodeSrbMode **Meaning**: Caller is in SRB mode. **Action**: Avoid requesting this function while in SRB mode.
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: Caller is locked. **Action**: Avoid requesting this function while locked.
8	xxxx080B	**Equate Symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list. **Action**: 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	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: Caller invoked service but was in 24 bit addressing mode. **Action**: Request this function only when you are in 31 bit addressing mode.
8	xxxx0825	**Equate Symbol**: IwmRsnCodeAscModeNotPrimary **Meaning**: Caller invoked service but was not in primary ASC mode. **Action**: Avoid requesting this function in this environment.
8	xxxx0828	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: Version number in parameter list is not valid. **Action**: Check for possible storage overlay of the parameter list.

Table 130. Return and Reason Codes for the IWMSSEM Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx0840	**Equate Symbol**: IwmRsnCodeServiceNotEnabled **Meaning**: Caller's space connection is not enabled for this service **Action**: Make sure that SERVER_MANAGER=YES is specified on the IWMCONN request to enable this service.
8	xxxx0841	**Equate Symbol**: IwmRsnCodeXmemMode **Meaning**: Caller is in cross-memory mode. **Action**: Request this function only when you are not in cross-memory mode.
8	xxxx0842	**Equate Symbol**: IwmRsnCodeNoWLMConnect **Meaning**: Caller's space is not connected to WLM. **Action**: Invoke the IWMCONN macro before invoking this macro.
8	xxxx0862	**Equate Symbol**: IwmRsnCodeNoPriorSelect **Meaning**: The caller has not previously selected work using the IWMSSEL macro. **Action**: Invoke the IWMSSEL macro before invoking this macro.
8	xxxx0863	**Equate Symbol**: IwmRsnCodeNoExecEnv **Meaning**: The caller has not established an execution environment using IWMSTBGN. **Action**: Invoke the IWMSTBGN macro before invoking this macro.
8	xxxx089E	**Equate Symbol**: IwmRsnCodeServiceAModeMismatch **Meaning**: The caller is in 64-bit address mode and tried to invoke a service macro that is only enabled for a 31-bit environment. **Action**: Use the 64-bit enabled service macro (IWM4SSM) or change the address mode of the caller to 31-bit.
C	—	**Equate Symbol**: IwmRetCodeEnvError **Meaning**: Environmental error.
C	xxxx0C14	**Equate Symbol**: IwmRsnCodeNoWorkShutDown **Meaning**: No work selected. Caller is to shutdown. **Action**: Caller must disconnect by invoking the IW DISC macro.
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error. **Action**: Contact your system programmer.

Example

To select a secondary work request from the WLM queue manager queues:

```
         IWMSSEM 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 — Begin a request from a caller's work manager queue

IWMSTBGN establishes the environment to process a work request that was previously selected using IWMSSEL. 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 equivalent service, IWM4STBG, which also supports 64-bit addressing. For more information, see “IWM4STBG — WLM begin server transaction service” on page 752.

Environment

The requirements for the caller are:

Minimum authorization:	Problem state. Any PSW key
Dispatchable unit mode:	Task
Cross memory mode:	PASN=HASN=SASN
AMODE:	31-bit
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks may be held.
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, 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 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.

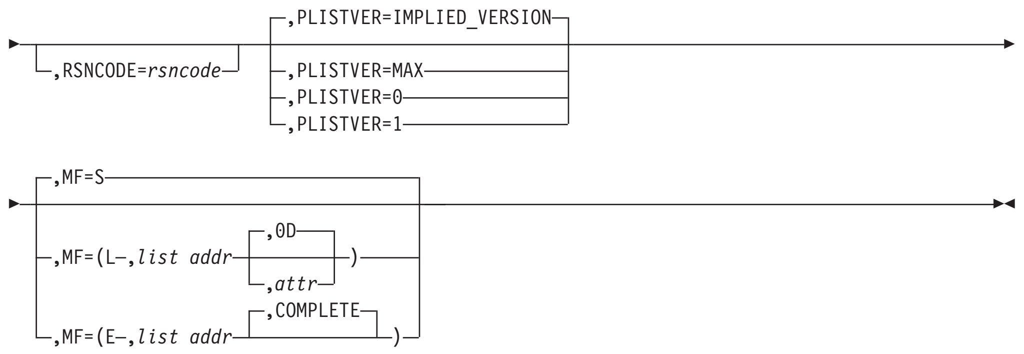
Performance implications

None.

