IMS

Common Service Layer Guide and Reference

Version 9

SC18-7816-00

IMS

Common Service Layer Guide and Reference

Version 9

SC18-7816-00

Note

Before using this information and the product it supports, be sure to read the general information under “Notices” on page 227.

First Edition (October 2004) (Softcopy Only)

This edition applies to Version 9 of IMS (product number 5655-J38) and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Figures

Tables

About This Book

This information is available as part of the DB2® Information Management Software Information Center for z/OS® Solutions. To view the information within the DB2 Information Management Software Information Center for z/OS Solutions, go to http://publib.boulder.ibm.com/infocenter/dzichelp. This information is also available in PDF and BookManager® formats. To get the most current versions of the PDF and BookManager formats, go to the IMS™ Library page at www.ibm.com/software/data/ims/library.html.

The *IMS Version 9: Common Service Layer Guide and Reference* helps IMS system programmers and system operators manage system administration and operations tasks across an IMS sysplex (hereafter called IMSplex).

Summary of Contents

This book includes the following information:

IBM Product Names Used in This Information

In this information, the licensed programs shown in Table 1 are referred to by their short names.

Table 1. Licensed Program Full Names and Short Names

Licensed program full name	Licensed program short name
IBM® Application Recovery Tool for IMS and DB2	Application Recovery Tool
IBM CICS® Transaction Server for OS/390®	CICS
IBM CICS Transaction Server for z/OS	CICS
IBM DB2 Universal Database™	DB2 Universal Database
IBM DB2 Universal Database for z/OS	DB2 UDB for z/OS
IBM Enterprise COBOL for z/OS and OS/390	Enterprise COBOL
IBM Enterprise PL/I for z/OS and OS/390	Enterprise PL/I
IBM High Level Assembler for MVS™ & VM & VSE	High Level Assembler
IBM IMS Advanced ACB Generator	IMS Advanced ACB Generator
IBM IMS Batch Backout Manager	IMS Batch Backout Manager
IBM IMS Batch Terminal Simulator	IMS Batch Terminal Simulator
IBM IMS Buffer Pool Analyzer	IMS Buffer Pool Analyzer
IBM IMS Command Control Facility for z/OS	IMS Command Control Facility
IBM IMS Connect for z/OS	IMS Connect

Table 1. Licensed Program Full Names and Short Names (continued)

Licensed program full name	Licensed program short name
IBM IMS Connector for Java™	IMS Connector for Java
IBM IMS Database Control Suite	IMS Database Control Suite
IBM IMS Database Recovery Facility for z/OS	IMS Database Recovery Facility
IBM IMS Database Repair Facility	IMS Database Repair Facility
IBM IMS DataPropagator™ for z/OS	IMS DataPropagator
IBM IMS DEDB Fast Recovery	IMS DEDB Fast Recovery
IBM IMS Extended Terminal Option Support	IMS ETO Support
IBM IMS Fast Path Basic Tools	IMS Fast Path Basic Tools
IBM IMS Fast Path Online Tools	IMS Fast Path Online Tools
IBM IMS Hardware Data Compression-Extended	IMS Hardware Data Compression-Extended
IBM IMS High Availability Large Database (HALDB) Conversion Aid for z/OS	IBM IMS HALDB Conversion Aid
IBM IMS High Performance Change Accumulation Utility for z/OS	IMS High Performance Change Accumulation Utility
IBM IMS High Performance Load for z/OS	IMS HP Load
IBM IMS High Performance Pointer Checker for OS/390	IMS HP Pointer Checker
IBM IMS High Performance Prefix Resolution for z/OS	IMS HP Prefix Resolution
IBM Tivoli® NetView® for z/OS	Tivoli NetView for z/OS
IBM WebSphere® Application Server for z/OS and OS/390	WebSphere Application Server for z/OS
IBM WebSphere MQ for z/OS	WebSphere MQ
IBM WebSphere Studio Application Developer Integration Edition	WebSphere Studio
IBM z/OS	z/OS

How to Read Syntax Diagrams

The following rules apply to the syntax diagrams that are used in this information:

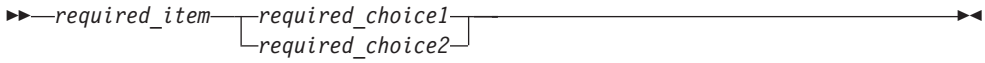
- Read the syntax diagrams from left to right, from top to bottom, following the path of the line. The following conventions are used:
  - The >>--- symbol indicates the beginning of a syntax diagram.
  - The ---> symbol indicates that the syntax diagram is continued on the next line.
  - The >--- symbol indicates that a syntax diagram is continued from the previous line.
  - The --->< symbol indicates the end of a syntax diagram.
- Required items appear on the horizontal line (the main path).

- Optional items appear below the main path.

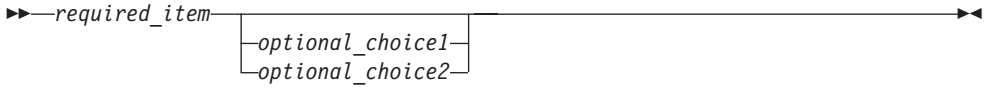
If an optional item appears above the main path, that item has no effect on the execution of the syntax element and is used only for readability.

- If you can choose from two or more items, they appear vertically, in a stack.

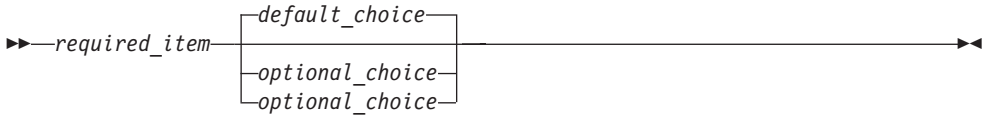
  If you *must* choose one of the items, one item of the stack appears on the main path.



  If choosing one of the items is optional, the entire stack appears below the main path.

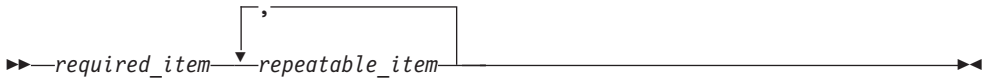


  If one of the items is the default, it appears above the main path, and the remaining choices are shown below.

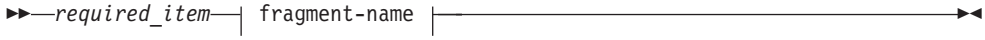


- An arrow returning to the left, above the main line, indicates an item that can be repeated.

  If the repeat arrow contains a comma, you must separate repeated items with a comma.



  A repeat arrow above a stack indicates that you can repeat the items in the stack.
- Sometimes a diagram must be split into fragments. The syntax fragment is shown separately from the main syntax diagram, but the contents of the fragment should be read as if they are on the main path of the diagram.



fragment-name:

- In IMS, a b symbol indicates one blank position.
- Keywords, and their minimum abbreviations if applicable, appear in uppercase. They must be spelled exactly as shown. Variables appear in all lowercase italic letters (for example, *column-name*). They represent user-supplied names or values.
- Separate keywords and parameters by at least one space if no intervening punctuation is shown in the diagram.
- Enter punctuation marks, parentheses, arithmetic operators, and other symbols, exactly as shown in the diagram.
- Footnotes are shown by a number in parentheses, for example (1).

How to Send Your Comments

Your feedback is important in helping us provide the most accurate and highest quality information. If you have any comments about this or any other IMS information, you can take one of the following actions:

- Go to the IMS Library page at www.ibm.com/software/data/ims/library.html and click the Library Feedback link, where you can enter and submit comments.
- Send your comments by e-mail to imspubs@us.ibm.com. Be sure to include the title, the part number of the title, the version of IMS, and, if applicable, the specific location of the text on which you are commenting (for example, a page number in the PDF or a heading in the Information Center).

Summary of Changes

Changes to This Book for IMS Version 9

This edition contains information related to the following IMS Version 9 functional line items and other quality line items:

- IMSplex Commands - see page 78.
- IMS Type-2 commands - see “A Simplified CSL Configuration” on page 3 and other additions to Chapter 1, “Common Service Layer Introduction,” on page 1. See also “CSL OM Command Security” on page 39.
- Significant changes to XML output samples have been made in Appendix A, “CSL Operations Manager XML Output,” on page 203.
- The term “enhanced command” has been replaced with “type-2 command”.

This edition also includes new information about:

- The IMS Application Menu, from which you can start the TSO SPOC, among others. Refer to the index for locations of this information.
- A new reason code (X'00000048') from the INIT OLCSTAT command regarding CSLRMPRS, which is added to the description of the OUTPUT= parameter for CSLRMPRS.

Library Changes for IMS Version 9

Changes to the IMS Library for IMS Version 9 include the addition of one title, a change of one title, organizational changes, and a major terminology change. Changes are indicated by a vertical bar (|) to the left of the changed text.

The IMS Version 9 information is now available in the DB2 Information Management Software Information Center for z/OS Solutions, which is available at http://publib.boulder.ibm.com/infocenter/dzichelp. The DB2 Information Management Software Information Center for z/OS Solutions provides a graphical user interface for centralized access to the product information for IMS, IMS Tools, DB2 Universal Database (UDB) for z/OS, DB2 Tools, and DB2 Query Management Facility (QMF™).

New and Revised Titles

The following list details the major changes to the IMS Version 9 library:

- *IMS Version 9: IMS Connect Guide and Reference*

  The library includes new information: *IMS Version 9: IMS Connect Guide and Reference*. This information is available in softcopy format only, as part of the DB2 Information Management Software Information Center for z/OS Solutions, and in PDF and BookManager formats.

  IMS Version 9 provides an integrated IMS Connect function, which offers a functional replacement for the IMS Connect tool (program number 5655-K52). In this information, the term *IMS Connect* refers to the integrated IMS Connect function that is part of IMS Version 9, unless otherwise indicated.
- The information formerly titled *IMS Version 8: IMS Java User's Guide* is now titled *IMS Version 9: IMS Java Guide and Reference*. This information is available in softcopy format only, as part of the DB2 Information Management Software Information Center for z/OS Solutions, and in PDF and BookManager formats.

- To complement the IMS Version 9 library, a new book, *An Introduction to IMS* by Dean H. Meltz, Rick Long, Mark Harrington, Robert Hain, and Geoff Nicholls (ISBN # 0-13-185671-5), is available starting February 2005 from IBM Press. Go to the IMS Web site at www.ibm.com/ims for details.

Organizational Changes

Organization changes to the IMS Version 9 library include changes to:

- *IMS Version 9: IMS Java Guide and Reference*
- *IMS Version 9: Messages and Codes, Volume 1*
- *IMS Version 9: Utilities Reference: System*

The chapter titled ″DLIModel Utility″ has moved from *IMS Version 9: IMS Java Guide and Reference* to *IMS Version 9: Utilities Reference: System*.

The DLIModel utility messages that were in *IMS Version 9: IMS Java Guide and Reference* have moved to *IMS Version 9: Messages and Codes, Volume 1*.

Terminology Changes

IMS Version 9 introduces new terminology for IMS commands:

type-1 command
A command, generally preceded by a leading slash character, that can be entered from any valid IMS command source. In IMS Version 8, these commands were called *classic* commands.

type-2 command
A command that is entered only through the OM API. Type-2 commands are more flexible than type-2 commands and can have a broader scope. In IMS Version 8, these commands were called *IMSplex* commands or *enhanced* commands.

Accessibility Enhancements

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use software products. The major accessibility features in z/OS products, including IMS, enable users to:

- Use assistive technologies such as screen readers and screen magnifier software
- Operate specific or equivalent features using only the keyboard
- Customize display attributes such as color, contrast, and font size

User Assistive Technologies

Assistive technology products, such as screen readers, function with the IMS user interfaces. Consult the documentation of the assistive technology products for specific information when you use assistive technology to access these interfaces.

Accessible Information

Online information for IMS Version 9 is available in BookManager format, which is an accessible format. All BookManager functions can be accessed by using a keyboard or keyboard shortcut keys. BookManager also allows you to use screen readers and other assistive technologies. The BookManager READ/MVS product is included with the z/OS base product, and the BookManager Softcopy Reader (for workstations) is available on the IMS Licensed Product Kit (CD), which you can download from the Web at www.ibm.com.

Keyboard Navigation of the User Interface

Users can access IMS user interfaces using TSO/E or ISPF. Refer to the *z/OS V1R1.0 TSO/E Primer*, the *z/OS V1R5.0 TSO/E User's Guide*, and the *z/OS V1R5.0 ISPF User's Guide, Volume 1*. These guides describe how to navigate each interface, including the use of keyboard shortcuts or function keys (PF keys). Each guide includes the default settings for the PF keys and explains how to modify their functions.

Chapter 1. Common Service Layer Introduction

This topic provides an introduction to the concepts of the IMS Common Service Layer.

What Is The CSL?

The IMS *Common Service Layer* (CSL) is a collection of IMS manager address spaces that provide the infrastructure needed for systems management tasks. The CSL address spaces include Operations Manager (OM), Resource Manager (RM), and Structured Call Interface (SCI). They are described in "CSL Managers" on page 3.

The CSL is built on the IMS Base Primitive Environment (BPE) layer. Therefore, all commands, messages, abends, configurations, and user exit interfaces that apply to BPE also apply to all CSL manager address spaces.

The IMS CSL provides the following:

- Improved systems management
- A single system image
- Ease of use through a single point of control
- Shared resources across all IMS systems

The role of the CSL is described in "The CSL in An IMSplex." If you do not use shared queues or sysplex technology, you can take advantage of a simplified CSL configuration to issue *type-2 commands* through the CSL OM. This is described in "A Simplified CSL Configuration" on page 3.

The CSL in An IMSplex

The IMS CSL reduces the complexity of managing multiple IMS systems by providing you with a single-image perspective in an *IMSplex*. An IMSplex is one or more IMS subsystems (control, manager, or server) that can work together as a unit. Typically, but not always, these subsystems:

- Share either databases or resources or message queues (or any combination)
- Run in a z/OS sysplex environment
- Include an IMS CSL

Within an IMSplex, you can now manage multiple IMS subsystems as if they were one system. For example, instead of entering commands on each IMS system during local online change, you can enter commands from one single point of control, and the commands will run on each IMS system in the IMSplex. An IMSplex can also exist in a non-sysplex environment. Use of the CSL is optional.

An IMSplex component is an IMS-defined entity that typically runs in its own address space; it manages resources, manages operations, or facilitates

communications between other IMS-defined entities. After the necessary started procedures for these components are initialized, they become *IMSplex members*. Examples of IMSplex components are:

- IMS subsystems (DB/DC, DBCTL, DCCTL)
- Resource Manager
- Operations Manager
- Structured Call Interface
- A DLIBATCH or DBBBATCH region

A DLIBATCH or DBBBATCH region is considered a special type of IMSplex component in that it does not interact with RMs and OMs.

A typical IMSplex environment including a CSL is shown in Figure 1. There are three separate operating system (OS) images. Each OS has an IMS control region and an SCI. OS1, in addition, has an OM and an RM. All three OS images share a coupling facility, which has database sharing structures, a message queue structure, and a resource structure.

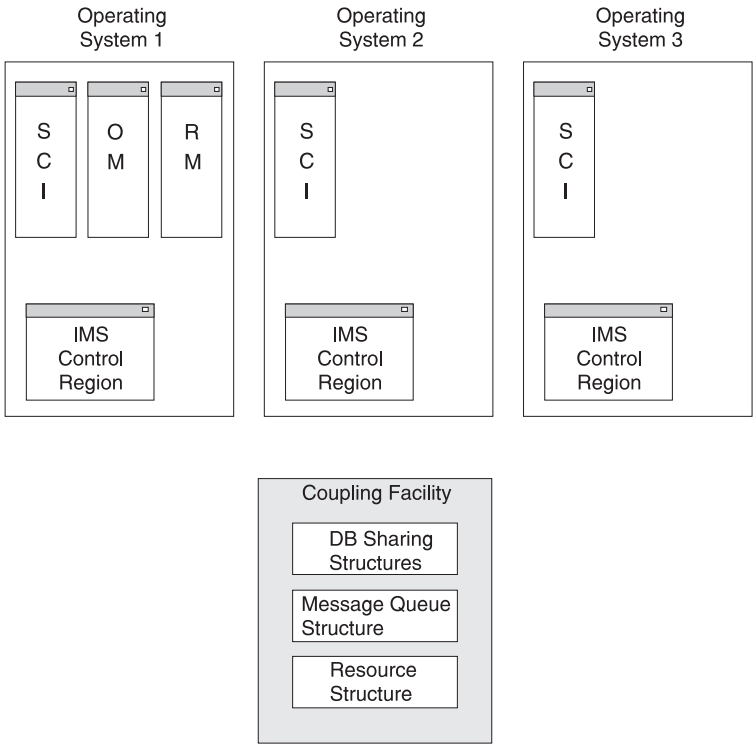


Figure 1. IMSplex Environment Including a CSL

The CSL is composed of three IMS address spaces:

- OM
- RM
- SCI

The address spaces that can participate in an IMSplex are:

- IMS control region address spaces
- IMS CSL manager address spaces (OM, RM, SCI)
- IMS server address spaces (Common Queue Server)

- Non-IMS address spaces

The IMSIDs of IMSs in the IMSplex must be unique.

A Simplified CSL Configuration

If your IMS configuration does not require RM services, you can still use OM and SCI to take advantage of type-2 IMS commands. Prior to IMS Version 9, type-2 commands required an RM. The commands could only be issued using the TSO SPOC, the IMS Control Center, or any user-written or vendor-written automation program that used the OM API. You can now issue type-2 commands using OM in a CSL configured without an RM.

The IMS Application Menu provides a common interface to enable you to start applications such as TSO SPOC, Syntax Checker, IVP, and more. For more information about the IMS Application Menu, see *IMS Version 9: Installation Volume 1: Installation Verification*.

For more information on type-2 commands, see the *IMS Version 9: Command Reference*.

CSL Managers

The CSL is a key component in the evolving architecture of IMS. Using the CSL in an IMSplex provides you with the infrastructure for improved systems management. The CSL address spaces are also referred to as CSL managers.

CSL Operations Manager

OM helps you control the operations of all IMS systems in an IMSplex. OM receives processing control when an OM request (an IMS command, for example) is received by the OM application programming interface (API). All commands and responses to those commands must come through the OM API. You can access this API through a Single Point of Control (SPOC) application, such as the TSO SPOC, REXX SPOC API, or a SPOC that you develop.

The OM API is described in "CSLOMI: API Request" on page 63. The REXX SPOC is described in Appendix B, "REXX SPOC API and the CSL," on page 219.

In an IMSplex, OM:

- Routes IMS commands to IMSplex members registered for the command.
- Consolidates command responses from individual IMSplex members into a single response to present to the command originator.
- Provides a programming API for IMS commands for automation.
- Provides a programming interface to support any command processing client.
- Provides user exits for input command and output response modification and security customization.

To use OM functions, an IMSplex must include at least one OM. There can be one or more OM on each z/OS image. Any OM can process work from a z/OS image within an IMSplex. Configuration examples are provided in "CSL Configuration Examples" on page 7.

For more information on OM, see Chapter 3, "CSL Operations Manager," on page 31.