Syntax

The syntax of the IWMSTBGN macro is as follows:

```
►►─┬──────┬─IWMSTBGN──WLMEUTKN=wlmeutkn──┬─────────────────┬──┬───────────────────┬──►
   └─name─┘                              └─,ETOKEN=etoken─┘  └─,RETCODE=retcode─┘
```



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*,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**, which supports all parameters except those specifically referenced in higher versions.
- **1**, which supports the following parameter 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).

WLMEUTKN=*wlmeutkn*
A required input parameter, execution unit token that was returned by a prior invocation of IWMSSEL.

To code: Specify the RS-type address, or address in register (2)-(12), of an 8-character field.

ABEND codes

None.

Return codes 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 131. Return and Reason Codes for the IWMSTBGN Macro

Return Code	Reason Code	Equate Symbol, Meaning, and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.
4	xxxx041F	**Equate Symbol**: IwmRsnCodeExecEnvChanged **Meaning**: The execution environment has changed while the requested function is in progress. **Action**: None required.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0801	**Equate Symbol**: IwmRsnCodeSrbMode **Meaning**: The caller is in SRB mode. **Action**: Avoid requesting this function while in SRB mode.
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: The caller is locked. **Action**: Avoid requesting this function while locked.
8	xxxx080B	**Equate Symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list. **Action**: 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	**Equate Symbol**: IwmRsnCodeEutFrr **Meaning**: The caller has EUT FRR established. **Action**: Avoid requesting this function with an EUT FRR set.

Table 131. Return and Reason Codes for the IWMSTBGN Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx0823	**Equate Symbol**: IwmRsnCodeDatoff **Meaning**: The caller invoked the service while DATOFF **Action**: Avoid requesting this function in this environment.
8	xxxx0824	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: The caller invoked the service but was in 24-bit addressing mode. **Action**: Request this function only when you are in 31-bit addressing mode.
8	xxxx0825	**Equate Symbol**: IwmRsnCodeAscModeNotPrimary **Meaning**: The caller invoked the service but was not DAT on Primary ASC mode. **Action**: Avoid requesting this function in this environment.
8	xxxx0826	**Equate Symbol**: IwmRsnCodeTaskTerm **Meaning**: The caller invoked the service while task termination is in progress for the task associated with the owner. **Action**: Avoid requesting this function while task termination is in progress.
8	xxxx0828	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: Version number in parameter list is not valid or version length field is not valid. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx083A	**Equate Symbol**: IwmRsnCodeBadEnclave **Meaning**: Enclave token does not pass verification. **Action**: Check for possible storage overlay of the enclave token, or asynchronous events which may have deleted the enclave.
8	xxxx0840	**Equate Symbol**: IwmRsnCodeServiceNotEnabled **Meaning**: The caller's space connection is not enabled for this service **Action**: Make sure that SERVER_MANAGER=YES is specified on the IWMCONN request to enable this service.
8	xxxx0841	**Equate Symbol**: IwmRsnCodeXmemMode **Meaning**: The caller is in cross-memory mode. **Action**: Request this function only when you are not in cross-memory mode.
8	xxxx0842	**Equate Symbol**: IwmRsnCodeNoWLMConnect **Meaning**: The caller's space is not connected to WLM. **Action**: Invoke the IWMCONN macro before invoking this macro.

Table 131. Return and Reason Codes for the IWMSTBGN Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx0850	**Equate Symbol**: IwmRsnCodeBeginEnvOutstanding **Meaning**: The caller is already operating under an outstanding Begin environment. **Action**: Avoid requesting this function in this environment.