CSL Resource Manager

RM helps you manage resources that are shared by multiple IMS systems in an IMSplex. RM provides the infrastructure for managing global resource information and coordinating IMSplex-wide processes. RM provides the following functions to an IMSplex:

- Maintains global resource information in a *resource structure*, which is a coupling facility list structure that all RMs in the IMSplex can access.
- Ensures resource consistency so that a resource defined as a transaction, lterm, or msname is defined as the same resource type for all IMSs in the IMSplex.
- Supports resource services.
- Supports client services.
- Uses the Common Queue Server (CQS) to maintain global resource information.
- Coordinates IMSplex-wide processes (such as global online change).

To use RM functions, an IMSplex must include at least one RM, and each IMS must specify that RM is to be used. You can configure your IMS system without an RM, but with OM and SCI, to take advantage of type-2 commands. Configuration examples are provided in "CSL Configuration Examples" on page 7. For more information on RM, see Chapter 4, "CSL Resource Manager," on page 97.

CSL Structured Call Interface

This SCI allows IMSplex members to communicate with one another. Communication between IMSplex members can occur within a single z/OS image or among multiple z/OS images. The individual IMSplex members do not need to know where the other members reside or what communication interface to use.

SCI provides the following functions to an IMSplex:

- Routes messages and requests within an IMSplex.
- Registers and deregisters IMSplex members.
- Notifies IMSplex members when a member joins or leaves the IMSplex.
- Provides security authentication of members when they join the IMSplex.
- Provides a single call interface to isolate the client and server from the underlying communications technology.

Any IMSplex member that requires SCI services must have an SCI on its z/OS image. There can be at most one SCI address space per IMSplex on each z/OS image. Configuration examples are provided in "CSL Configuration Examples" on page 7.

Note: DBRC RECON Loss Notification enhancement uses SCI, but not RM or OM. Although it is recommended that you initialize each CSL manager address space in an IMSplex, you can bring up an SCI address space without requiring the other CSL manager address spaces for DBRC RECON Loss Notification.

For more information on SCI and its operations, see Chapter 5, "CSL Structured Call Interface," on page 149.

Using a Single Point of Control (SPOC) Program in CSL

An IMS single point of control (SPOC) is a program with which you can manage operations of all IMS systems within an IMSplex. A SPOC communicates with one OM address space; OM then communicates with all of the other IMS address spaces in the IMSplex, through SCI, as required for operations.

A SPOC provides the following functions to an IMSplex:

- It allows you to submit commands to all IMSs in the IMSplex from a single console.
- It displays consolidated command responses from multiple IMS address spaces.
- It sends a message to an IMS terminal, connected to any IMS control region in the IMSplex, by issuing the IMS /BROADCAST command.

Figure 2 shows a SPOC application in an IMSplex. The IMSplex in this example has three identical OS images: each has an IMS control region, an IMS CQS address space, and an SCI, OM, and RM. All three OS images share a coupling facility, which includes database sharing structures, a message queue structure, and a resource structure. the IMSplex configuration also includes shared databases and RECON data sets.

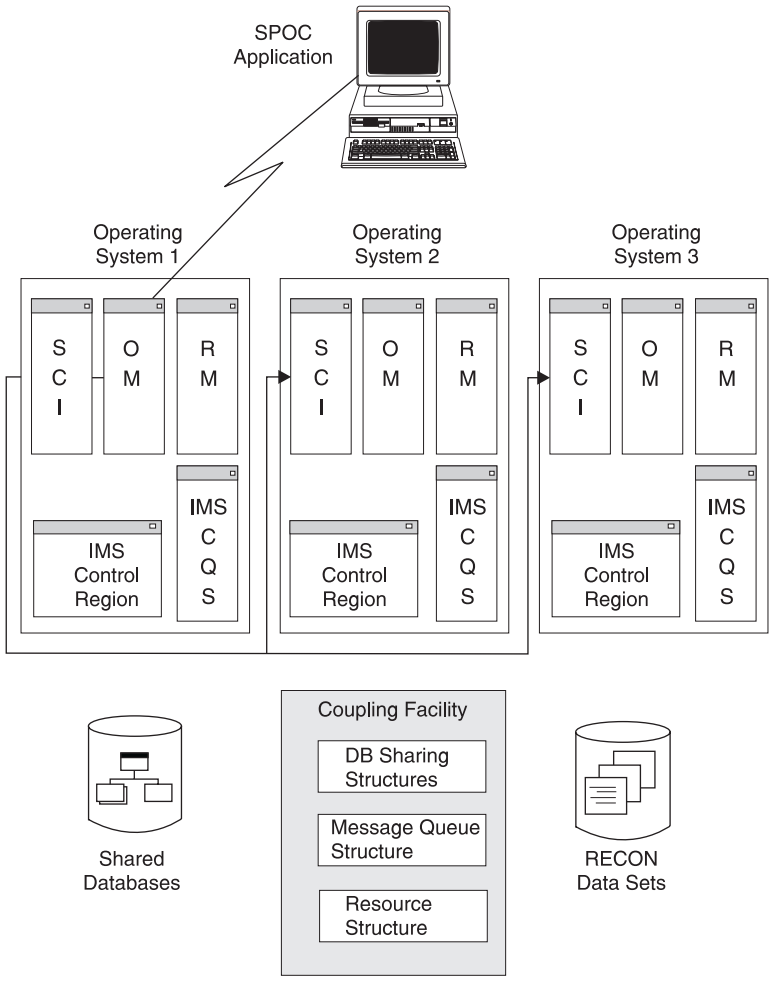


Figure 2. SPOC Application in an IMSplex

With a SPOC, you can issue commands to all members of an IMSplex at once. There can be more than one type of SPOC in an IMSplex, and there can be any number of SPOCs active, including:

TSO SPOC
: An IMS system management application comprised of an ISPF panel interface on a TSO terminal. It performs SPOC functions in an IMSplex. You can start the TSO SPOC using the IMS Application Menu. For more information about using the IMS Application Menu, see *IMS Version 9: Installation Volume 1: Installation Verification*. For more information on issuing commands with the TSO SPOC, see *IMS Version 9: Operations Guide*.

IMS Control Center
: An IMS system management application with a graphical user interface (GUI). It also performs SPOC functions in an IMSplex.

REXX program using the REXX SPOC API
: An application programming interface that allows automation programs to perform SPOC functions. See Appendix B, “REXX SPOC API and the CSL,” on page 219 for more information.

Vendor- or user-written SPOC
: A program written to use or access the OM API to perform SPOC functions.

Figure 3 shows an IMSplex environment with multiple SPOC users. The IMSplex environment includes four IMS control regions, each having its own SCI, and one OM. The multiple SPOC users include two SPOC TSO/ISPF applications, a REXX SPOC API, and a vendor-written SPOC program. Each SPOC can access the IMSplex environment.

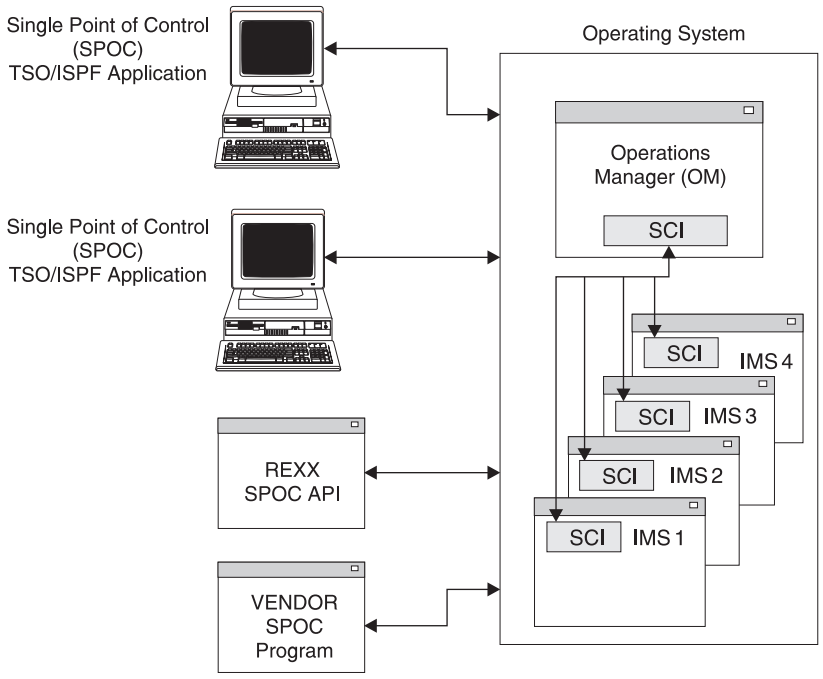


Figure 3. Multiple SPOC Users in an IMSplex

A SPOC is optional in an IMSplex. It uses the OM API macros to communicate with the IMSplex; however, you can write your own SPOC, and you can continue to use

your existing automated procedures. In addition, the existing command interfaces for the WTOR, MTO, and E-MCS console are still supported for type-1 commands only.

Note: Type-2 commands can only be issued from the TSO SPOC, the IMS Control Center, or other user- or vendor-written automation programs that use the OM API.

Related Reading: For more information about the TSO SPOC, see *IMS Version 9: Operations Guide*.

CSL Configuration Examples

When an IMS control region (DL/I, DBRC, dependent regions) requires the use of the CSL, SCI, OM, and RM are all required. However, different configurations are possible for the use of the CSL in an IMSplex. The basic configuration requirements for the CSL are:

Operations Manager
At least one OM must be available in the IMSplex. You can define additional OMs in the IMSplex for performance and availability enhancements.

Resource Manager
If any IMS in the IMSplex requires RM services, at least one RM must be available when the IMSplex is initialized. You can define additional RMs in the IMSplex for performance and availability enhancements if a resource structure is used. However, only one RM can be started in an IMSplex if a resource structure is not used.

Structured Call Interface
One SCI is required on each operating system image where an IMSplex member is running. Only one SCI is allowed on each operating system image for a given IMSplex.

Figure 4 on page 8 illustrates a sample IMSplex configuration that includes the CSL, a SPOC, and automated procedures.

- The OS image includes addresses spaces for OM, SCI, RM, an IMS control region, and IMS CQS.
- The OS image shares a coupling facility and databases.
- A SPOC application, an automation application, a master terminal, and an end user terminal all access the OS image.

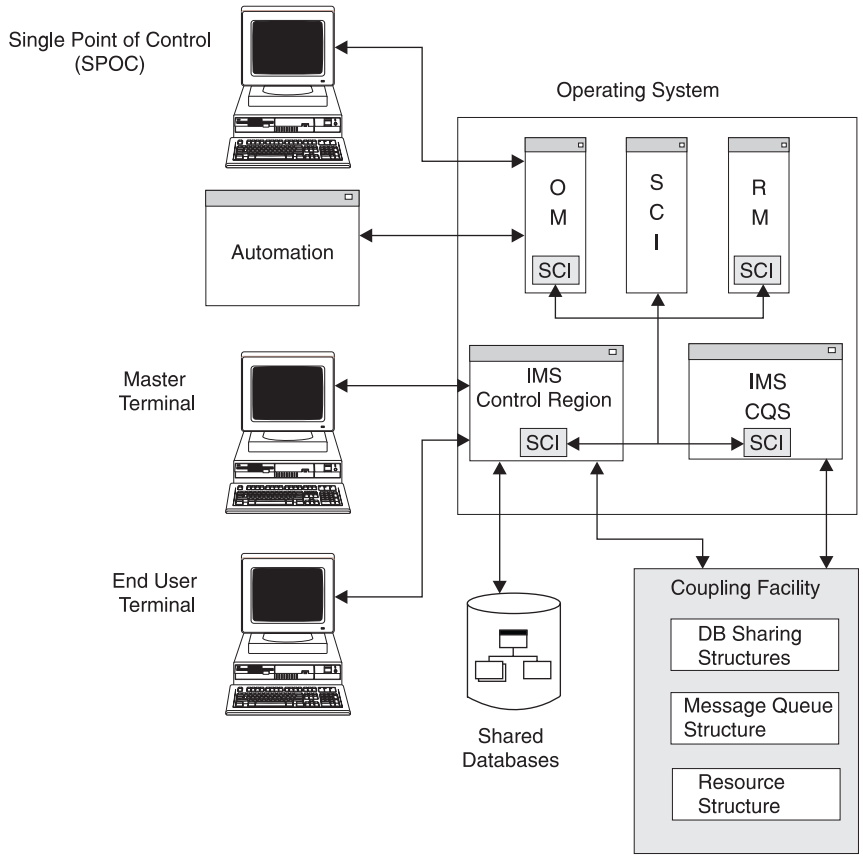


Figure 4. Sample IMSplex Configuration with CSL

To obtain the fastest communication, define an OM and RM on each z/OS image; an IMSplex component can communicate more quickly with an OM or RM on the same z/OS image as that component rather than an OM or RM on a different z/OS image. However, this configuration also increases the number of address spaces on each z/OS image, which can create a more complex operating environment.

Note: If only one RM and only one OM are defined across an IMSplex, there is no backup to perform that manager's work in case of failure.

Recommendation: Define more than one RM, OM, and SCI across an IMSplex.

Figure 5 on page 9 illustrates the minimum configuration possible for the CSL in an IMSplex. Each OS image has an IMS control region and an SCI; in addition, the first OS image also has an OM and RM.

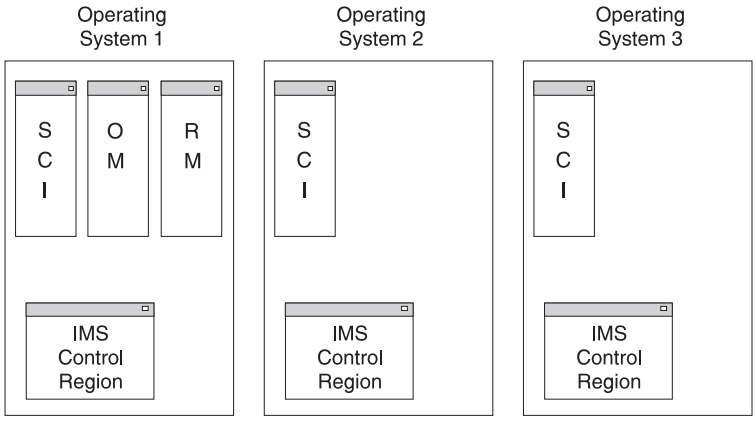


Figure 5. IMSplex Minimum CSL Configuration

In Figure 5, each z/OS image has a separate SCI since each has a distinct IMS control region. OM and RM can reside on one z/OS image and still be used by other images in the IMSplex.

Figure 6 on page 10 illustrates a more complex configuration having multiple versions of IMS within the IMSplex.

- OS1 has IMS Version 9 control regions and is the only OS image running an OM
- OS2 has IMS Version 8 control regions and an RM
- OS3 has an IMS Version 7 control region and an IMS Version 8 DBRC batch region

All three OS images share the coupling facility. Within the coupling facility is a resource structure. Therefore, an additional RM could be defined in another OS image. OM, RM and SCI are supported in an IMS Version 8 and IMS Version 9 system, but not an IMS Version 7 system.

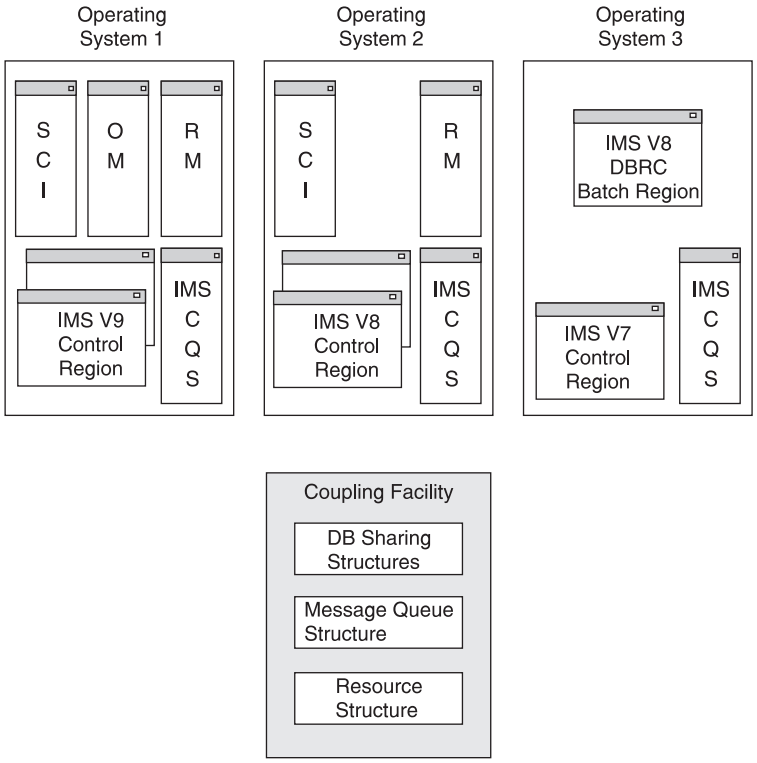


Figure 6. IMSplex Mixed Version CSL Configuration

Figure 7 illustrates the configuration of the Common Service Layer in a DBCTL environment. Here, three OS images each have an IMS DBCTL control region. All have an SCI and an OM. Only one RM is allowed in the IMSplex, here in OS1, because no resource structure is defined; however, it is recommended that you define a resource structure for DBCTL.

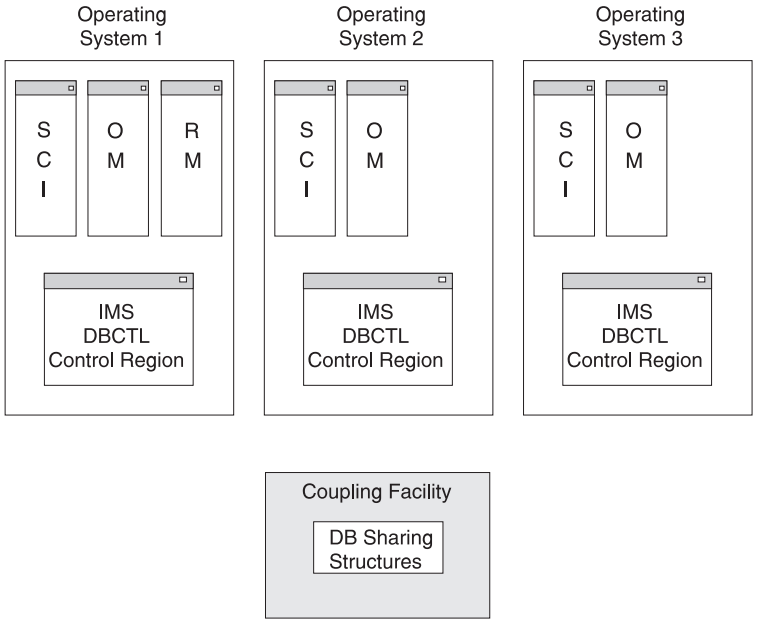


Figure 7. IMSplex DBCTL CSL Configuration

In Figure 7 on page 10, an OM is defined in each OS image to enhance overall system performance. If you do not use global online change, neither RM services nor an RM address space is required. In such a configuration, the CSL includes SCI and OM.