8	xxxx0851	**Equate Symbol**: IwmRsnCodeSecEnvOutstanding **Meaning**: The caller is already operating under an outstanding security environment. **Action**: Avoid requesting this function while there is a task level security environment outstanding.
8	xxxx0852	**Equate Symbol**: IwmRsnCodeExecTokenNotCorrect **Meaning**: The execution unit token does not identify a previously selected work unit. **Action**: Verify that you have coded the WLMEUTKN parameter correctly.
8	xxxx0857	**Equate Symbol**: IwmRsnCodeAlreadyInEnclave **Meaning**: Current dispatchable work unit is already in an enclave. **Action**: Avoid requesting this function while the caller is already in an enclave.
8	xxxx085A	**Equate Symbol**: IwmRsnCodeSelectedWorkActive **Meaning**: The selected work element associated with the input execution unit token is already in execution. **Action**: You may have invoked IWMSTBGN from multiple tasks in the server address space passing the same WLMEUTKN. Avoid requesting this function in this environment.
8	xxxx089E	**Equate Symbol**: IwmRsnCodeServiceAModeMismatch **Meaning**: The caller is in 64-bit address mode and tried to invoke a service macro that is only enabled for a 31-bit environment. **Action**: Use the 64-bit enabled service macro (IWM4STBG) or change the address mode of the caller to 31-bit.
C	—	**Equate Symbol**: IwmRetCodeEnvError **Meaning**: Environmental error.
C	xxxx0C17	**Equate Symbol**: IwmRsnCodeSecEnvCreateFailed **Meaning**: A user security environment cannot be created. **Action**: Verify that the user ID is defined to RACF or check the SAF installation exit routine to enable the function.
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error. **Action**: Contact your system programmer.

Example

Suppose a work request was selected using IWMSSEL and the execution unit token returned by IWMSSEL 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
*                              IWMSSEL
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 equivalent service, IWM4STEN, which also supports 64-bit addressing. For more information, see "IWM4STEN — End a request from a caller's work manager queue" on page 760.

Environment

The requirements for the caller are:

Minimum authorization:	Problem state. Any PSW key
Dispatchable unit mode:	Task
Cross memory mode:	PASN=HASN=SASN
AMODE:	31-bit
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks may be held.
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, 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 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

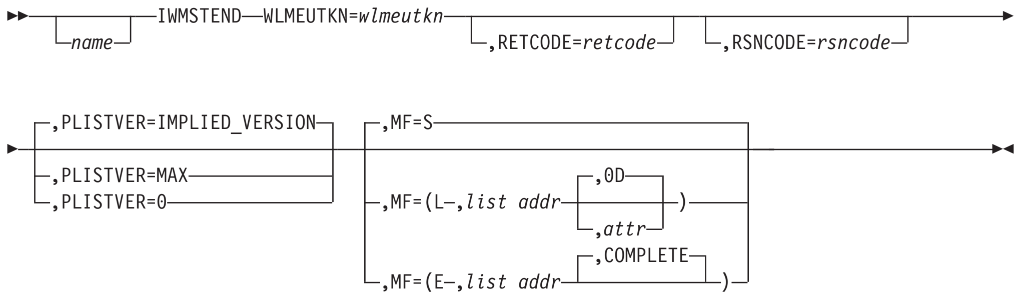
Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

Performance implications

None.

Syntax

The syntax of the IWMSTEND macro is as follows:



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 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 codes 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 132. Return and Reason Codes for the IWMSTEND Macro

Return Code	Reason Code	Equate Symbol, Meaning, and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted.
4	xxxx041C	**Equate Symbol**: IwmRsnCodeNotEnclave **Meaning**: The current dispatchable work unit is not associated with an enclave. **Action**: None required.
4	xxxx041F	**Equate Symbol**: IwmRsnCodeExecEnvChanged **Meaning**: The execution environment has changed while the requested function is in progress. **Action**: None required.