In other examples, Figure 8 and Figure 9 on page 12 illustrate what a shared queues IMSplex environment looks like, both with and without a CSL.

In Figure 8, three OS images are each defined with an IMS control region and IMS CQS. Each is associated with a Master Terminal Operator (MTO) console. All three OS images share a coupling facility that includes database sharing structures and a message queue structure. No CSL manager address spaces are defined.

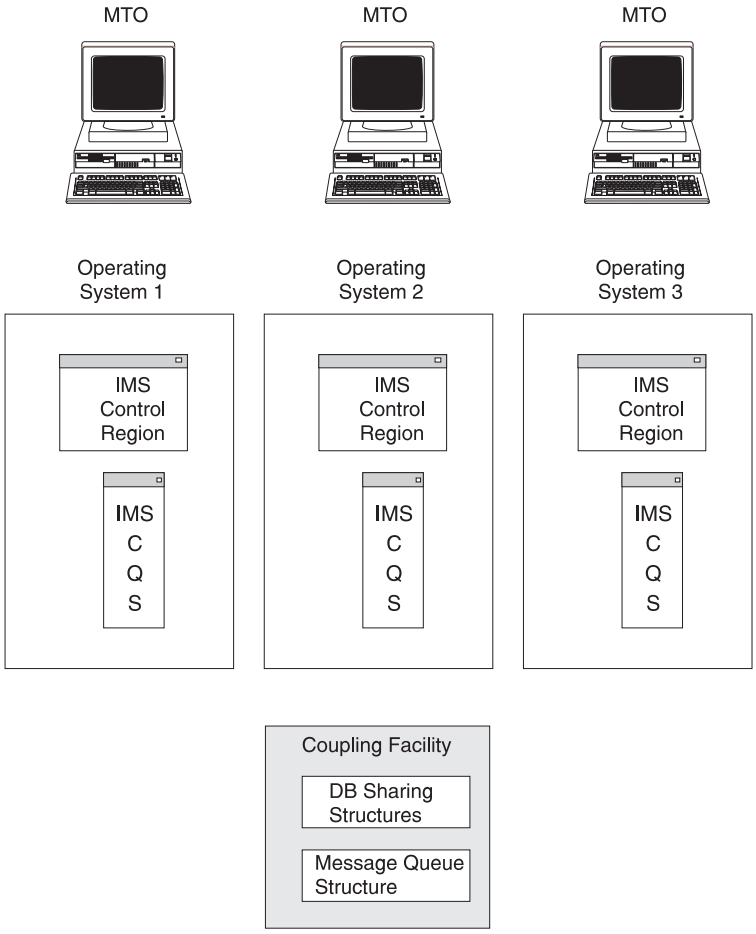


Figure 8. Shared Queues in an IMSplex without a CSL

In Figure 9 on page 12, three OS images are defined. Each is identical, having an IMS control region, IMS CQS, and all CSL managers (SCI, OM, and RM). The three OS images share the coupling facility, which includes database sharing structures, a message queue structure, and a resource structure. A SPOC application can access any of the OS images.

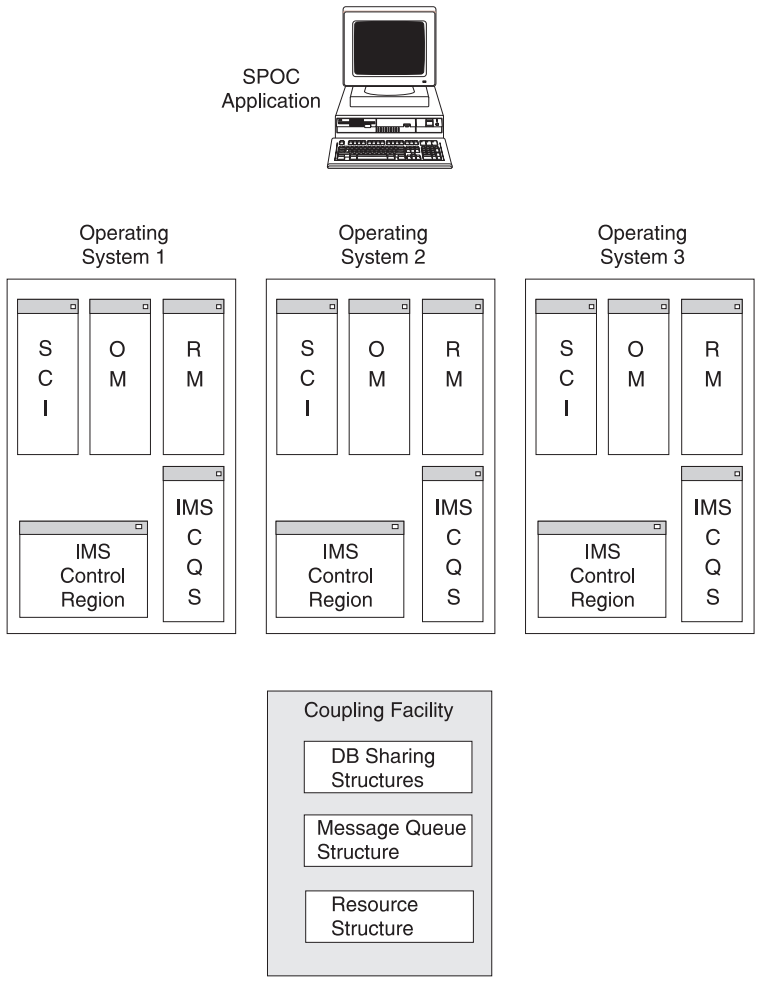


Figure 9. Shared Queues in an IMSplex Environment with a CSL

In a single system IMSplex environment, each address space of the Common Service Layer is defined in a single OS image. This simple configuration is shown in Figure 10. The OS image is defined with multiple IMS control regions, and one SCI, RM, and OM.

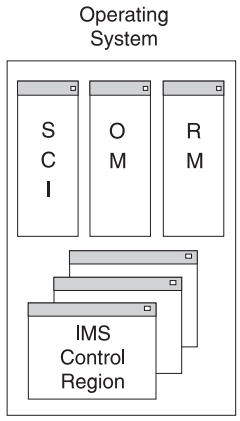


Figure 10. IMSplex Single System CSL Configuration

Chapter 2. Using The Common Service Layer in an IMSplex

In this topic, the basic operations of an IMS CSL in an IMSplex are described:

System Definition and Tailoring Considerations for the CSL

Use of the CSL is optional. To use the CSL, the system programmer must:

- Update the z/OS program properties table to include the CSL
- Define or change the procedures required to start and stop an IMSplex

Each of these tasks is described in this topic. Definition and tailoring information specific to each CSL manager is provided in these topics:

For additional information on IMS system definition and tailoring, refer to *IMS Version 9: Installation Volume 2: System Definition and Tailoring*.

Updating the z/OS Program Properties Table for the CSL

The system programmer must add an entry in the z/OS program properties table (PPT) for the CSL. The steps to add the entry are:

1. Edit the SCHEDxx member of the SYS1.PARMLIB data set.
2. Add the PPT entry `PGMNAME(BPEINI00)` to the SCHEDxx member, as shown in Figure 11:

```
PPT PGMNAME(BPEINI00)  /* PROGRAM NAME BPEINI00                    */
        CANCEL             /* PROGRAM CAN BE CANCELLED         (DEFAULT) */
    KEY(7)             /* PROTECT KEY ASSIGNED IS 7                */
    NOSWAP             /* PROGRAM IS NOT-SWAPPABLE                 */
    NOPRIV             /* PROGRAM NOT PRIVILEGED           (DEFAULT) */
    SYST               /* PROGRAM IS A SYSTEM TASK                 */
    DSI                /* REQUIRES DATA SET INTEGRITY      (DEFAULT) */
    PASS               /* PASSWORD PROTECTION ACTIVE       (DEFAULT) */
    AFF(NONE)          /* NO CPU AFFINITY                  (DEFAULT) */
    NOPREF             /* NO PREFERRED STORAGE FRAMES    (NODEFAULT) */
```

Figure 11. SCHEDxx member

Note: You can also use BPEINI00 to start CQS with the other CSL components. CQSINIT0 is still supported, but it is not needed.

3. Either re-IPL the z/OS system or issue the z/OS `SET SCH=` command.

For more information about updating the z/OS program properties table and editing the SCHED xx member, see *z/OS MVS Initialization and Tuning Reference*.

Defining PROCLIB Members for the CSL

This topic describes the IMSplex components and the procedure library (PROCLIB) members that apply to the CSL.

CQS PROCLIB Members

The IMS Common Queue Server (CQS) supports two PROCLIB members that are applicable to the CSL. CQS also supports the resource structure, which contains global resource information maintained by RM.

Two PROCLIB members apply to the CSL:

CQSIPxxx
Use the CQSIPxxx PROCLIB member to specify CQS initialization parameters and to initialize the CQS address space. The IMSPLEX() and NAME= parameters define the IMSplex name.

CQSSGxxx
Use the CQSSGxxx PROCLIB member to define global CQS parameters related to one or more coupling facility structures. The STRUCTURE() parameter defines a queue structure. The RSRCSTRUCTURE() parameter defines a resource structure. You must provide at least one STRUCTURE or RSRCSTRUCTURE definition. Both parameters can be defined to one CQS.

For complete information on these procedures and their parameters, see *IMS Version 9: Common Queue Server Guide and Reference*.

IMS PROCLIB Members

The IMS control region supports the PROCLIB members listed here, which are applicable to the CSL. For complete information on the IMS PROCLIB members that must be initialized for the CSL, see *IMS Version 9: Installation Volume 2: System Definition and Tailoring*.

DFSDCxxx
Use the IMS DC execution parameters PROCLIB member to define the status recovery mode of DC resources.

DFSVSMxxx
Use the OCMD= and CSLT= parameters to activate traces related to IMSplex activity.

DFSPBxxx
Use the CSLG= parameter to specify a 3-character suffix for the DFSCGxxx PROCLIB member.

DFSCGxxx
Use this PROCLIB member to specify parameters related to the CSL, OM, and RM. The suffix is specified on the CSLG= parameter. All IMSplex members that are in the same IMSplex group sharing databases, message queues, or both, must specify the same values except OLC=, which specifies either LOCAL or GLOBAL. The parameters are:

- CMDSEC=
- IMSPLEX=
- LEOPT=
- NORSCCC=
- OLC=
- OLCSTAT=

- RMENV=
- OMPROC=
- SCIPROC=

Recommendation: You should specify OM command security instead of IMS security. See *IMS Version 9: Installation Volume 2: System Definition and Tailoring* for more information on IMS security. Refer to "CSL OM Command Security" on page 39 for more information on OM command security.

Base Primitive Environment PROCLIB Members

All CSL components use the Base Primitive Environment (BPE). There are two BPE PROCLIB members that are applicable to each CSL component address space:

- BPE configuration PROCLIB member - contains statements that configure the BPE execution environment parameters.
- BPE user exit PROCLIB member - associates user exit points with one or more user exit modules.

Each CSL address space can have its own BPE configuration and user exit definition members, or they can share common members. For complete information on BPE PROCLIB members, see *IMS Version 9: Base Primitive Environment Guide and Reference*. Information on BPE considerations for each CSL manager is provided in these topics:

- "BPE Considerations for the CSL OM" on page 34
- "BPE Considerations for the CSL RM" on page 103
- "BPE Considerations for the CSL SCI" on page 151

CSL Manager PROCLIB Members

Each CSL manager has a separate initialization PROCLIB member that must be defined. See:

- "CSL OM Initialization Parameters PROCLIB Member" on page 35
- "CSL RM Initialization Parameters PROCLIB Member" on page 101
- "CSL SCI Initialization Parameters PROCLIB Member" on page 152

Global Online Change in a CSL

Global online change is an IMS function that uses the RM component of CSL to coordinate online change of resources across an IMSplex. This topic provides basic information about the differences between global online change and local online change. The CSL RM coordinates global online change activity, and that activity is described in more detail in "Enabling the Resource Manager for Global Online Change" on page 16.

For additional information on global online change, see "Making Global Online Changes in an IMSplex" in *IMS Version 9: Operations Guide*.

Comparing Global and Local Online Change

In an IMSplex, *global online change* allows you to perform online change to resources across the IMSplex. This is different than *local online change*, which performs online change to a local IMS system. To perform local online change in an IMSplex, you must manually coordinate local online change commands on the IMS systems. You issue separate commands to each IMS in the sysplex to prepare for, commit, and abort changes. In this example, changes can be committed on some IMSs in the sysplex, but not others. RM can optionally coordinate all phases of online change to achieve global online change on resources across an IMSplex.

The initial steps taken for local online change (for example, an ACBGEN followed by the online change copy utility, OLCUTL) are still required for global online change.

To enable global online change for an IMSplex, the system programmer must:
- Define the global OLCSTAT data set.
- Remove local MODSTAT definitions.
- Define the OLCSTAT data set name in the DFSCGxxx PROCLIB member.

For complete information on the tasks associated with enabling global online change, see *IMS Version 9: Administration Guide: Database Manager*. For information on the utilities associated with global online change, see *IMS Version 9: Utilities Reference: System*.

You can coordinate global online changes for the following resources:
- Databases
- Database directories
- MFS formats
- Programs
- Program directories
- Security matrices
- Transactions

Enabling the Resource Manager for Global Online Change

Because RM coordinates the global online change activity, it must be started and active in the IMSplex. RM can be optionally defined with a resource structure. If it is, global online change uses this resource structure to save IMSplex process status; IMS uses it to perform consistency checking between OLCSTAT and the online change libraries (ACBLIB, FMTLIB, and MODBLKS).

Using a resource structure provides more recovery capability during online change in the event of failure; however, it is not required for global online change.

If no resource structure is defined, the IMSplex can contain only one RM. In this example, no consistency checking is performed between OLCSTAT and the online change libraries, and IMS does not ensure that all IMSs in the IMSplex use the same OLCSTAT data set name.

For more information about the role of RM, refer to Chapter 4, “CSL Resource Manager,” on page 97.

General Guidelines for Writing CSL Requests

This topic documents general guidelines for writing OM, RM, and SCI requests for the CSL managers. This topic includes:

- “Using an ECB with CSL Requests”
- “CSL Manager Requests” on page 17
- “Releasing Storage with CSLSCBFR” on page 19
- “Environmental Requirements for SCI Requests” on page 19

Using an ECB with CSL Requests

Most CSL requests allow an ECB to be specified. The ECB= parameter is optional and specifies the address of the z/OS ECB. When a CSL request completes, the

ECB specified in on the ECB= parameter is posted. If the parameter is not included, the requesting module does not receive control back from SCI until the request has completed.

If an ECB is specified, the invoker of the request must issue a WAIT (or equivalent) after receiving control from the request, before using or examining any data returned by this request (including the RETCODE and RSNCODE fields).

CSL Manager Requests

Table 2 lists all of the CSL manager requests, where you can find that information, a brief description of their use, and the location of their return and reason codes, if applicable.

Table 2. A List of All of the CSL Manager Requests

Request Name	Described here	Use	Return and Reason Codes
CSLOMCMD	"CSLOMCMD: Command Request" on page 55	Used by an automated operator program client to send commands.	"CSLOMCMD Return and Reason Codes" on page 59
CSLOMI	"CSLOMI: API Request" on page 63	Used by z/OS automated operator client to issue IMS commands to an OM.	"CSLOMI Return and Reason Codes" on page 69
CSLOMQRY	"CSLOMQRY: Query Request" on page 74	Used by an automated operator client to request OM-specific information	"CSLOMQRY Return and Reason Codes" on page 78
CSLOMBLD	"CSLOMBLD: Command Registration Build" on page 79	Used to build the command list that is passed to OM on the CSLOMREG request.	Not applicable.
CSLOMDRG	"CSLOMDRG: Command Deregistration Request" on page 81	Used by command processing clients to tell OM that it no longer wants to process commands.	"CSLOMDRG Return and Reason Codes" on page 82
CSLOMOUT	"CSLOMOUT: Unsolicited Output Request" on page 82	Used by a command processing client to send a message not directly in response to a command.	"CSLOMOUT Return and Reason Codes" on page 84
CSLOMRDY	"CSLOMRDY: Ready Request" on page 84	Used by a command processing client to notify OM that it is ready to process commands.	"CSLOMRDY Return and Reason Codes" on page 85
CSLOMREG	"CSLOMREG: Command Registration Request" on page 85	Used by a command processing client to register commands with OM.	"CLSOMREG Return and Reason Codes" on page 87
CSLOMRSP	"CSLOMRSP: Command Response Request" on page 88	Used by a command processing client to respond to a command.	"CSLOMRSP Return and Reason Codes" on page 91

Table 2. A List of All of the CSL Manager Requests (continued)

Request Name	Described here	Use	Return and Reason Codes
CSLRMDEL	"CSLRMDEL: Delete Resources" on page 113	Used to delete one or more uniquely named resources on a resource structure.	"CSLRMDEL Return and Reason Codes" on page 116
CSLRMDRG	"CSLRMDRG: Deregister Clients" on page 117	Used by a client to communicate that it no longer wants to process resource requests from RM.	Not applicable.
CSLRMPRI	"CSLRMPRI: Process Initiate" on page 118	Used by a client to initiate a process across an IMSplex.	"CSLRMPRI Return and Reason Codes" on page 120
CSLRMPRR	"CSLRMPRR: Process Respond" on page 121	Used by a client to respond to a step in an IMSplex-wide process.	"CSLRMPRR Return and Reason Codes" on page 123
CSLRMPRS	"CSLRMPRS: Process Step" on page 123	Used by a client to perform a step in a process.	"CSLRMPRS Return and Reason Codes" on page 127
CSLRMPRT	"CSLRMPRT: Process Terminate" on page 129	Used by a client to terminate an IMSplex-wide process.	"CSLRMPRT Return and Reason Codes" on page 131
CSLRMQRY	"CSLRMQRY: Query Resources" on page 131	Used by a client to query one or more uniquely named resources on a resource structure.	"CSLRMQRY Return and Reason Codes" on page 135
CSLRMREG	"CSLRMREG: Register Clients" on page 136	Used to register a client and, optionally, the client's resource types and associated name types, to RM.	"CSLRMREG Return and Reason Codes" on page 138
CSLRMUPD	"CSLRMUPD: Update Resources" on page 139	Used to create a resource or update a previously created resource.	"CSLRMUPD Return and Reason Codes" on page 143
CSLSCBFR	"CSLSCBFR: Buffer Return Request" on page 171	Used to release storage that SCI allocated for an IMSplex member.	"CSLSCBFR Return and Reason Codes" on page 173
CSLSCDRG	"CSLSCDRG: Deregistration Request" on page 173	Used to break a connection between an IMSplex member and SCI.	"CSLSCDRG Return and Reason Codes" on page 174
CSLSCMSG	"CSLSCMSG: Send Message Request" on page 174	Used to send a message to one or more IMSplex members.	"CSLSCMSG Return and Reason Codes" on page 180
CSLSCQRY	"CSLSCQRY: Query Request" on page 181	Used by IMSplex member to obtain information about other members of an IMSplex.	"CSLSCQRY Return and Reason Codes" on page 184

Table 2. A List of All of the CSL Manager Requests (continued)

Request Name	Described here	Use	Return and Reason Codes
CSLSCQSC	"CSLSCQSC: Quiesce Request" on page 184	Used to tell SCI to stop routing messages and requests to the issuing IMSplex member.	"CSLSCQSC Return and Reason Codes" on page 185
CSLSCRDY	"CSLSCRDY: Ready Request" on page 186	Used to enable an IMSplex member to receive messages and requests.	"CSLSCRDY Return and Reason Codes" on page 187
CSLSCREG	"CSLSCREG: Registration Request" on page 187	Used to create a connection between an IMSplex member and SCI.	"CSLSCREG Return and Reason Codes" on page 192
CSLSCRQR	"CSLSCRQR Request Return Request" on page 193	Used to return a request to the IMSplex member from which the request originated.	"CSLSCRQR Return and Reason Codes" on page 195
CSLSCRQS	"CSLSCRQS: Send Request Request" on page 196	Used to allow an IMSplex member to send a request to another member in the IMSplex.	"CSLSCRQS Return and Reason Codes" on page 201

Releasing Storage with CSLSCBFR

The CSLSCBFR request, which is described more fully in "CSLSCBFR: Buffer Return Request" on page 171, is used to release storage that is obtained during the processing of OM and RM requests. You must issue this request to release storage, which would otherwise accumulate. If the storage is not released, an abend could occur.

Environmental Requirements for SCI Requests

For SCI requests, the environmental requirements depend on the SCI interface assigned to the client.

For clients using the authorized SCI interface, refer to Table 3:

Table 3. Environment for SCI Requests Using the Authorized Interface

Environmental Characteristic	Requirement
Authorization	Supervisor state and PSW key 0-7 (PSW key must match the PSW key when the CSLSCREG request was issued)
Dispatchable unit mode	Task
Cross memory mode	Any, however, PASN must equal the primary address space where the CSLSCREG request was issued
AMODE	31
ASC Mode	Primary
Home address space	Any
Locks	No locks held

Table 3. Environment for SCI Requests Using the Authorized Interface (continued)

Environmental Characteristic	Requirement
Interrupt status	Enabled for interrupts
Control parameters	In primary address space

For clients using the non-authorized SCI interface, refer to Table 4:

Table 4. Environment for SCI Requests Using the Non-Authorized Interface

Environmental Characteristic	Requirement
Authorization	Problem state or PSW key 8 (PSW key must match the PSW key when the CSLSCREG request was issued)
Dispatchable unit mode	Task
Cross memory mode	None (PASN=SASN=HASN)
AMODE	31
ASC Mode	Primary
Home address space	Address space where CSLSCREG was issued
Locks	No locks held
Interrupt status	Enabled for interrupts
Control parameters	In primary address space

The environmental requirements for the SCI register and deregister requests (`CSLSCREG` and `CSLSCDRG`) are different from all of the other SCI requests. Authorized clients must issue `CSLSCREG` and `CSLSCDRG` requests in the environment shown in Table 5:

Table 5. Environment for CSLSCREG and CSLSCDRG Requests Using the Authorized Interface

Environmental Characteristic	Requirement
Authorization	Supervisor state and PSW key 0-7
Dispatchable unit mode	Task
Cross memory mode	None (PASN=SASN=HASN)
AMODE	31
ASC Mode	Primary
Locks	No locks held
Interrupt status	Enabled for interrupts
Control parameters	In primary address space

Non-authorized clients must issue `CSLSCREG` and `CSLSCDRG` requests in the environment described in Table 6:

Table 6. Environment for CSLSCREG and CSLSCDRG Requests Using the Non-Authorized Interface

Environmental Characteristic	Requirement
Authorization	Problem state or PSW key 8

Table 6. Environment for CSLSCREG and CSLSCDRG Requests Using the Non-Authorized Interface (continued)

Environmental Characteristic	Requirement
Dispatchable unit mode	Task
Cross memory mode	None (PASN=SASN=HASN)
AMODE	31
ASC Mode	Primary
Locks	No locks held
Interrupt status	Enabled for interrupts
Control parameters	In primary address space

Considerations for Writing Clients for the CSL

This topic describes what you need to consider when you plan to write OM, RM, or SCI clients for your installation. A high-level sequence is provided, beginning with "Planning Considerations for Writing Clients for the CSL." The sequence is provided as an example only; your installation's planning and implementation of the CSL might vary.

Planning Considerations for Writing Clients for the CSL

Planning tasks are decisions that you must make to determine how you will use the CSL managers. These decisions include:

- **What authorization level to use**

  You must decide whether your program needs to run authorized (supervisor state, PSW key 0-7), or non-authorized (problem state, PSW key 8). SCI initializes the appropriate environment based on your program's state and PSW key when it registers with SCI.

  Note: A non-authorized client cannot register with RM, issue RM requests, register commands with OM, or process requests issued using CSLSCRQS.

- **Whether to use SCI exit routines**

  You must decide specifically whether to use the SCI Input and Notify exit routines. An OM command processing client, for example, must have the SCI Input exit to process OM directives; it must have the SCI Notify exit to be notified when new OMs join the IMSplex, so the OM command processing client can register to those OMs.

- **TCB Association**

  SCI registration (with the CSLSCREG request) enables an IMSplex member to be associated with a specific different TCB. The authorization level you use must also be considered regarding TCB association.

- **Whether to use RM services and OM services**

  You can choose to manage your own global resources. However, if you want to access IMS global resources, you must code an RM client.

  If you plan to develop your own command set and your own command processing client (that would coordinate its own command registration and security), you can write an OM command processing client. If you plan to develop your own SPOC or AOP to enter your own commands, you can write an

OM AOP client. OM's role is to transport commands throughout an IMSplex and to consolidate those command responses, in XML tags, for a SPOC or AOP.

- **Whether to use message or request protocol when issuing requests**

  Use message protocol either when you do not need a synchronous response, or when you want an asynchronous response. IMSplex command responses that are sent with message protocol are sent asynchronously.

Registering Clients to CSL Managers

To use any of the CSL managers, you must first complete registration steps. OM and RM clients must register with the SCI. This topic describes SCI registration and how OM and RM clients register with the SCI. It also describes how to set SCI to a ready state, and the sequence in which CSL requests must be issued:

- "Registering to SCI"
- "Registering an OM Client"
- "Registering an RM Client" on page 23
- "Enabling SCI Ready State" on page 23
- "Sequence for Coding CSL Requests" on page 23

Registering to SCI

Registering to SCI with the CSLSCREG request (described in "CSLSCREG: Registration Request" on page 187) is the foundation for using the CSL managers. It must be the first request issued. When you register to SCI, you identify:

- The name of the IMSplex.
- Your client name, which must be unique if it is an authorized client.
- Exit routines, if you elect to use them.
- Your type of address space.

  Recommendation: Use a type of AOP or OTHER for the address space. Defining your address space by a type that is not AOP or OTHER could interfere with IMS address spaces.

After you register to SCI, an SCI token is returned. The token uniquely identifies an IMSplex member's connection to SCI. It should be saved for future OM, RM, and SCI requests.

Registering an OM Client

OM clients can be categorized as command processing clients (see "Registering Command Processing Clients in a CSL" on page 37 and "Command Processing Clients and the CSL OM" on page 93) or AOP clients. AOP clients do not register to OM; command processing clients do. To register an OM command processing client:

1. Identify the SCI Notify exit with the CSLSCREG request; this allows the client to be notified when a new OM joins the IMSplex, so the client can register to the new OM.
2. Identify the SCI Input exits with the CSLSCREG request so that command processing clients can process OM directives.
3. Issue CSLSCQRY to determine which OMs are in the IMSplex.
4. Issue CSLOMREG to all OMs in the IMSplex that are reachable and ready to register a command list to OM.

It is important to consider the level of command security for OM. See "CSL OM Command Security" on page 39 for information on OM command security.

Recommendation: Command processing clients should return their responses using the XML statements described in Appendix A, "CSL Operations Manager XML Output," on page 203. This allows the responses to be viewed from a SPOC terminal and processed by an AOP.

Registering an RM Client

If you have resources that you want to manage, or if you want to access or use IMSplex-wide processes, you must register an RM client:

1. Identify the SCI Notify exit routine with the CSLSCREG request; this allows the client to be notified when a new RM joins the IMSplex, so the client can register to the new RM.
2. Identify the SCI Input exit routine with the CSLSCREG request; you must have an SCI input exit routine if you are using IMSplex-wide processes to handle RM directives.
3. Issue CSLSCQRY to determine which RMs are in the IMSplex.
4. Issue CSLRMREG to all RMs in the IMSplex that are reachable and ready.
5. Register the resource type and associated name type if you want RM to manage global resources.

Enabling SCI Ready State

With the SCI, there are two states: registered and ready. The CSLSCRDY request (described in "CSLSCRDY: Ready Request" on page 186) enables an IMSplex member to receive messages and requests routed by type. An IMSplex member that is registered but has not issued a CSLSCRDY request can process only messages and requests that are specifically directed to it.

Sequence for Coding CSL Requests

When you are ready to begin coding requests, note that there is a sequence in which the requests should be issued.

- Table 32 on page 92 lists the sequence for an AOP OM client running on the host.
- Table 33 on page 92 lists the sequence for an AOP OM client running on the workstation.
- Table 34 on page 93 lists the sequence for an OM command processing client.
- Table 44 on page 111 lists the sequence for an RM client.
- Table 45 on page 111 lists the sequence for an RM client participating in IMSplex-wide processes.

Sending Commands to the IMSplex

After setting up an IMSplex, you can issue commands using the TSO single point of control (SPOC) interface. You can also write an automated operator program that can issue the same commands. See *IMS Version 9: Installation Volume 1: Installation Verification* for sample invocation information for the TSO SPOC. For information on issuing commands using the TSO SPOC, refer to *IMS Version 9: Operations Guide*. The TSO SPOC also provides integrated online help.

The IMS Application Menu provides a common interface to enable you to start applications such as TSO SPOC, Syntax Checker, IVP, and more. For more information about the IMS Application Menu, see *IMS Version 9: Installation Volume 1: Installation Verification*.

If you write automated programs, those programs must include logic to handle the responses from the commands; the automated programs have to parse the XML statements described in Appendix A, “CSL Operations Manager XML Output,” on page 203.

Most commands that are issued to an IMSplex are issued to OM. The exceptions are:

- BPE commands, which can be issued directly to CSL members and to CQS.
- The SHUTDOWN command, which can be issued directly to SCI to shut down one or more CSL members.
- Query requests issued by a z/OS master console to the CSL.

In an IMSplex, the format, behavior, and responses to certain IMS commands has changed. Some IMS commands are no longer recoverable; others are not supported in an IMSplex. Some commands are supported only in an IMSplex. For complete information on submitting commands in an IMSplex, see *IMS Version 9: Command Reference*.

Querying Statistics from the IMSplex Using CSLZQRY

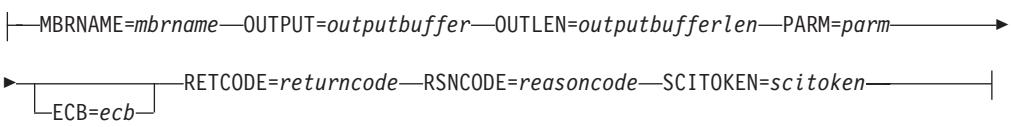
In an IMSplex, you might want to query statistics about one or more components in the CSL. You can write an IMSplex member program, for example, an automated operations program (AOP), that uses the CSLZQRY request to obtain statistics. Any member of an IMSplex can issue the CSLZQRY request.

CSLZQRY: Query Request

Use `FUNC=DSECT` to include equate (EQU) statements in your program for the CSLZQRY parameter list length and the CSLZQRY return and reason codes.

Use `FUNC=STATS` to request statistics from OM, RM, or SCI. The information that is returned from the CSLZQRY request is the same information that is passed to the STATS exit for that particular OM, RM, or SCI.

A:



CSLZQRY Request Parameters

The parameters for the CSLZQRY request are described in this topic.

ECB=*symbol*
ECB=*(r2-r12)*
(Optional) - Specifies a z/OS event control block (ECB) used for asynchronous requests. When the request is complete, the ECB specified is posted. If an ECB is not specified, the task is suspended until the request is complete. If an ECB

is specified, the invoker of the request must issue a WAIT (or equivalent) after receiving control from CSLZQRY and before using or examining any data returned by this request (including the RETCODE and RSNCODE fields).

MBRNAME=*symbol*
MBRNAME=*(r2-r12)*
(Required) - A four-byte input parameter that specifies the address of the eight-byte CSL member name to which to send the query.

OUTLEN=*symbol*
OUTLEN=*(r2-r12)*
(Required) - A four-byte output parameter that is used to receive the length of the output buffer. When the request returns, this word contains the length of the buffer pointed to by the OUTPUT= parameter. The output length is zero if no output is built, for example, when an error is detected before any output can be built. When the caller is done with this storage, it is the caller's responsibility to release the storage by issuing a CSLSCBFR request. See "CSLSCBFR: Buffer Return Request" on page 171 for more information.

OUTPUT=*symbol*
OUTPUT=*(r2-r12)*
(Required) - Specifies a 4-byte field to receive the address of the variable length output returned by the CSLZQRY request. The output contains the results of the CSLZQRY. The output length is returned in the OUTLEN= field. The output address is zero if no output was built, for example, if an error was detected before any output could be built. This buffer is not preallocated by the caller. When the caller is done with this storage, it is the caller's responsibility to release the storage by issuing a CSLSCBFR request. See "CSLSCBFR: Buffer Return Request" on page 171 for more information.

PARM=*symbol*
PARM=*(r2-r12)*
(Required) - Specifies the CSLZQRY parameter list. The length of the parameter list must be equal to the parameter list length EQU value defined by ZQRY_PARMLN.

RETCODE=*symbol*
RETCODE=*(r2-r12)*
(Required) - Specifies a 4-byte field to receive the return code on output. This can be returned by OM, RM, or SCI. OM return codes are defined in CSLORR. RM return codes are defined in CSLRRR. SCI return codes are defined in CSLSRR.

RSNCODE=*symbol*
RSNCODE=*(r2-r12)*
(Required) - Specifies a 4-byte field to receive the reason code on output. This can be returned by OM, RM, or SCI. OM reason codes are defined in CSLORR. RM reason codes are defined in CSLRRR. SCI reason codes are defined in CSLSRR.

SCITOKEN=*symbol*
SCITOKEN=*(r2-r12)*
(Required) - Specifies a 16-byte field containing the SCI token. This token uniquely identifies this connection to SCI. The SCI token was returned by a successful CSLSCREG FUNC=REGISTER request.

Shutting Down the CSL

A CSL is comprised of multiple components. Accordingly, you can shut down either an entire CSL or individual CSL manager address spaces. You can shut down:

- A single CSL component such as an OM, RM, or SCI
- A CSL, including all of its components, on a single z/OS image
- A CSL, including all of its components, that spans an IMSplex across multiple z/OS images

Before you shut down the CSL or any of its components, consider the following:

- You do not need to shut down other IMSplex members when you shut down CSL managers (SCI, OM, or RM) for maintenance.
- If you shut down an OM or RM for maintenance and there are other OMs or RMs active in the IMSplex, IMSplex members can still participate in IMSplex activities. However, an IMSplex member can communicate only with the SCI with which it is registered. If that SCI is shut down, any IMSplex members on the same z/OS image as that SCI cannot communicate with other IMSplex members until that SCI is restarted.

You can shut down the CSL by issuing:

1. The CSLZSHUT request, described in "CSLZSHUT: Shut Down Request."
2. A z/OS STOP command to individual CSL manager address spaces.
3. A CSL SHUTDOWN command using the z/OS MODIFY COMMAND interface.

Sample syntax for the z/OS commands is described in "Shutting Down the CSL Using z/OS Commands" on page 28.

CSLZSHUT: Shut Down Request

CSLZSHUT is a programming interface. It allows you to shut down one or more CSL address spaces from within an assembler program. With the CSLZSHUT request, you can terminate:

- A single CSL manager (OM, RM, or SCI)
- A CSL and all of its components on a single z/OS image
- A CSL and all of its components for an IMSplex across multiple z/OS images

The CSLZSHUT request is sent as a message, so control returns to the program that issued the request after the request is sent.

To shut down a single CSL component, send the `CSLZSHUT FUNC=QUIESCE,SCOPE=CSLMEMBER` message to the component you want to shut down.

To shut down a CSL and all of its components on a single z/OS image, either:

- Send a `CSLZSHUT FUNC=QUIESCE,SCOPE=CSLLOCAL` message to the SCI that is active on the z/OS image that contains the CSL to be shut down.
- Send a `CSLZSHUT FUNC=QUIESCE,SCOPE=CSLLOCAL,OSNAME=xxxx` message to any SCI active in the IMSplex (where `xxxx` is the z/OS image where the CSL to be shut down is active). SCI sends a CSLZSHUT request to all of the CSL components to be shut down.

To shut down the CSL on an entire IMSplex, send a `CSLZSHUT FUNC=QUIESCE,SCOPE=CSLPLEX` message to any SCI active in the IMSplex. SCI sends a CSLZSHUT request to all the CSL components in the IMSplex.

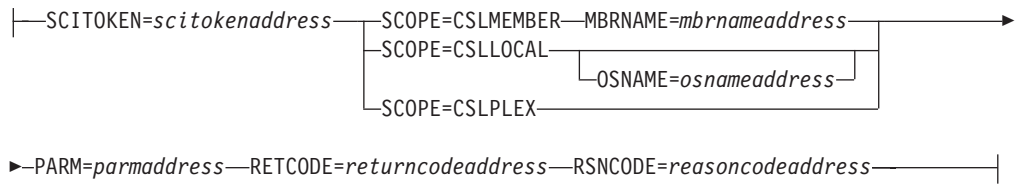
The parameters for the CSLZSHUT request are described in "The CSLZSHUT Request Parameters."

Format of the CSLZSHUT Request

CSLZSHUT DSECT Syntax: Use `FUNC=DSECT` to include equate (EQU) statements in your program for the CSLZSHUT parameter list length and the CSLZSHUT return and reason codes.

CSLZSHUT QUIESCE Syntax: Use `FUNC=QUIESCE` to request that a CSL component shut down normally. Any work that the CSL component is currently processing is completed, and then the component shuts down. After processing the request, that component will not accept any new work.

A:



If the component that is being shut down is an SCI, the IMS plex members that are currently registered with that SCI are not deregistered before SCI terminates. This can impact event notification. These IMSplex members cannot communicate with other IMSplex members because their SCI is shut down. If one or more of the "orphaned" members is shut down or fails, the other IMSplex members are not notified of the shutdown or failure event until SCI comes back online.

Notification of the shutdown or failure depends on the authorization level of the members. If the terminating member is non-authorized, other members are notified when SCI restarts. If the terminating member is authorized, other authorized members, including orphaned authorized members, are notified before SCI restarts.

The CSLZSHUT Request Parameters

The following is a description of the CSLZSHUT request parameters.

MBRNAME=*symbol*
MBRNAME=*(r2-r12)*
(Required if SCOPE=CSLMEMBER) - Specifies the eight-byte CSL member name to which to send the shutdown request.