Table 132. Return and Reason Codes for the IWMSTEND Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
4	xxxx042F	**Equate Symbol**: IwmRsnCodeSecondaryWorkDeleted **Meaning**: 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. **Action**: Select all secondary work requests before issuing IWMSTEND
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0801	**Equate Symbol**: IwmRsnCodeSrbMode **Meaning**: The caller is in SRB mode. **Action**: Avoid requesting this function while in SRB mode.
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: The caller is locked. **Action**: Avoid requesting this function while locked.
8	xxxx080B	**Equate Symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list. **Action**: 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	**Equate Symbol**: IwmRsnCodeEutFrr **Meaning**: The caller has EUT FRR established. **Action**: Avoid requesting this function with an EUT FRR set.
8	xxxx0823	**Equate Symbol**: IwmRsnCodeDatoff **Meaning**: The caller invoked the service while DATOFF **Action**: Avoid requesting this function in this environment.
8	xxxx0824	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: The caller invoked service but was in 24-bit addressing mode. **Action**: Request this function only when you are in 31-bit addressing mode.
8	xxxx0825	**Equate Symbol**: IwmRsnCodeAscModeNotPrimary **Meaning**: The caller invoked the service but was not DAT on Primary ASC mode. **Action**: Avoid requesting this function in this environment.

Table 132. Return and Reason Codes for the IWMSTEND Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx0828	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: The Version number in the parameter list is not valid or version length field is not valid. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx0841	**Equate Symbol**: IwmRsnCodeXmemMode **Meaning**: Caller invoked service but was in cross-memory mode. **Action**: Avoid requesting this function in cross-memory mode.
8	xxxx084F	**Equate Symbol**: IwmRsnCodeWrongExecToken **Meaning**: Current dispatchable work unit is not associated with the input execution unit token. **Action**: Check for possible storage overlay of the execution unit token.
8	xxxx0859	**Equate Symbol**: IwmRsnCodeEnclaveSubTaskExists **Meaning**: The current dispatchable work unit has residual subtasks propagated to the enclave which are still associated with the enclave. **Action**: Avoid requesting this function in this environment.
8	xxxx089E	**Equate Symbol**: IwmRsnCodeServiceAModeMismatch **Meaning**: The caller is in 64-bit address mode and tried to invoke a service macro that is only enabled for a 31-bit environment. **Action**: Use the 64-bit enabled service macro (IWM4STEN) or change the address mode of the caller to 31-bit.
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error. **Action**: Contact your system programmer.

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 equivalent service, IWM4TAF, which also supports 64-bit addressing. For more information, see “IWM4TAF — WLM temporal affinity service” on page 766.

Environment

The requirements for the caller are:

Minimum authorization:	Problem state. Any PSW key
Dispatchable unit mode:	Task
Cross memory mode:	PASN=HASN=SASN
AMODE:	31-bit
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	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.

Restrictions

1. This macro may not be used during task/address space termination.
2. 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

The syntax of the IWMTAFF macro is as follows:

```
                                         ┌─,REGION_TOKEN=0────────────┐
►►──┬──────┬──IWMTAFF──┬─AFFINITY=YES─┬──┼────────────────────────────┼──────────────────►
    └─name─┘           └─AFFINITY=NO──┘  └─,REGION_TOKEN=region_token─┘

                                               ┌─,PLISTVER=IMPLIED_VERSION─┐
►──┬───────────────────┬──┬───────────────────┬──┼───────────────────────────┼──────────────►
   └─,RETCODE=retcode──┘  └─,RSNCODE=rsncode──┘  ├─,PLISTVER=MAX─────────────┤
                                               └─,PLISTVER=0───────────────┘
   ┌─,MF=S──────────────────────────────────┐
►──┼────────────────────────────────────────┼───────────────────────────────────►◄
   │                        ┌─,0D───┐       │
   ├─,MF=(L─,list addr──────┼───────┼─)─────┤
   │                        └─,attr─┘       │
   │                        ┌─,COMPLETE─┐   │
   └─,MF=(E─,list addr──────┴───────────┴─)─┘
```

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***,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

,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 service is used from the queueing manager. The region token is returned by the IWM4CON and IWMSSEL 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 codes 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 133. Return and Reason Codes for the IWMTAFF Macro

Return Code	Reason Code	Equate Symbol, Meaning, and Action
0	—	**Equate Symbol**: IwmRetCodeOk **Meaning**: Successful completion. **Action**: None required.