OSNAME=*symbol*
OSNAME=*(r2-r12)*
(Required if SCOPE=CSLLOCAL) - Specifies the eight-byte name of the CSL, running on the z/OS image, that is to be shut down. If the OSNAME parameter is specified and the SCI is not active on the z/OS image specified, the command will not be processed.

PARM=*symbol*

PARM=*(r1-r12)*
(Required) - Specifies the CSLZSHUT parameter list. The length of the parameter list must be equal to the parameter list length EQU value defined by ZSHUT_PARMLN.

RETCODE=*symbol*
RETCODE=*(r2-r12)*
(Required) - Specifies a 4-byte field to receive the return code on output. SCI return codes are defined in CSLSRR.

RSNCODE=*symbol*
RSNCODE=*(r2-r12)*
(Required) - Specifies a 4-byte field to receive the reason code on output. SCI reason codes are defined in CSLSRR.

SCITOKEN=*symbol*
SCITOKEN=*(r2-r12)*
(Required) - Specifies a 16-byte field containing the SCI token. This token uniquely identifies this connection to SCI. The SCI token was returned by a successful CSLSCREG FUNC=REGISTER request.

SCOPE=CSLMEMBER | CSLLOCAL | CSLPLEX
(Required) - Specifies the scope of the CSL termination. Valid values for the SCOPE parameter are:

CSLMEMBER
This requests the CSL component receiving the request to shut itself down. CSLMEMBER can be processed by any CSL component.

CSLLOCAL
This requests that the CSL components on a single z/OS image be shut down. If the OSNAME parameter is also specified, the CSL components on that particular z/OS image are shut down. If the OSNAME parameter is specified *and* the SCI is not active on the z/OS image specified, the command will not be processed. If the OSNAME parameter is not specified, the SCI receiving the message shuts down the CSL on the local z/OS image. Only an SCI can process a SCOPE=CSLLOCAL request. If this request is sent to other CSL components, it is ignored.

CSLPLEX
This requests that the CSL components in an entire IMSplex be shut down. Only an SCI can process a SCOPE=CSLPLEX request. If this request is sent to other CSL components, it is ignored.

Shutting Down the CSL Using z/OS Commands

You can shut down the CSL as one unit by issuing the CSL SHUTDOWN command to any SCI in the IMSplex with the z/OS MODIFY command interface. You can stop individual modular units in the IMSplex by issuing the z/OS STOP command to the address space you want to stop.

Recommendation: Use the CSL SHUTDOWN command with the z/OS MODIFY command interface to shut down the CSL, rather than stopping individual components.

To shut down a CSL on one z/OS image, issue the z/OS MODIFY command as follows:

```
F scijobname,SHUTDOWN CSLLCL
```

This command shuts down the CSL on the z/OS image associated with the SCI that receives the command. Use this version of the command to shut down the CSL on a single z/OS image in an orderly way.

To shut down an entire IMSplex, issue the z/OS MODIFY command as follows:

```
F scijobname,SHUTDOWN CSLPLEX
```

This command shuts down the CSL managers on all z/OS images in a single IMSplex associated with the SCI that receives the command.

Note: To shut down the CSL managers using the SHUTDOWN CSLPLEX command, a local SCI is required. If you issue the SHUTDOWN CSLPLEX command on a system without an active SCI, the CSL managers will not shut down.

In each of these examples, `scijobname` is the name of the SCI in the CSL. After it receives the command, SCI notifies other CSL managers (OMs and RMs) to stop, and then SCI stops. If clients are currently connected to any CSL manager and were not first stopped with a `/CHE FREEZE` or other command, message CSL0300I is issued, work is quiesced, and then the CSL manager stops.

Using the z/OS Automatic Restart Manager with the CSL

A CSL address space (OM, RM, SCI), if requested, can register with the z/OS Automatic Restart Manager (ARM). The ARM is a z/OS recovery function that can improve the availability of started tasks. When a task fails or the system on which it is running fails, the ARM can restart the task without operator intervention.

IBM provides policy defaults for automatic restart management. You can use these defaults, or you can define your own ARM policy to specify how CSL address spaces should be restarted. The ARM policy specifies what should be done if the system fails or if a CSL address space fails.

To enable the ARM, you can specify `ARMRST=Y` in one of two ways:

- In the CSL address space initialization PROCLIB member
  - CSLSIxxx for SCI
  - CSLRIxxx for RM
  - CSLOIxxx for OM
- As an execution parameter

When ARM is enabled, the CSL address spaces register to ARM with an ARM element name, which are defined in Table 7.

Table 7. ARM element names

CSL Address Space	ARM element name
OM	“CSL” + omname + “OM”
RM	“CSL” + rmname + “RM”
SCI	“CSL” + sciname + “SCI”

Note: The name of the CSL address space is the name defined either as an execution parameter, or in the initialization PROCLIB member of that CSL address space. For example, if OMNAME=OM1A in the CSLOIxxx PROCLIB member, the ARM element name is CSLOM1AOM.

Use the appropriate ARM element name in your ARM policy for each CSL address space. For more information, see *z/OS MVS: Setting Up a Sysplex*.

An abend table exists in the module for each CSL address space:

- CSLOARM0 for OM
- CSLRARM0 for RM
- CSLSARM0 for SCI

The table lists the abends for which the ARM does not restart the CSL address space after the abend occurs. You can modify this table.

Chapter 3. CSL Operations Manager

This topic describes the OM component of the CSL:
- "Overview of the CSL Operations Manager"
- "CSL OM Definition and Tailoring"
- "CSL OM Administration" on page 37
- "CSL OM User Exit Routines" on page 40
- "CSL OM Automated Operator Program Clients" on page 91
- "CSL OM Command Processing Client Requests" on page 78
- "CSL OM Directives" on page 94

Overview of the CSL Operations Manager

OM controls the operations of an IMSplex. OM provides an application programming interface (the OM API) through which commands can be issued and responses received. With a single point of control (SPOC) interface, you can submit commands to OM. The SPOC interfaces include the TSO SPOC, the REXX SPOC API, and the IMS Control Center. You can also write your own application to submit commands.

OM:
- Routes IMS commands to IMSplex members registered for the command.
- Consolidates command responses from individual IMSplex members into a single response and provides that response to the originator of the command.
- Provides an API for automated operator commands.
- Provides a general use interface to register commands to support any command processing client.
- Provides user exits for command and response edit and command security.

One OM must be defined in the IMSplex to use OM functions. Each z/OS image can have more than one OM. If multiple OMs are defined in the IMSplex, any OM defined can perform work from any z/OS image in the IMSplex.

CSL OM Definition and Tailoring

This topic describes the system definition and tailoring considerations for an OM in the CSL. For information on IMS system definition and tailoring, see *IMS Version 9: Installation Volume 2: System Definition and Tailoring*.

This following topics provide additional information:
- "CSL OM Startup Procedure"
- "CSL OM Execution Parameters" on page 32
- "BPE Considerations for the CSL OM" on page 34
- "CSL OM Initialization Parameters PROCLIB Member" on page 35

CSL OM Startup Procedure

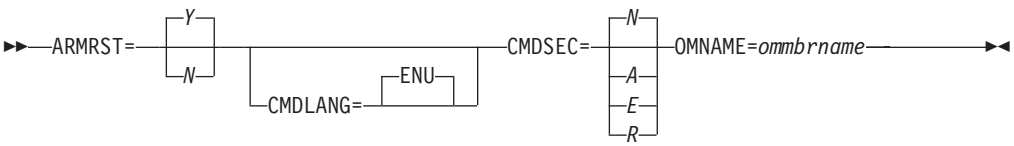
You can start OM as a started procedure or with JCL. A sample startup procedure, shown in Figure 12 on page 32, is called CSLOM and can be found in IMS.PROCLIB.

```
//*****************************************************************
//*     OM Procedure
//*
//*
//*     Parameters:
//*     BPECFG  - Name of BPE member
//*     OMINIT  - Suffix for your CSLOIxxx member
//*     PARM1   - other override parameters:
//*               ARMRST  - Indicates if ARM should be used
//*               CMDLANG - Language for command description text
//*               CMDSEC  - Command security method
//*               OMNAME  - Name of OM being started
//*
//*               example:
//*               PARM1='ARMRST=Y,CMDSEC=R,OMNAME=OM1,CMDLANG=ENU'
//*
//*************************************************************@SCPYRT**
//*                                                                   *
//*       Licensed Materials - Property of IBM                        *
//*                                                                   *
//*       "Restricted Materials of IBM"                               *
//*                                                                   *
//*       5655-J38 (C) Copyright IBM Corp. 2003                       *
//*                                                                   *
//*************************************************************@ECPYRT**
//*
//CSLOM    PROC RGN=3000K,SOUT=A,
//             RESLIB='IMS.SDFSRESL',
//             BPECFG=BPECONFG,
//             OMINIT=000,
//             PARM1=
//*
//OMPROC   EXEC PGM=BPEINI00,REGION=&RGN,
//  PARM='BPECFG=&BPECFG,BPEINIT=CSLOINI0,OMINIT=&OMINIT,&PARM1'
//*
//STEPLIB  DD  DSN=&RESLIB,DISP=SHR
//         DD  DSN=SYS1.CSSLIB,DISP=SHR
//PROCLIB  DD  DSN=IMS.PROCLIB,DISP=SHR
//SYSPRINT DD  SYSOUT=&SOUT
//SYSUDUMP DD  SYSOUT=&SOUT
//*
```

Figure 12. Sample OM Startup Procedure

CSL OM Execution Parameters

The parameters that you can specify as execution parameters on the startup procedure for OM follow. Some parameters that are required for OM initialization can also be specified in the initialization parameters PROCLIB member (see “CSL OM Initialization Parameters PROCLIB Member” on page 35).



ARMRST= Y | N
Specifies whether the z/OS Automatic Restart Manager (ARM) is to be used to restart the OM address space after an abend. If you specify **Y** (yes), ARM

restarts the OM address space after most system failures. If you specify **N** (no), ARM does not restart the OM address space after any system failure.

ARM does not restart the OM address space if OM abends before restart is complete.

This is an optional execution parameter. If specified, it overrides the value specified in the CSLOIxxx PROCLIB member. For more information on ARM, see “Using the z/OS Automatic Restart Manager with the CSL” on page 29.

BPECFG=
Specifies an 8-character name for the BPE configuration parameters PROCLIB member. This parameter can only be specified as an execution parameter. If a PROCLIB member is not specified, BPE uses default values for all parameters. This parameter is optional. If it is not specified, the BPE defaults are no user exits, a trace level of error, and US English as the language.

BPEINIT=CSLOINI0
Specifies the name of the module that contains OM start up values required by BPEINI00 to start an OM address space. For OM, this value must be CSLOINI0. This required parameter can only be specified as an execution parameter.

CMDLANG=ENU
Specifies the language to be used for IMS command text that is distributed to OM automation clients upon request. This affects only the command descriptions that are displayed on a workstation SPOC that requests command text from OM. This value defaults to ENU for US English.

The value is not validated at OM initialization time. It is validated only when a `CSLOMQRY QUERY TYPE(CMDSYNTAX)` request is issued. OM attempts to read a partitioned data set (PDS) member in the data set specified by the CMDTEXTDSN= with a member name of “CSLOT” concatenated with the 3 character CMDLANG= value. This is the member that contains the command syntax translatable text. The CMDLANG= value can be overridden on the CSLOMQRY request.

This is an optional execution parameter. If specified, it overrides the value specified in the CSLOIxxx PROCLIB member.

CMDSEC=
Specifies the security method to be used for OM command security.

A Specifies that both RACF® (or an equivalent security product) and the OM Security exit are to be called (options E and R). RACF is called first. Then the security authorization facility (SAF) return code, RACF return code, and RACF reason code are passed to the exit. These return codes are decoded into a security code, which is also passed to the exit for processing.

E Specifies that the OM Security user exit routine is to be called for command authorization.

N Specifies that no authorization checking is to be done. This is the default.

R Specifies that RACF (or an equivalent security product) is to be called for command authorization.

This is an optional execution parameter. If specified, it overrides the value specified in the CSLOIxxx PROCLIB member. If not specified, the value in the CSLOIxxx PROCLIB member is used.

OMINIT=000
Specifies a 3-character suffix for the OM initialization parameters PROCLIB member, CSLOIxxx. This parameter can only be specified as an execution parameter. The default suffix is 000.

OMNAME=*ommbrname*
Specifies the name for the OM address space. This is an optional 1-6 character name. If specified, it overrides the value specified in the CSLOIxxx PROCLIB member. You must specify this parameter either as an execution parameter or in the CSLOIxxx PROCLIB member. This name is used to create the OMID, which is used in OM processing. The 8-character OMID is the OMNAME followed by the characters ″OM″. Trailing blanks in the OMNAME are deleted, and the OMID is padded with blanks. For example, if OMNAME=ABC then OMID=″ABCOM ″.

BPE Considerations for the CSL OM

Use the OM BPE user exit list PROCLIB member to define OM user exits to BPE. The member is the PROCLIB member specified by the EXITMBR= parameter in the BPE configuration parameter PROCLIB member.

Use the user exit list PROCLIB member to specify the modules to be called for specific exit types. Each user exit type can have one or more exit modules associated with it. Use the EXITDEF statement to define the user exit modules to be called for a given exit type.

The BPE user exit PROCLIB member and BPE configuration PROCLIB member are described in *IMS Version 9: Base Primitive Environment Guide and Reference*.

A sample OM BPE user exit list PROCLIB member is shown in Figure 13 on page 35.

```
*********************************************************************
* OM USER EXIT LIST PROCLIB MEMBER                                  *
*********************************************************************

#-------------------------------------------------------------------#
# DEFINE 1 OM CLIENT CONNECTION USER EXIT: ZOCLNCN0                 #
#-------------------------------------------------------------------#
EXITDEF(TYPE=CLNTCONN,EXITS=(ZOCLNCN0),COMP=OM)

#-------------------------------------------------------------------#
# DEFINE 1 OM INIT/TERM USER EXIT: ZOINTM00                         #
#-------------------------------------------------------------------#
EXITDEF(TYPE=INITTERM,EXITS=(ZOINTM00),COMP=OM)

#-------------------------------------------------------------------#
# DEFINE 1 OM INPUT USER EXIT: ZINPUT00                             #
# WITH AN ABEND LIMIT OF 8.                                         #
#-------------------------------------------------------------------#
EXITDEF(TYPE=INPUT,EXITS=(ZINPUT00),ABLIM=8,COMP=OM)

#-------------------------------------------------------------------#
# DEFINE 1 OM OUTPUT USER EXIT: ZOUTPUT0                            #
#-------------------------------------------------------------------#
EXITDEF(TYPE=OUTPUT,EXITS=(ZOUTPUT0),COMP=OM)

#-------------------------------------------------------------------#
# DEFINE 1 OM SECURITY USER EXIT: ZSECURE0                          #
#-------------------------------------------------------------------#
EXITDEF(TYPE=SECURITY,EXITS=(ZSECURE0),COMP=OM)
```

Figure 13. OM User Exit List PROCLIB Member

CSL OM Initialization Parameters PROCLIB Member

Use the CSLOIxxx PROCLIB member to specify parameters that initialize the OM address space. Certain parameters within CSLOIxxx can be overridden with the OM execution parameters.

A CSLOIxxx member consists of one or more fixed-length character records (the configuration data set can be of any LRECL greater than eight, but it must be fixed record format). The rightmost-eight columns are ignored but can be used for sequence numbers or any other notation. Keyword parameters can be coded in the remaining columns in free format, and can contain leading and trailing blanks. You can specify multiple keywords in each record; use commas or spaces to delimit keywords. Statements that begin with a "*" or "#" in column 1 are comment lines and are ignored. Additionally, comments can be included anywhere within a statement by enclosing them between " /* "and "*/", for example, `/* PROCLIB comments */`. Values coded in this PROCLIB member are case-sensitive. In general, you should use upper case for all parameters.

ARMRST= Y | N

Specifies whether the z/OS Automatic Restart Manager (ARM) is to be used to restart the OM address space after an abend. If you specify **Y** (yes), ARM restarts the OM address space after most system failures. If you specify **N** (no), ARM does not restart the OM address space after any system failure.

ARM does not restart the OM address space if OM abends before restart is complete. For more information on ARM, see "Using the z/OS Automatic Restart Manager with the CSL" on page 29.

CMDLANG= ENU

The language to be used for IMS command text that is distributed to OM

automation clients upon request. This affects only the command descriptions that are displayed on a workstation SPOC that requests command text from OM. This value defaults to ENU for US English.

The value is not validated at OM initialization time. It is only validated when a `CSLOMQRY QUERY TYPE(CMDSYNTAX)` request is issued. OM attempts to read a PDS member in the data set specified by the CMDTEXTDSN= with a member name of "CSLOT" concatenated with the 3 character CMDLANG= value. This is the member that contains the command syntax translatable text. The CMDLANG= value can be overridden on the CSLOMQRY request.

This parameter can be specified as an execution parameter on the OM procedure to override the value in CSLOIxxx.

CMDSEC=
Specifies the security method to be used for OM command security.

A Specifies that both RACF (or an equivalent security product) and the exit are to be called (options E and R). RACF is called first. Then the security authorization facility (SAF) return code, RACF return code, and RACF reason code are passed to the exit. These return codes are decoded into a security code, which is also passed to the exit for processing.

E Specifies that the OM Security user exit routine is to be called for command authorization.

N Specifies that no authorization checking is to be done. This is the default.

R Specifies that RACF (or an equivalent security product) is to be called for command authorization.

CMDTEXTDSN=
Specifies the data set name for the PDS that contains the command syntax translatable text. This keyword is required. The parameter value is the 1-44 character data set name. The data set must be a PDS with fixed length record members.

IMSPLEX()
Specifies definitions for an IMSplex managed by OM. IMSPLEX is a required parameter. There is no default. Only one IMSPLEX keyword can be specified. The IMSPLEX keyword must precede the left parenthesis. The IMSPLEX definition parameters follow:

NAME=
Specifies a 1-5 character identifier that specifies the IMSplex group name. OM concatenates this identifier to "CSL" to create the IMSplex group name. All IMSplex member address spaces that are in the same IMSplex group sharing either databases or message queues must specify the same identifier. The same identifier must also be used for the IMSPLEX= parameter in the CSLSIxxx, CSLRIxxx and DFSCGxxx PROCLIB members.

OMNAME=*ommbrname*
Specifies the name for the OM address space. This is an optional 1-6 character name. Specify this parameter either as an execution parameter or in the CSLOIxxx PROCLIB member. This name is used to create the OMID which is used in OM processing. The 8-character OMID is the OMNAME followed by the characters ″OM″. Trailing blanks in the OMNAME are deleted and the OMID is padded with blanks. For example, if OMNAME=ABC then OMID=″ABCOM ″.

A sample CSLOIxxx PROCLIB member is shown in Figure 14. The sample, called CSLOI000, is provided as part of IMS.PROCLIB.

```
*-------------------------------------------------------------------*
* Sample OM Initialization PROCLIB Member.
*-------------------------------------------------------------------*

ARMRST=Y,                    /* ARM should restart OM on failure  */
CMDLANG=ENU,                 /* Use English for Command Desc      */
CMDSEC=N,                    /* No Command Security               */
OMNAME=OM1,                  /* OM Name (OMID = OM1OM)            */
IMSPLEX(NAME=PLEX1)          /* IMSplex Name (CSLPLEX1)           */
CMDTEXTDSN=IMSTESTG.DUMMY.TRNTBL  /* CMD Syntax Translation Table */

*-------------------------------------------------------------------*
* End of Member CSLOI000                                            *
*-------------------------------------------------------------------*
```

Figure 14. CSLOIxxx PROCLIB member

CSL OM Administration

This topic describes the administrative tasks associated with OM:

- "Starting or Restarting the CSL OM"
- "Registering Command Processing Clients in a CSL"
- "Shutting Down the CSL OM" on page 38
- "Command Processing Considerations in a CSL OM" on page 38

Starting or Restarting the CSL OM

You start OM by issuing a z/OS START command to run the OM procedure. To start an OM address space with a started procedure, issue the z/OS `START` command as follows:

```
S omjobname
```

In this example, `omjobname` is the job name of the OM address space to be started. For more information on the initialization parameters, see "CSL OM Execution Parameters" on page 32. You can also provide JCL to run the OM procedure.

After OM is started, if it is abnormally terminated, it can be restarted using the z/OS Automatic Restart Manager (ARM). OM must complete initialization for ARM to restart the address space if an abend occurs. Use of ARM to restart OM is the default.

Registering Command Processing Clients in a CSL

Before a command processing client, such as IMS, can process commands from an OM, it must first register its commands with an OM. By registering, it identifies the commands that it intends to process. You can register a command processing client by using CSLOMBLD to build a list of commands to be registered and the CSLOMREG request to send the list to an OM.

After the command processing client is registered, it must notify OM that it is ready to process commands before OM sends any commands. It does this by issuing the CSLOMRDY request to OM.

When the command processing client wants to stop processing commands from an OM, it issues a CSLOMDRG request. This request deregisters the client from OM and prevents further command processing.

For more information on these requests, see:
- "CSLOMREG: Command Registration Request" on page 85
- "CSLOMRDY: Ready Request" on page 84
- "CSLOMDRG: Command Deregistration Request" on page 81

Shutting Down the CSL OM

Recommendation: Although you can shut down OM by itself, IBM recommends that you shut down OM by shutting down the CSL as one unit. For information about shutting down the CSL, see "Shutting Down the CSL" on page 26.

To shut down OM by itself, issue one of the following:
- The CSLZSHUT request, described in "CSLZSHUT: Shut Down Request" on page 26
- The z/OS STOP command:

```
P omjobname
```

In this example, `omjobname` is the job name of the OM address space to stop. If no clients are connected to OM, OM shuts down. If clients are connected to OM, message CSL0300I is issued, and OM quiesces in-flight work. After all work is quiesced, the OM address space terminates.

Before shutting down an OM, consider the reasons for shutting down and how shutting down OM can impact other IMSplex members. For more information, see "Shutting Down the CSL" on page 26.

Command Processing Considerations in a CSL OM

In an IMSplex environment, commands issued to OM can behave differently than commands issued to a single IMS system. This topic describes some of these behavioral and usage differences in specific configurations, as well as OM command security considerations.

CSL OM Command Routing

Commands that are issued to OM are, by default, routed to all IMSs in the IMSplex that are active and registered to process that particular command. If you want to route a command to one or more specific command processing clients in the IMSplex, use the ROUTE parameter on the CSLOMCMD request or the CSLOMI API. You can also specify routing information with the TSO SPOC and IMS control center. For information on how to specify the ROUTE parameter, see "CSLOMI: API Request" on page 63 and "CSLOMCMD: Command Request" on page 55. Refer also to the online help provided with the TSO SPOC and IMS Control Center.

For information about installing and using the IMS Control Center, go to the DB2 Information Management Software Information Center for z/OS Solutions on the Web at http://publib.boulder.ibm.com/infocenter/dzichelp/ and click IMS Version 9 in the Contents pane, then select IMS Control Center.

In an IMSplex, a command comes from a SPOC or AOP to OM. It is then routed to IMS systems; each IMS system returns its command output. OM then consolidates the individual responses and sends consolidated output, encapsulated in XML tags, back to the client originating the request. This routing is shown in Figure 15 on page 39 OS1 has an SCI, a SPOC TSO/ISPF application, an automation program.

OS2 has an OM with an SCI, and IMS control region with an SCI. OS3 has two IMS control regions, each with an SCI. The command is routed from OS1's SCI to OM in OS2. The SCI that is associated with that OM routes the command to other IMS systems through those system's SCI.

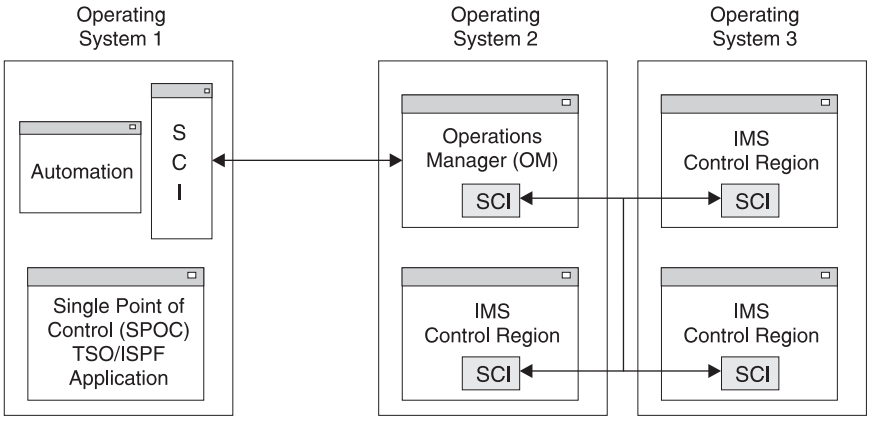


Figure 15. Command Routing in an IMSplex with CSL

CSL OM Command Responses

When commands are issued to an OM, command responses are encapsulated in XML tags. More information on this XML output is in "CSL OM XML Output" on page 93.

For comprehensive information about IMS commands and their responses, see *IMS Version 9: Command Reference*.

CSL OM Command Security

OM command security is optionally performed during command processing. Command security allows:

- The user to control which user IDs can enter IMS commands through OM
- The user ID to be associated with an application program address space
- The user ID to be the end user logged onto TSO SPOC

The CMDSEC= parameter is available on the OM startup procedure (CSLOM), the OM initialization PROCLIB member (CSLOIxxx), and the DFSCGxxx PROCLIB member. When it is issued as part of the OM startup procedure, it applies to all IMS commands, type-1 and type-2. When it is issued using the DFSCGxxx PROCLIB member, it applies only to type-1 commands entered through OM. The differences in OM and IMS security are described in Table 8.

Table 8. Comparing OM and IMS Security

Security Method	N	A	E	R
OM Execution Parameter (CSLOM and CSLOIxxx)	No authorization checking is performed. This is the default.	Calls both RACF and the OM Command Security Exit routine for command authorization.	Calls the OM Command Security Exit routine for command authorization.	Calls RACF for command authorization. Commands are part of the OPERCMDS resource class.

Table 8. Comparing OM and IMS Security (continued)

Security Method	N	A	E	R
DFSCGxxx PROCLIB member	No authorization checking is performed. This is the default. OM might perform command authorization.	Calls both RACF and the IMS Command Authorization Exit routine (DFSCCMD0) for command authorization.	Calls the IMS Command Authorization Exit routine (DFSCCMD0) for command authorization.	Calls RACF for command authorization. Commands are part of the CIMS resource class.

Recommendation: Use OM command security rather than IMS command security.

RACF access authorities (READ or UPDATE) and resource names for all commands supported through the OM API are described in *IMS Version 9: Command Reference*. The RACF authorities indicate the access authority with which the command was registered.

Commands are registered to OM with the CSLOMBLD request. The access authority on the RACF PERMIT command must match the access authority with which the command was registered.

For more information on registering commands with CSLOMBLD, see "CSLOMBLD: Command Registration Build" on page 79.

CSL OM User Exit Routines

You can write OM user exits to customize and monitor the OM environment. No sample exits are provided.

OM uses BPE services to call and manage its user exits. BPE enables you to externally specify the user exit modules to be called for a particular user exit type by using EXITDEF= statements in the BPE user exit list PROCLIB members. BPE also provides a common user exit runtime environment for all user exits. This environment includes a standard user exit parameter list, callable services, static and dynamic work areas for the exits, and a recovery environment for user exit abends. For more information about the BPE user exit interface, see *IMS Version 9: Base Primitive Environment Guide and Reference*.

CSL OM Client Connection User Exit

This exit is called when a client registers or deregisters commands with OM. This exit is optional.

This exit is called for the following event:

- A client issues the CSLOMRDY request to indicate that the client is ready to accept commands for processing.

This exit is defined as TYPE=CLNTCONN in the EXITDEF statement in the BPE user exit list PROCLIB member. You can specify one or more user exits of this type. When this exit is invoked, all user exits of this type are driven in the order specified by the EXITS= keyword. For more information on how to define user exit module names, see the OM BPE user exit list PROCLIB member topic in *IMS Version 9: Base Primitive Environment Guide and Reference*.

This exit is invoked amode 31 and should be reentrant.

Contents of Registers on Entry

Register	Contents
1	Address of BPE user exit parameter list (mapped by macro BPEUXPL).
13	Address of the first of 2 prechained 72-byte save areas. These save areas are chained according to standard z/OS save area linkage convention. The first save area can be used by the exit to save registers on entry. The second save area is for use by routines called from the user exit.
14	Return address.
15	Entry point of exit routine.

On entry to the Client Connection exit, register 1 points to a standard BPE user exit parameter list. Field UXPL_EXITPLP in this list contains the address of the OM Client Connection user exit parameter list, which is mapped by the CSLOCLX macro. Field UXPL_COMPTYPEP in this list points to the character string “OM,” indicating an OM address space.

OM Client Connection User Exit Parameter List--Client Connect: Table 9 lists the user exit parameter list for OM Client Connection. Included are the field name, the offset value and length, both in hexadecimal, how the field is used, and a brief description of the field.

Table 9. OM Client Connection User Exit Parameter List--Client Connect

Field Name	Offset	Length	Field Usage	Description
OCLX_PVER	X’00’	X’04’	Input	Parameter list version number (00000001).
OCLX_FUNC	X’04’	X’04’	Input	Function code: **3** Client ready to process commands.
OCLX_MBRNAME	X’08’	X’08’	Input	Client (IMSplex member) name.
OCLX_MBRTYPE	X’10’	X’02’	Input	IMSplex member type (mapped by CSLSTPIX).
	X’12’	X’02’	None	Reserved.
OCLX_MBRSTYPE	X’14’	X’08’	Input	IMSplex member subtype.
	X’1C’	X’04’	None	Reserved.

OM Client Connection User Exit Parameter List--Client Disconnect: Table 10 lists the user exit parameter list for OM Client Disconnect. Included are the offset value and length, both in hexadecimal, how the field is used, and a brief description of the field.

Table 10. OM Client Connection User Exit Parameter List--Client Disconnect

Offset	Length	Field Usage	Description
X’00’	X’04’	Input	Parameter list version number (00000001).
X’04’	X’04’	Input	Function code: **2** Client no longer processing commands.
X’08’	X’08’	Input	Client (IMSplex member) name.
X’10’	X’02’	Input	IMSplex member type (mapped by CSLSTPIX).

Table 10. OM Client Connection User Exit Parameter List--Client Disconnect (continued)

Offset	Length	Field Usage	Description
X'12'	X'01'	Input	Flag byte indicates whether the client disconnect is normal or abnormal. **X'80'** Client disconnect is abnormal.
X'13'	X'01'	None	Reserved.
X'14'	X'08'	Input	IMSplex member subtype.
X'1C'	X'04'	None	Reserved.

Contents of Registers on Exit

Register	Contents	
15	**Return Code**	**Meaning**
	0	Always zero

All other registers must be restored.

CSL OM Initialization/Termination User Exit

This exit enables you to initialize or terminate work areas or control blocks specific to a user-written SPOC application. This exit is not called during OM address space abnormal termination or IMSplex abnormal termination. This exit is optional.

This exit is called for the following events:

- After OM has completed initialization
- After each IMSplex has initialized
- When OM is normally terminating
- When an IMSplex is normally terminating

This exit is defined as TYPE=INITTERM in the EXITDEF statement in the BPE user exit list PROCLIB member. You can specify one or more user exits of this type. When this exit is invoked, all user exits of this type are driven in the order specified by the EXITS= keyword. For more information on how to define user exit module names, see the OM BPE user exit list PROCLIB member topic in *IMS Version 9: Base Primitive Environment Guide and Reference*.

This exit is invoked amode 31 and should be reentrant.

Contents of Registers on Entry

Register	Contents
1	Address of BPE user exit parameter list (mapped by macro BPEUXPL).
13	Address of the first of 2 prechained 72-byte save areas. These save areas are chained according to standard z/OS save area linkage convention. The first save area can be used by the exit to save registers on entry. The second save area is for use by routines called from the user exit.
14	Return address.
15	Entry point of exit routine.

On entry to the Initialization/Termination exit, register 1 points to a standard BPE user exit parameter list. Field UXPL_EXITPLP in this list contains the address of the

OM Initialization/Termination user exit parameter list, which is mapped by macro CSLOITX. Field UXPL_COMPTYPEP in this list points to the character string “OM,” indicating an OM address space.

OM Init/Term User Exit Parameter List--OM Initialization: Table 11 lists the user exit parameter list for OM Initialization. Included are the field name, the offset value and length, both in hexadecimal, how the field is used, and a brief description of the field.

Table 11. OM Init/Term User Exit Parameter List--OM Initialization

Field Name	Offset	Length	Field Usage	Description
OITX_PVER	X’00’	X’04’	Input	Parameter list version number (00000001).
OITX_FINIT	X’04’	X’04’	Input	Function code **1** OM initialization.

OM Init/Term User Exit Parameter List--OM Termination: Table 12 lists the user exit parameter list for OM Termination. Included are the field name, the offset value and length, both in hexadecimal, how the field is used, and a brief description of the field.

Table 12. OM Init/Term User Exit Parameter List--OM Termination

Field Name	Offset	Length	Field Usage	Description
OITX_PVER	X’00’	X’04’	Input	Parameter list version number (00000001).
OITX_FTERM	X’04’	X’04’	Input	Function code **1** OM normal termination.

OM Init/Term User Exit Parameter List--IMSplex Initialization: Table 13 lists the user exit parameter list for IMSplex initialization. Included are the offset value and length, both in hexadecimal, how the field is used, and a brief description of the field.

Table 13. OM Init/Term User Exit Parameter List--IMSplex Initialization

Field Name	Offset	Length	Field Usage	Description
OITX_PVER	X’00’	X’04’	Input	Parameter list version number (00000001).
OITX_FPLXINIT	X’04’	X’04’	Input	Function code **3** IMSplex normal initialization.
OITX_IPLEXNM	X’08’	X’08’	Input	IMSplex name.

OM Init/Term User Exit Parameter List--IMSplex Termination: Table 14 lists the user exit parameter list for IMSplex termination. Included are the field name, the offset value and length, both in hexadecimal, how the field is used, and a brief description of the field.

Table 14. OM Init/Term User Exit Parameter List--IMSplex Termination

Field Name	Offset	Length	Field Usage	Description
OITX_PVER	X’00’	X’04’	Input	Parameter list version number (00000001).
OITX_FPLXTERM	X’04’	X’04’	Input	Function code **4** IMSplex normal termination.

Table 14. OM Init/Term User Exit Parameter List--IMSplex Termination (continued)

Field Name	Offset	Length	Field Usage	Description
OITX_TPLEXNM	X'08'	X'08'	Input	IMSplex name.

Contents of Registers on Exit

Register	Contents	
15	**Return Code**	**Meaning**
	0	Always zero

All other registers must be restored.

CSL OM Input User Exit

This exit is called to allow a user to view and manipulate command input from an OM automation client. This exit is optional.

This exit is called for the following event:

- OM receives a command. This exit is called before OM processes the command, which allows the command to be modified or rejected.

This exit is defined as TYPE=INPUT in the EXITDEF statement in the BPE user exit list PROCLIB member. You can specify one or more user exits of this type. When this exit is invoked, all user exits of this type are driven in the order specified by the EXITS= keyword. For more information on how to define user exit module names, see the OM BPE user exit list PROCLIB Member topic in *IMS Version 9: Base Primitive Environment Guide and Reference*.

This exit is invoked amode 31 and should be reentrant.

Contents of Registers on Entry

Register	Contents
1	Address of BPE user exit parameter list (mapped by macro BPEUXPL).
13	Address of the first of 2 prechained 72-byte save areas. These save areas are chained according to standard z/OS save area linkage convention. The first save area can be used by the exit to save registers on entry. The second save area is for use by routines called from the user exit.
14	Return address.
15	Entry point of exit routine.

On entry to the Input exit, register 1 points to a standard BPE user exit parameter list. Field UXPL_EXITPLP in this list contains the address of the OM Input user exit parameter list, which is mapped by macro CSLOINX. Field UXPL_COMPTYPEP in this list points to the character string “OM,” indicating an OM address space.

OM Input User Exit Parameter List--Command Input: Table 15 on page 45 lists the user exit parameter list for command input. Included are the field name, the offset value and length, both in hexadecimal, how the field is used, and a brief description of the field.

Table 15. OM Input User Exit Parameter List--Command Input

Field Name	Offset	Length	Field Usage	Description
OINX_PVER	X'00'	X'04'	Input	Parameter list version number (00000001).
OINX_FUNC	X'04'	X'04'	Input	Function code **1** Command input.
OINX_MBRNAME	X'08'	X'08'	Input	Client (IMSplex member) where command originated.
OINX_MBRTYPE	X'10'	X'02'	Input	IMSplex member type where command originated.
OINX_CMDMOD	X'12'	X'01'	Output	Command input modified field. This field indicates that the exit modified the command input string and that the updated command input should be processed. • **4** Command input was modified by the exit. This is the only valid value. All other values are ignored.
	X'13'	X'01'	None	Reserved.
OINX_MBRSTYPE	X'14'	X'08'	Input	IMSplex member subtype where command originated.
OINX_USERID	X'1C'	X'08'	Input	user ID of application where the command originated.
OINX_INPUTLEN	X'24'	X'04'	Input	Length of the command input string. This length does not include 80 bytes for command expansion.
OINX_INPUTPTR	X'28'	X'04'	Input	Address of the command input string. The command input string is followed by 80 blanks that can be used by the exit to expand the command input.
OINX_INMODLEN	X'2C'	X'04'	Output	New length of command input string after being modified by the exit. The exit must set this field if it modifies the command input string. If the exit indicates that the command input string was modified and this field does not contain a value, the command will be rejected.
OINX_ROUTLLEN	X'30'	X'04'	Input	Length of the ROUTE list. If this field is zero, there is no ROUTE list; the default option of routing to all clients was selected.
OINX_ROUTLPTR	X'34'	X'04'	Input	Address of the ROUTE list. The ROUTE list cannot be modified by this exit. The ROUTE list is a list of client names separated by commas. The ROUTE list can contain a single asterisk as a client name, which routes to all clients.
	X'38'	X'10'	None	Reserved.