4	—	**Equate Symbol**: IwmRetCodeWarning **Meaning**: Successful completion, unusual conditions noted. **Action**: None required.
4	xxxx0439	**Equate Symbol**: IwmRsnCodeNoAffinityFound **Meaning**: 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. **Action**: 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	**Equate Symbol**: IwmRsnCodeRegionNotFound **Meaning**: The region token does not identify a valid server region. **Action**: Please specify the correct region token.
8	—	**Equate Symbol**: IwmRetCodeInvocError **Meaning**: Invalid invocation environment or parameters.
8	xxxx0801	**Equate Symbol**: IwmRsnCodeSrbMode **Meaning**: Caller is in SRB mode. **Action**: Avoid requesting this function while in SRB mode.

Table 133. Return and Reason Codes for the IWMTAFF Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx0803	**Equate Symbol**: IwmRsnCodeDisabled **Meaning**: Caller is disabled. **Action**: Avoid requesting this function while disabled.
8	xxxx0804	**Equate Symbol**: IwmRsnCodeLocked **Meaning**: Caller is locked. **Action**: Avoid requesting this function while locked.
8	xxxx080B	**Equate Symbol**: IwmRsnCodeBadPl **Meaning**: Error accessing parameter list. **Action**: 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	**Equate Symbol**: IwmRsnCodeAmode24 **Meaning**: Caller invoked service but was in 24 bit addressing mode. **Action**: Request this function only when you are in 31 bit addressing mode.
8	xxxx0825	**Equate Symbol**: IwmRsnCodeAscModeNotPrimary **Meaning**: Caller invoked service but was not in primary ASC mode. **Action**: Avoid requesting this function in this environment.
8	xxxx0828	**Equate Symbol**: IwmRsnCodeBadVersion **Meaning**: Version number in parameter list is not valid. **Action**: Check for possible storage overlay of the parameter list.
8	xxxx0840	**Equate Symbol**: IwmRsnCodeServiceNotEnabled **Meaning**: Caller's space connection is not enabled for this service **Action**: Make sure that SERVER_MANAGER=YES and SERVER_TYPE=QUEUE is specified on the IWMCONN request to enable this service.
8	xxxx0841	**Equate Symbol**: IwmRsnCodeXmemMode **Meaning**: Caller is in cross-memory mode. **Action**: Request this function only when you are not in cross-memory mode.
8	xxxx0842	**Equate Symbol**: IwmRsnCodeNoWLMConnect **Meaning**: Caller's space is not connected to WLM. **Action**: Invoke the IWMCONN macro before invoking this macro.

Table 133. Return and Reason Codes for the IWMTAFF Macro (continued)

Return Code	Reason Code	Equate Symbol, Meaning, and Action
8	xxxx084D	**Equate Symbol**: IwmRsnCodeNotAuthConnect **Meaning**: 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. **Action**: Avoid requesting this function in this environment.
8	xxxx089E	**Equate Symbol**: IwmRsnCodeServiceAModeMismatch **Meaning**: The caller is in 64-bit address mode and tried to invoke a service macro that is only enabled for a 31-bit environment. **Action**: Use the 64-bit enabled service macro (IWM4TAF) or change the address mode of the caller to 31-bit.
10	—	**Equate Symbol**: IwmRetCodeCompError **Meaning**: Component error. **Action**: Contact your system programmer.

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 F. Accessibility

Accessible publications for this product are offered through IBM Knowledge Center (http://www.ibm.com/support/knowledgecenter/SSLTBW/welcome).

If you experience difficulty with the accessibility of any z/OS information, send a detailed message to the "Contact us" web page for z/OS (http://www.ibm.com/systems/z/os/zos/webqs.html) or use the following mailing address.