Contents of Registers on Exit

Register	Contents	
15	**Return Code**	**Meaning**

Register	Contents	
	0	Continue command processing
	4	Reject the command. This return code is ignored unless one of the following is true: • The exit routine is the last routine defined in the exit list for the input exit. • The exit routine sets the byte pointed to by UXPL_CALLNEXTP to the value UXPL_CALLNEXTNO.

All other registers must be restored.

CSL OM Output User Exit

This exit is called to allow a user to view and manipulate output from OM. This exit is optional.

This exit is called for the following events:

- A command has been processed and is ready to be delivered to the originator of the command. The exit can modify the command response text before the response is delivered.
- When an unsolicited message is received from a client (for example, an IMS control region) using the CSLOMOUT API.

This exit is defined as TYPE=OUTPUT in the EXITDEF statement in the BPE user exit list PROCLIB member. You can specify one or more user exits of this type. When this exit is invoked, all user exits of this type are driven in the order specified by the EXITS= keyword. For more information on how to define user exit module names, see the OM BPE user exit list PROCLIB member topic in *IMS Version 9: Base Primitive Environment Guide and Reference*.

This exit is invoked amode 31 and should be reentrant.

Contents of Registers on Entry

Register	Contents
1	Address of BPE user exit parameter list (mapped by macro BPEUXPL).
13	Address of the first of 2 prechained 72-byte save areas. These save areas are chained according to standard z/OS save area linkage convention. The first save area can be used by the exit to save registers on entry. The second save area is for use by routines called from the user exit.
14	Return address.
15	Entry point of exit routine.

On entry to the Output exit, register 1 points to a standard BPE user exit parameter list. Field UXPL_EXITPLP in this list contains the address of the OM Output user exit parameter list, which is mapped by macro CSLOOUX. Field UXPL_COMPTYPEP in this list points to the character string “OM,” indicating an OM address space.

OM Output User Exit Parameter List--Command Response: Table 16 on page 47 lists the user exit parameter list for command response. Included are the field name, the offset value and length, both in hexadecimal, how the field is used, and a brief description of the field.

Table 16. OM Output User Exit Parameter List--Command Response

Field Name	Offset	Length	Field Usage	Description
OOUX_PVER	X'00'	X'04'	Input	Parameter list version number (00000001).
OOUX_FUNC	X'04'	X'04'	Input	Function code **2** Command response.
OOUX_MBRNAME	X'08'	X'08'	Input	Client (IMSplex member) name that sent the command to OM.
OOUX_MBRTYPE	X'10'	X'02'	Input	IMSplex member type that sent the command to OM.
OOUX_OUTMOD	X'12'	X'01'	Output	Output modified indicator. This field indicates that the command output has been modified. The field should be set to 4 to have OM process the modified command response; otherwise, set the field to 0. • **0** Output was not modified. • **4** Output modified by the exit.
	X'13'	X'01'	None	Reserved.
OOUX_MBRSTYPE	X'14'	X'08'	Input	IMSplex member subtype that sent the command to OM.
OOUX_INPUTLEN	X'1C'	X'04'	Input	Length of the command input, if available.
OOUX_INPUTPTR	X'20'	X'04'	Input	Address of the command input, if available.
OOUX_OUTPTLEN	X'24'	X'04'	Input	Length of the command response.
OOUX_OUTPTPTR	X'28'	X'04'	Input	Address of the command response. Command response output is in XML format wrapped with the tags <imsout>...</imsout>.
OOUX_OUTMDLEN	X'2C'	X'04'	Output	Modified command output length. The exit must set this field if it modifies the command response output. This field must not be greater than the input command response length passed to this exit. If the exit does not set this field appropriately and does modify the command response output, the modified command response output will not be delivered to the client. Instead, the original command response output will be sent to the client.
OOUX_RQTKN1	X'30'	X'10'	Input	Request token 1.
OOUX_RQTKN2	X'40'	X'10'	Input	Request token 2.
OOUX_RETCODE	X'50'	X'04'	Input	Return code being sent to the client.
OOUX_RSNCODE	X'54'	X'04'	Input	Reason code being sent to the client
	X'58'	X'10'	None	Reserved.

OM Output User Exit Parameter List--Undeliverable Output: Table 17 lists the user exit parameter list for undeliverable output. Included are the field name, the offset value and length, both in hexadecimal, how the field is used, and a brief description of the field.

Table 17. OM Output User Exit Parameter List--Undeliverable Output

Field Name	Offset	Length	Field Usage	Description
OOUX_PVER	X'00'	X'04'	Input	Parameter list version number (00000001).

Table 17. OM Output User Exit Parameter List--Undeliverable Output (continued)

Field Name	Offset	Length	Field Usage	Description
OOUX_FUNC	X'04'	X'04'	Input	Function code **3** Undeliverable command response.
OOUX_MBRNAME	X'08'	X'08'	Input	Client (IMSplex member) name sending the command response.
OOUX_MBRTYPE	X'10'	X'02'	Input	IMSplex member type that sending the command response.
OOUX_OUTMOD	X'12'	X'01'	Output	Output modified field. This field indicates that the exit modified the command response string and that the updated command response should be processed. • **0** Output was not modified. • **4** Output modified by the exit. Undeliverable output does not get passed to any client.
	X'13'	X'01'	None	Reserved.
OOUX_MBRSTYPE	X'14'	X'08'	Input	IMSplex member subtype sending the command response.
OOUX_INPUTLEN	X'1c'	X'04'	Input	Length of the command input (if available)
OOUX_INPUTPTR	X'20'	X'04'	Input	Address of the command input (if available)
OOUX_OUTPTLEN	X'24'	X'04'	Input	Length of the command response or 0 if command response not available.
OOUX_OUTPTPTR	X'28'	X'04'	Input	Address of the command response if available. If the client failed to process the command, the client has returned only return/reason codes and no command response. In this case, the command response length field and this field will be zero.
OOUX_OUTMDLEN	X'2c'	X'04'	Output	Modified command output length. The exit must set this field if it modifies the command response output. This field must not be greater than the input command response length passed to this exit. If the exit does not set this field appropriately and does modify the command response output, the modified command response output will not be delivered to the client. Instead, the original command response output will be sent to the client. This field is ignored in IMS Version 9.
OOUX_RQTKN1	X'30'	X'10'	Input	Request token 1.
OOUX_RQTKN2	X'40'	X'10'	Input	Request token 2.
OOUX_RETCODE	X'50'	X'04'	Input	Return code from client.
OOUX_RSNCODE	X'54'	X'04'	Input	Reason code from client.
	X'58'	X'10'	None	Reserved.

OM Output User Exit Parameter List--Unsolicited Output: Table 18 lists the user exit parameter list for unsolicited output. Included are the field name, the offset value and length, both in hexadecimal, how the field is used, and a brief description of the field.

Table 18. OM Output User Exit Parameter List--Unsolicited Output

Field Name	Offset	Length	Field Usage	Description
OOUX_PVER	X'00'	X'04'	Input	Parameter list version number (00000001).
OOUX_FUNC	X'04'	X'04'	Input	Function code **4** Unsolicited output message.
OOUX_MBRNAME	X'08'	X'08'	Input	Client (IMSplex member) name sending the message.
OOUX_MBRTYPE	X'10'	X'02'	Input	IMSplex member type sending the message.
OOUX_OUTMOD	X'12'	X'01'	Output	Output modified field. This field indicates that the exit modified the output message string and that the updated output should be passed to the client. • **0** Output was not modified. • **4** Output modified by the exit. Unsolicited output does not get passed to any client.
	X'13'	X'01'	None	Reserved.
OOUX_MBRSTYPE	X'14'	X'08'	Input	IMSplex member subtype sending the message.
	X'1C'	X'04'	None	Reserved.
	X'20'	X'04'	None	Reserved.
OOUX_INPUTLEN	X'24'	X'04'	Input	Length of the message.
OOUX_INPUTPTR	X'28'	X'04'	Input	Address of the message.
OOUX_OUTMDLEN	X'2C'	X'04'	Output	Modified command output length. The exit must set this field if it modifies the command response output. This field must not be greater than the input command response length passed to this exit. If the exit does not set this field appropriately and does modify the command response output, the modified command response output will not be delivered to the client. Instead, the original command response output will be sent to the client. This field is ignored in IMS Version 9.
OOUX_RQTKN1	X'30'	X'10'	Input	Request token 1.
OOUX_RQTKN2	X'40'	X'10'	Input	Request token 2.
	X'50'	X'18'	None	Reserved.

Contents of Registers on Exit

Register	Contents	
	Return Code	**Meaning**
15	0	Always zero.

Register	Contents
	All other registers must be restored.

CSL OM Security User Exit

Use the OM Security user exit to perform security checking during command processing. This exit is given control after the OM Input exit. This exit is optional.

This exit is invoked when the CMDSEC= parameter on the OM procedure is specified as A or E:

A Both this exit and RACF (or equivalent) are used for OM command security

E Only this exit is called for OM command security

This exit is defined as `TYPE=SECURITY` in the EXITDEF statement in the BPE user exit list PROCLIB member. You can specify one or more user exits of this type. When this exit is invoked, all user exits of this type are driven in the order specified by the EXITS= keyword. For more information on how to define user exit module names, see the OM BPE user exit list PROCLIB member topic in *IMS Version 9: Base Primitive Environment Guide and Reference*.

This exit is invoked amode 31 and should be reentrant.

Contents of Registers on Entry

Register	Contents
1	Address of BPE user exit parameter list (mapped by macro BPEUXPL).
13	Address of the first of 2 prechained 72-byte save areas. These save areas are chained according to standard z/OS save area linkage convention. The first save area can be used by the exit to save registers on entry. The second save area is for use by routines called from the user exit.
14	Return address.
15	Entry point of exit routine.

On entry to the OM Security exit, register 1 points to a standard BPE user exit parameter list. Field UXPL_EXITPLP in this list contains the address of the OM Security user exit parameter list, which is mapped by macro CSLOSCX. Field UXPL_COMPTYPEP in this list points to the character string “OM,” indicating an OM address space.

OM Security User Exit Parameter List: Table 19 lists the user exit parameter list for the security user exit. Included are the field name, the offset value and length, both in hexadecimal, how the field is used, and a brief description of the field.

Table 19. OM Security User Exit Parameter List

Field Name	Offset	Length	Field Usage	Description
OSCX_PVER	X'00'	X'04'	Input	Parameter list version number (00000002).
OSCX_FUNC	X'04'	X'04'	Input	Function code **1** Perform user command security checking.
OSCX_MBRNAME	X'08'	X'08'	Input	Client (IMSplex member) name that sent the command to OM.

Table 19. OM Security User Exit Parameter List (continued)

Field Name	Offset	Length	Field Usage	Description
OSCX_MBRTYPE	X'10'	X'02'	Input	IMSplex member type that sent the command to OM.
	X'12'	X'02'	None	Reserved.
OSCX_MBRSTYPE	X'14'	X'08'	Input	IMSplex member subtype that sent the command to OM.
OSCX_USERID	X'1C'	X'08'	Input	User ID of application where the command originated.
OSCX_VERB	X'24'	X'10'	Input	Command verb.
OSCX_KEYWORD	X'34'	X'10'	Input	Primary keyword.
OSCX_INPUTLEN	X'44'	X'04'	Input	Length of the command input string.
OSCX_INPUTPTR	X'48'	X'04'	Input	Address of the command input string.
OSCX_SECCODE	X'4C'	X'04'	Input	Decoded security code. Only valid when the CMDSEC= parameter on the OM procedure is specified as A. • X'00000000': RACF security permits command. • X'00000004': RACF security was not requested. • X'00000008': RACF security requested, but RACF is not available. • X'0000000C': User ID not defined to RACF. • X'00000010': Command not protected by RACF. • X'00000014': User ID is not authorized for the command.
OSCX_SAFCODE	X'50'	X'04'	Input	Security Authorization Facility (SAF) return code. This is only valid when CMDSEC=A is specified.
OSCX_RETCODE	X'54'	X'04'	Input	RACF return code. This is only valid when CMDSEC=A is specified.
OSCX_RSNCODE	X'58'	X'04'	Input	RACF reason code. This is only valid when CMDSEC=A is specified.
OSCX_USERDATA	X'5C'	X'20'	Output	User data. This data is encapsulated by the <userdata> tags in the <cmdsecerr> section of the XML output, if this exit has rejected the command. This user data can contain alphanumeric characters (A-Z, 0-9), or printable characters (not case sensitive), with the exception of the characters &, <, and >. OM will convert any invalid data placed in this field to periods (.) before sending the XML output to the client.
OSCX_ROUTLEN	X'7C'	X'04'	Input	The length of the ROUTE list. If this value is zero (0), no route list exists. The command was routed to all command processing clients that were Ready or Registered.
OSCX_ROUTLPTR	X'80'	X'04'	Input	The address of the ROUTE list. You cannot use this exit to modify the ROUTE list.
	X'84'	X'08'	None	Reserved.

Contents of Registers on Exit

Register	Contents	
15	**Return Code**	**Meaning**
	0	Accept the command for processing.
	4	Reject the command due to an unauthorized user ID. This return code is ignored unless the exit routine is one of the following: • The exit routine is the last routine defined in the exit list for the security exit • The exit routine sets the byte pointed to by UXPL_CALLNEXTP to the value UXPL_CALLNEXTNO

All other registers must be restored.

CSL OM Statistics Available through BPE Statistics User Exit

The BPE Statistics user exit can be used to gather both BPE and OM statistics. Refer to the BPE user exit topic of *IMS Version 9: Base Primitive Environment Guide and Reference* for details on the exit and the events that drive it.

This topic describes OM statistics that are:

- available to the BPE Statistics user exit when driven from an OM address space
- returned on a `CSLZQRY FUNC=STATS` request directed to the OM address space

When the user exit is driven, field BPESTXP_COMPSTATS_PTR in the BPE Statistics user exit parameter list, BPESTXP, contains the pointer to the OM statistics header. When the `CSLZQRY FUNC=STATS` request is made, the OUTPUT= buffer points to the output area mapped by CSLZQRYO. The output area field ZQYO_STXOFF contains the offset to the OM statistics header. The header is mapped by CSLOSTX.

CSL OM Statistics Header

Table 20 lists the OM statistics header. Included are the offset value and length, both in hexadecimal, how the field is used, and a brief description of the field.

Table 20. OM Statistics Header

Field Name	Offset	Length	Field Usage	Description
OSTX_ID	X'00'	X'08'	Input	Eyecatcher "CSLOSTX".
OSTX_LEN	X'08'	X'04'	Input	Length of header.
OSTX_PVER	X'0C'	X'04'	Input	Header version number (0000001).
OSTX_PLEXCNT	X'10'	X'04'	Input	Number of IMSplexes for which statistics are available.
OSTX_STATCNT	X'14'	X'04'	Input	Number of statistics areas available for each IMSplex.
OSTX_STATLEN	X'18'	X'04'	Input	Length of all statistics areas for each IMSplex.
OSTX_STATOFF	X'1C'	X'04'	Input	Offset to statistics area for first IMSplex. This is the offset from the beginning of CSLOSTX. The offset points to the CSLOST1 area for the first IMSplex.

Table 20. OM Statistics Header (continued)

Field Name	Offset	Length	Field Usage	Description
OSTX_OST1OFF	X'20'	X'04'	Input	Offset to the OM request statistics record for activity performed by OM requests (mapped by macro CSLOST1). The offset is from the start of the statistics area for this IMSplex. Refer to Table 21 for a description of the OM Request statistics record.
OSTX_OST2OFF	X'24'	X'04'	Input	Offset to OM IMSplex statistics record for activity performed by OM for an IMSplex (mapped by macro CSLOST2). The offset is from the start of the statistics area for this IMSplex. Refer to Table 22 on page 54 for a description of the OM IMSplex statistics record.
	X'28'	X'04'	None	Reserved.
	X'2C'	X'04'	None	Reserved.

CSL OM Statistics Record CSLOST1

CSLOST1 contains statistics related to specific requests and commands that are processed by OM. Table 21 lists the OM statistics record CSLOST1. Included are the field names, the offset value and length, both in hexadecimal, how the field is used, and a brief description of the field.

Table 21. OM Statistics Record CSLOST1

Field Name	Offset	Length	Field Usage	Description
OST1_ID	X'00'	X'08'	Input	Eyecatcher "CSLOST1".
OST1_LEN	X'08'	X'04'	Input	Length of valid data.
OST1_PVER	X'0C'	X'04'	Input	Statistics version number (00000001).
OST1_OMREG	X'10'	X'04'	Input	Number of CSLOMREG requests.
OST1_OMRDY	X'14'	X'04'	Input	Number of CSLOMRDY requests.
	X'18'	X'04'	None	Reserved.
OST1_OMDRG	X'1C'	X'04'	Input	Number of CSLOMDRG requests.
OST1_OMDRGIN	X'20'	X'04'	Input	Number of internal deregister (normal term) requests.
OST1_OMDRGIA	X'24'	X'04'	Input	Number of internal deregister (abnormal term) requests.
OST1_OMICMD	X'28'	X'04'	Input	Number of CSLOMI command requests.
OST1_OMIQRY	X'2C'	X'04'	Input	Number of CSLOMI query requests.
	X'30'	X'04'	None	Reserved.
	X'34'	X'04'	None	Reserved.
	X'38'	X'04'	None	Reserved.
	X'3C'	X'04'	None	Reserved.
OST1_OMCMD	X'40'	X'04'	Input	Number of CSLOMCMD requests.
OST1_OMQRYCLN	X'44'	X'04'	Input	Number of CSLOMQRY client requests.
OST1_OMQRYSYN	X'48'	X'04'	Input	Number of CSLOMQRY syntax requests.
	X'4C'	X'04'	None	Reserved.
	X'50'	X'04'	None	Reserved.

Table 21. OM Statistics Record CSLOST1 (continued)

Field Name	Offset	Length	Field Usage	Description
	X'54'	X'04'	None	Reserved.
	X'58'	X'04'	None	Reserved.
OST1_OMRSP	X'5C'	X'04'	Input	Number of CSLOMRSP requests.
OST1_OMOUT	X'60'	X'04'	Input	Number of CSLOMOUT requests.
	X'64'	X'04'	None	Reserved.
	X'68'	X'04'	None	Reserved.
	X'6C'	X'04'	None	Reserved.
	X'70'	X'04'	None	Reserved.
	X'74'	X'04'	None	Reserved.
OST1_ZQRY	X'78'	X'04'	Input	Number of CSLZQRY requests.
OST1_ZSHUT	X'7C'	X'04'	Input	Number of CSLZSHUT requests.
	X'80'	X'04'	None	Reserved.
	X'84'	X'04'	None	Reserved.
	X'88'	X'04'	None	Reserved.
OST1_QRYIPLX	X'8C'	X'04'	Input	Number of QRY IMSPLEX commands.
	X'90'	X'04'	None	Reserved.
	X'94'	X'04'	None	Reserved.
	X'98'	X'04'	None	Reserved.
	X'9C'	X'04'	None	Reserved.
	X'A0'	X'04'	None	Reserved.
	X'A4'	X'04'	None	Reserved.
	X'A8'	X'04'	None	Reserved.
	X'AC'	X'04'	None	Reserved.