IBM Corporation
Attention: MHVRCFS Reader Comments
Department H6MA, Building 707
2455 South Road
Poughkeepsie, NY 12601-5400
United States

Accessibility features

Accessibility features help users who have physical disabilities such as restricted mobility or limited vision use software products successfully. The accessibility features in z/OS can help users do the following tasks:

- Run assistive technology such as screen readers and screen magnifier software.
- Operate specific or equivalent features by using the keyboard.
- Customize display attributes such as color, contrast, and font size.

Consult assistive technologies

Assistive technology products such as screen readers function with the user interfaces found in z/OS. Consult the product information for the specific assistive technology product that is used to access z/OS interfaces.

Keyboard navigation of the user interface

You can access z/OS user interfaces with TSO/E or ISPF. The following information describes how to use TSO/E and ISPF, including the use of keyboard shortcuts and function keys (PF keys). Each guide includes the default settings for the PF keys.

- *z/OS TSO/E Primer*
- *z/OS TSO/E User's Guide*
- *z/OS V2R2 ISPF User's Guide Vol I*

Dotted decimal syntax diagrams

Syntax diagrams are provided in dotted decimal format for users who access IBM Knowledge Center with a screen reader. In dotted decimal format, each syntax element is written on a separate line. If two or more syntax elements are always present together (or always absent together), they can appear on the same line because they are considered a single compound syntax element.

Each line starts with a dotted decimal number; for example, `3` or `3.1` or `3.1.1`. To hear these numbers correctly, make sure that the screen reader is set to read out punctuation. All the syntax elements that have the same dotted decimal number (for example, all the syntax elements that have the number 3.1) are mutually exclusive alternatives. If you hear the lines 3.1 USERID and 3.1 SYSTEMID, your syntax can include either USERID or SYSTEMID, but not both.

The dotted decimal numbering level denotes the level of nesting. For example, if a syntax element with dotted decimal number 3 is followed by a series of syntax elements with dotted decimal number 3.1, all the syntax elements numbered 3.1 are subordinate to the syntax element numbered 3.

Certain words and symbols are used next to the dotted decimal numbers to add information about the syntax elements. Occasionally, these words and symbols might occur at the beginning of the element itself. For ease of identification, if the word or symbol is a part of the syntax element, it is preceded by the backslash (\) character. The * symbol is placed next to a dotted decimal number to indicate that the syntax element repeats. For example, syntax element *FILE with dotted decimal number 3 is given the format 3 * FILE. Format 3* FILE indicates that syntax element FILE repeats. Format 3* * FILE indicates that syntax element * FILE repeats.

Characters such as commas, which are used to separate a string of syntax elements, are shown in the syntax just before the items they separate. These characters can appear on the same line as each item, or on a separate line with the same dotted decimal number as the relevant items. The line can also show another symbol to provide information about the syntax elements. For example, the lines 5.1*, 5.1 LASTRUN, and 5.1 DELETE mean that if you use more than one of the LASTRUN and DELETE syntax elements, the elements must be separated by a comma. If no separator is given, assume that you use a blank to separate each syntax element.

If a syntax element is preceded by the % symbol, it indicates a reference that is defined elsewhere. The string that follows the % symbol is the name of a syntax fragment rather than a literal. For example, the line 2.1 %OP1 means that you must refer to separate syntax fragment OP1.

The following symbols are used next to the dotted decimal numbers.

? indicates an optional syntax element
The question mark (?) symbol indicates an optional syntax element. A dotted decimal number followed by the question mark symbol (?) indicates that all the syntax elements with a corresponding dotted decimal number, and any subordinate syntax elements, are optional. If there is only one syntax element with a dotted decimal number, the ? symbol is displayed on the same line as the syntax element, (for example 5? NOTIFY). If there is more than one syntax element with a dotted decimal number, the ? symbol is displayed on a line by itself, followed by the syntax elements that are optional. For example, if you hear the lines 5 ?, 5 NOTIFY, and 5 UPDATE, you know that the syntax elements NOTIFY and UPDATE are optional. That is, you can choose one or none of them. The ? symbol is equivalent to a bypass line in a railroad diagram.

! indicates a default syntax element
The exclamation mark (!) symbol indicates a default syntax element. A dotted decimal number followed by the ! symbol and a syntax element indicate that the syntax element is the default option for all syntax elements that share the same dotted decimal number. Only one of the syntax elements that share the dotted decimal number can specify the ! symbol. For example, if you hear the lines 2? FILE, 2.1! (KEEP), and 2.1 (DELETE), you know that (KEEP) is the

default option for the FILE keyword. In the example, if you include the FILE keyword, but do not specify an option, the default option KEEP is applied. A default option also applies to the next higher dotted decimal number. In this example, if the FILE keyword is omitted, the default FILE(KEEP) is used. However, if you hear the lines 2? FILE, 2.1, 2.1.1! (KEEP), and 2.1.1 (DELETE), the default option KEEP applies only to the next higher dotted decimal number, 2.1 (which does not have an associated keyword), and does not apply to 2? FILE. Nothing is used if the keyword FILE is omitted.

`* indicates an optional syntax element that is repeatable`
The asterisk or glyph (*) symbol indicates a syntax element that can be repeated zero or more times. A dotted decimal number followed by the * symbol indicates that this syntax element can be used zero or more times; that is, it is optional and can be repeated. For example, if you hear the line 5.1* data area, you know that you can include one data area, more than one data area, or no data area. If you hear the lines 3* , 3 HOST, 3 STATE, you know that you can include HOST, STATE, both together, or nothing.

Notes:

1. If a dotted decimal number has an asterisk (*) next to it and there is only one item with that dotted decimal number, you can repeat that same item more than once.
2. If a dotted decimal number has an asterisk next to it and several items have that dotted decimal number, you can use more than one item from the list, but you cannot use the items more than once each. In the previous example, you can write HOST STATE, but you cannot write HOST HOST.
3. The * symbol is equivalent to a loopback line in a railroad syntax diagram.

`+ indicates a syntax element that must be included`
The plus (+) symbol indicates a syntax element that must be included at least once. A dotted decimal number followed by the + symbol indicates that the syntax element must be included one or more times. That is, it must be included at least once and can be repeated. For example, if you hear the line 6.1+ data area, you must include at least one data area. If you hear the lines 2+, 2 HOST, and 2 STATE, you know that you must include HOST, STATE, or both. Similar to the * symbol, the + symbol can repeat a particular item if it is the only item with that dotted decimal number. The + symbol, like the * symbol, is equivalent to a loopback line in a railroad syntax diagram.

Notices

This information was developed for products and services offered in the U.S.A. or elsewhere.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, 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.

Policy for unsupported hardware

Various z/OS elements, such as DFSMS, HCD, JES2, JES3, and MVS, contain code that supports specific hardware servers or devices. In some cases, this device-related element support remains in the product even after the hardware devices pass their announced End of Service date. z/OS may continue to service element code; however, it will not provide service related to unsupported hardware devices. Software problems related to these devices will not be accepted for service, and current service activity will cease if a problem is determined to be associated with out-of-support devices. In such cases, fixes will not be issued.

Minimum supported hardware

The minimum supported hardware for z/OS releases identified in z/OS announcements can subsequently change when service for particular servers or devices is withdrawn. Likewise, the levels of other software products supported on a particular release of z/OS are subject to the service support lifecycle of those products. Therefore, z/OS and its product publications (for example, panels, samples, messages, and product documentation) can include references to hardware and software that is no longer supported.

- For information about software support lifecycle, see: IBM Lifecycle Support for z/OS (http://www.ibm.com/software/support/systemsz/lifecycle/)
- For information about currently-supported IBM hardware, contact your IBM representative.

Programming interface information

This information documents intended programming interfaces that allow you to write programs to obtain the services of z/OS.

Trademarks

Index

Product Number: 5650-ZOS

Printed in USA

SC34-2663-04