CSL OM Statistics Record CSLOST2

CSLOST2 contains statistics that are related to an IMSplex, but not to a specific request or command. Table 22 lists the OM statistics record CSLOST2. Included are the field name, the offset value and length, both in hexadecimal, how the field is used, and a brief description of the field.

Table 22. OM Statistics Record CSLOST2

Field Name	Offset	Length	Field Usage	Description
OST2_ID	X'00'	X'08'	Input	Eyecatcher "CSLOST2".
OST2_LEN	X'08'	X'04'	Input	Length of valid data.
OST2_PVER	X'0C'	X'04'	Input	Parameter list version number (00000001).
OST2_PLEXNAME	X'10'	X'08'	Input	IMSplex name.
OST2_CLIENTS	X'18'	X'04'	Input	Number of active clients in the IMSplex.
OST2_CMDTOUT	X'1C'	X'04'	Input	Number of times a command was timed out.
OST2_UNDELIV	X'20'	X'04'	Input	Number of times a command response output could not be returned to the client.
	X'24'	X'04'	None	Reserved.
	X'28'	X'04'	None	Reserved.

Table 22. OM Statistics Record CSLOST2 (continued)

Field Name	Offset	Length	Field Usage	Description
	X'2C'	X'04'	None	Reserved.
	X'30'	X'04'	None	Reserved.
	X'34'	X'04'	None	Reserved.
	X'38'	X'04'	None	Reserved.
	X'3C'	X'04'	None	Reserved.
	X'40'	X'04'	None	Reserved.
	X'44'	X'04'	None	Reserved.
	X'48'	X'04'	None	Reserved.
	X'50'	X'04'	None	Reserved.
	X'54'	X'04'	None	Reserved.
	X'58'	X'04'	None	Reserved.

CSL Automated Operator Program Requests

This topic describes the requests that can be used by AOP clients, for example, the TSO SPOC.

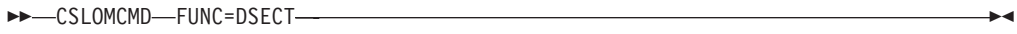
CSLOMCMD: Command Request

This API can be used by an AOP client application running on the host.

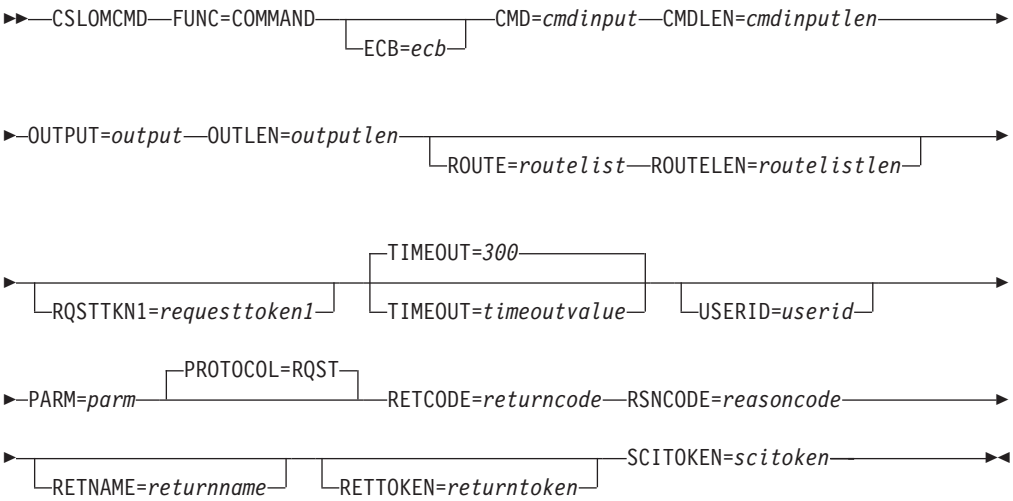
CSLOMCMD Syntax

The syntax for CSLOMCMD can vary depending on what the automated operator client intends to perform. Parameter descriptions for each syntax example are provided in "CSLOMCMD Parameters" on page 56.

DSECT Syntax: Use the DSECT function of a CSLOMCMD request to include equate (EQU) statements in your program for the CSLOMCMD parameter list length and return and reason codes.

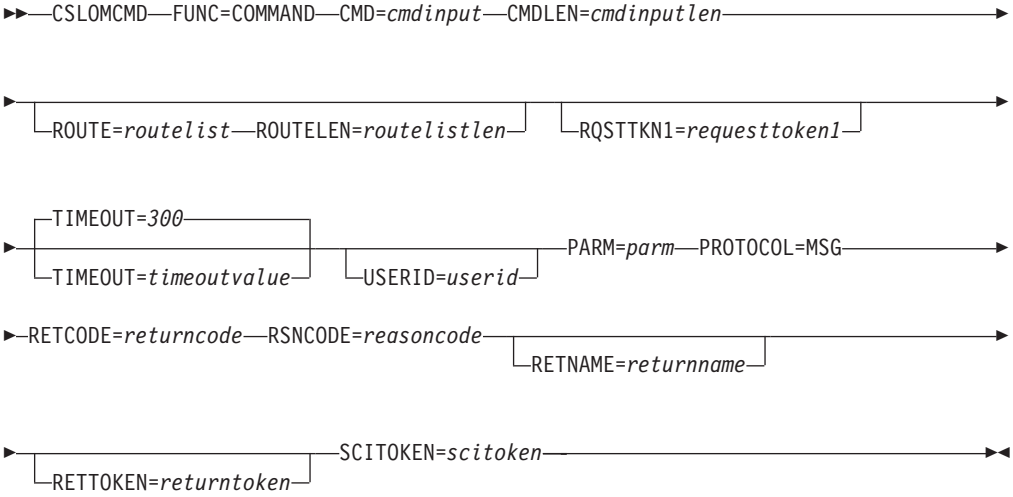


Request Protocol Syntax: For automation clients that want to wait for the output from the OM request, use this syntax.



The response is passed back to the client after the request is completed.

Message Protocol Syntax: For automation clients that want to receive command output through their user exit, use this syntax.



The response is passed back to the client using the SCI Input exit. The client must have specified an SCI Input exit (INPUTEXIT=) on the SCI registration request (CSLSCREG) to receive a response. For more information on the SCI registration request, see “CSLSCREG: Registration Request” on page 187.

CSLOMCMD Parameters

The CSLOMCMD request parameters follow.

CMD=*symbol*
CMD=*(r2-r12)*
(Required) - Specifies the command input buffer. This can be any IMS command that can be specified through the OM API. The first character of the command does not need to be a command recognition character (for example, /). The command recognition character does not control command routing in OM. The ROUTE= keyword controls which IMSplex members receive a

command. If a command recognition character is entered in the command string it is ignored. The first character in the command is considered a command recognition character if it is not a character between A-Z (either uppercase or lowercase).

CMDLEN=*symbol*
CMDLEN=*(r2-r12)*
(Required) - Specifies the length of the command input buffer.

ECB=*symbol*
ECB=*(r2-r12)*
(Optional) - Specifies the address of a z/OS event control block (ECB) used for asynchronous requests. When the request is complete, the ECB specified is posted. If an ECB is not specified, the task is suspended until the request is complete. If an ECB is specified, the invoker of the macro must issue a WAIT (or equivalent) after receiving control from CSLOMCMD before using or examining any data returned by this macro (including the RETCODE and RSNCODE fields).

OUTLEN=*symbol*
OUTLEN=*(r2-r12)*
(Required for RQST) - Specifies a 4-byte field to receive the length of the output returned by the CSLOMCMD request. OUTLEN contains the length of the output pointed to by the OUTPUT= parameter.

The output length is zero if no output is built, for example, if an error is detected before any output can be built.

OUTPUT=*symbol*
OUTPUT=*(r2-r12)*

(Required for RQST) - Specifies a 4-byte field to receive the address of the variable length output returned by the CSLOMCMD request. The output contains the command response output. The output length is returned in the OUTLEN= field.

The output address is zero if no output was built, for example, if an error was detected before any output could be built.

The output buffer is not preallocated by the caller. After the request returns it, this word contains the address of a buffer containing the update output. It is the caller's responsibility to release this storage by issuing the CSLSCBFR FUNC=RELEASE request when it is finished with the storage. The length of the output is returned in the OUTLEN= field.

PARM=*symbol*
PARM=*(r1-r12)*
(Required) - Specifies the CSLOMCMD parameter list. The length of the parameter list must be equal to the parameter list length EQU value defined by OCMD_PARMLN.

PROTOCOL=RQST
PROTOCOL=MSG
(Optional) - Specifies the SCI protocol for sending the request to OM.

- RQST - Send command to OM using the SCI request protocol.
- MSG - Send command to OM using the SCI message protocol.

RETCODE=*symbol*
RETCODE=*(r2-r12)*
(Required) - Specifies a 4-byte field to receive the return code on output. OM

return codes are defined in CSLORR. SCI return codes are defined in CSLSRR. Possible return codes are described in Table 23 on page 59.

The return code can be from OM (CSLOMCMD) or SCI (CSLSCMSG or CSLSCRQS). If ECB is specified, the RETCODE is not valid until the ECB is posted. All return codes contain the SCI member type indicator for either SCI, OM, or RM in the high order byte (X'01' for SCI, X'02' for OM, X'03' for RM).

RETNAME=*symbol*
RETNAME=*(r2-r12)*
(Optional) - Specifies an 8-byte output field to receive the OM name. This is the CSL member name of the target address space to which SCI sent the request.

RETTOKEN=*symbol*
RETTOKEN=*(r2-r12)*
(Optional) - Specifies a 16-byte output field to receive the OM SCI token returned to the caller. This is the OM SCI token for the target address space to which the request was sent.

ROUTE=*symbol*
ROUTE=*(r2-r12)*
(Optional) - Specifies a route list that identifies OM clients (for example, IMS control regions) in the IMSplex to which the command is sent. To explicitly route the command to all command processing clients that have registered for and are ready to process commands, specify ROUTE=*. If you do not specify ROUTE, OM routes to all clients that are registered and ready to process commands.

Note: Use commas to separate a list of client names.

ROUTELEN=*symbol*
ROUTELEN=*(r2-r12)*
(Optional) - Specifies the length of the list specified in the ROUTE= parameter.

RQSTTKN1=*symbol*
RQSTTKN1=*(r2-r12)*
(Optional) - Specifies a 16-byte user generated request token that is used to associate the request response with the request for asynchronous processing. RQSTTKN1 can include A-Z, 0-9, or printable characters (not case sensitive), except &, <, and >. OM returns the request token encapsulated in the <rqsttkn1></rqsttkn1> tags in the XML output. OM converts any invalid data to periods (.) before returning XML output to the client. For PROTOCOL=MSG requests, OM also returns the address of this token in the OM Directive parameter list (mapped by CSLOMDIR macro) in the field ODIR_CQRT1PTR. This parameter must be 16 bytes and, if necessary, padded with blanks.

For information on XML output, see Appendix A, “CSL Operations Manager XML Output,” on page 203. For information on CSLOMDIR, see “CSL OM Directives” on page 94.

RSNCODE=*symbol*
RSNCODE=*(r2-r12)*
(Required) - Specifies a 4-byte field to receive the reason code on output. OM reason codes are defined in CSLORR. SCI reason codes are defined in CSLSRR. Possible reason codes are described in Table 23 on page 59.

SCITOKEN=*symbol*
SCITOKEN=*(r2-r12)*
(Required) - Specifies a 16-byte field containing the SCI token. This token

uniquely identifies this connection to SCI. The SCI token is returned by a successful `CSLSCREG FUNC=REGISTER` request.

TIMEOUT=*timeoutvalue*
TIMEOUT=*symbol*
TIMEOUT=*(r2-r12)*

(Optional) - Specifies a 4-byte command timeout value in seconds. If the TIMEOUT value is reached during OM command processing and before all clients have responded to the command, OM terminates the command and returns all available responses. If too small a value is specified, an incomplete response is returned. The TIMEOUT value ensures a response is returned even if a client processing the command is unable to respond. The TIMEOUT keyword is ignored if no CMD keyword is specified. If a command is requested but no timeout value is specified, a timeout value of 5 minutes is used.

If TIMEOUT is specified as a symbol, the symbol must be an EQU symbol equated to the timeout value. If TIMEOUT is specified as a number, the number must be the timeout value.

USERID=*symbol*
USERID=*(r2-r12)*

(Optional) - Specifies the 8-byte user ID that should be used for security checking for the command and keyword combination. This user ID is used only if the client address space is an authorized caller. If the client address space is unauthorized, the user ID is obtained from z/OS control blocks. This user ID is intended for use by authorized system management address spaces that can issue an OM request on behalf of another address space or remote client. In this case, the user ID of the client address space is not the user ID of the actual client, so it must be passed to OM. This parameter must be 8 bytes, left-justified, and, if necessary, padded with blanks.

CSLOMCMD Return and Reason Codes

The return are reason codes in Table 23 can be returned on a CSLOMCMD macro request.

Table 23. CSLOMCMD Return and Reason Codes

Return Code	Reason Code	Meaning
X'00000000'	X'00000000'	The request completed successfully.

Table 23. CSLOMCMD Return and Reason Codes (continued)

Return Code	Reason Code	Meaning
X'02000004'	Any code	This return code indicates a warning. All or part of the request might have completed successfully. Additional information is returned with the response to the request.
	X'00001000'	The command timed out before all of the command response information could be collected. One or more clients might not be responding or a client might have needed more time to process the command. If you specified the TIMEOUT option, make sure that the interval is long enough to allow the command to process. All command response information that is collected prior to the time-out is returned. This reason code is also returned if CSL members such as SCI or RM are not active on the local or remote z/OS image and cannot process the request or return a response. To obtain more information, issue QUERY IMSPLEX to determine which CSL members are inactive. Restart those members and re-issue the request. If this reason code is returned after an INIT OLC or TERM OLC command, issue QUERY MEMBER to determine the online change status of the IMS systems participating in the online change and take action based on their status.
	X'00001004'	The INPUT exit rejected the command specified in the CMD field. The command was not processed.
	X'00001008'	The client (specified in the corresponding XML <mbr></mbr> tags in the <cmderr> section) was specified in the ROUTE list for the command specified in the CMD field. However, the specified client was overridden with the ANY option. This option enables routing to any client that processes commands.
	X'0000100C'	The client (specified in the corresponding XML <mbr></mbr> tags in the <cmderr> section) was specified in the ROUTE list for the command specified in the CMD field. However, the specified client was overridden with the ALL option. This option enables routing to all clients that processes commands.

Table 23. CSLOMCMD Return and Reason Codes (continued)

Return Code	Reason Code	Meaning
X'02000008'	Any code	This return code indicates a parameter error. The request was not processed due to the error.
	X'00002000'	The command specified in the CMD field is invalid.
	X'00002004'	The command specified in the CMD field contains a keyword that is invalid with that command.
	X'00002028'	The command string specified in the CMD field contains an invalid keyword.
	X'0000202C'	BPE detected an unknown positional parameter in the command string specified in the CMD field.
	X'00002030'	The command string specified in the CMD field contains a keyword with an equals sign when a sublist was expected. For example, *keyword*= was specified instead of *keyword*().
	X'00002034'	The command string specified in the CMD field contains an incomplete keyword or keyword parameter.
	X'00002038'	The command string specified in the CMD field is missing a keyword.
	X'0000203C'	The command string specified in the CMD field contains an invalid keyword parameter.
	X'00002040'	The command string specified in the CMD field contains a duplicate keyword.
	X'00002044'	The command contains invalid syntax. Text containing the syntax error is returned in the <message></message> XML tags in the error log.
	X'00002050'	The caller of the service attempted to pass an invalid parameter list. The request is rejected.
X'0200000C'	Any code	This return code indicates a list error. The request might or might not have processed. Refer to the <cmderr> section and the completion codes for each command processing client listed in the <cmdrspdata> section.
	X'00003000'	The command was routed to multiple clients. At least one client was able to process the request successfully and return either command response data or a response message to the SPOC. Refer to the completion codes, CC field, for further information.
	X'00003004'	The command was routed to multiple clients. None of the clients were able to process the request successfully. No command response data or response messages were returned.
	X'00003008'	The command was routed to multiple clients. None of the clients that processed the command returned a return code and reason code to the OM. At least one command client returned either command response data or a response message.
	X'0000300C'	The command was routed to multiple clients. Not all of the clients that processed the command returned a return code 0 and reason code 0 to the OM. Also, at least one client returned a return code 4. Refer to the completion codes returned on the request for additional information.

Table 23. CSLOMCMD Return and Reason Codes (continued)

Return Code	Reason Code	Meaning
X'02000010'	Any code	This return code indicates an environmental error. The request could not be processed due to the current environment. This condition might be temporary.
	X'00004000'	The command specified in the CMD field could not be processed by the client indicated in the corresponding <mbr></mbr> tags in the <cmderr> section because the client was not yet ready to process commands.
	X'00004004'	The command specified in the CMD field could not be processed by the client indicated in the corresponding XML <mbr></mbr> tags in the <cmderr> section because the client was not registered for the command.
	X'00004008'	The command specified in the CMD field could not be processed by the client indicated in the corresponding XML <mbr></mbr> tags in the <cmderr> section because the client is not active in the IMSplex.
	X'0000400C'	The command specified in the CMD field could not be processed by the client indicated in the corresponding XML <mbr></mbr> tags in the <cmderr> section because the client registered for the command with an invalid PADEF grammar.
	X'00004020'	This version of the parameter list is invalid.
	X'00004010'	The command specified in the CMD field could not be processed. The client that issued the command is not authorized. Examine the <cmdsecerr> section in the error log to determine why the client is not authorized.

Table 23. CSLOMCMD Return and Reason Codes (continued)

Return Code	Reason Code	Meaning
X'02000014'	Any code	This return code indicates a system error. An internal error occurred. The command was not processed.
	X'00005000'	An internal OM error occurred while allocating a CMD block for processing of the command specified in the CMD field. Contact the IBM Support Center.
	X'00005004'	An internal OM error occurred while allocating a CRSP block to process the command specified in the CMD field. Contact the IBM Support Center.
	X'00005008'	An internal OM error occurred while allocating the command input buffer to process the command specified in the CMD field. Contact the IBM Support Center.
	X'0000500C'	An internal OM error occurred while processing of the command specified in the CMD field. Contact the IBM Support Center.
	X'00005010'	An internal OM error occurred while obtaining storage for the parsed output blocks to parse the command specified in the CMD field. Contact the IBM Support Center.
	X'00005014'	An internal OM error occurred while adding the CMD block to the command instance hash table during processing of the command specified in the CMD field. Contact the IBM Support Center.
	X'00005018'	An internal OM error occurred while accessing the CMD block in the command instance hash table during processing of the command specified in the CMD field. Contact the IBM Support Center.
	X'0000501C'	An internal OM error occurred while scanning for the CMD block in the command instance hash table during processing of the command specified in the CMD field. Contact the IBM Support Center.
	X'00005020'	An internal OM error occurred while processing the command specified in the CMD field. The command was not processed by the command processing client. See the <cmderr> section of the error log for the member name of the command processing client, and contact the IBM Support Center.
	X'00005024'	An internal OM error occurred while processing the command specified in the CMD field. The command was not processed by the command processing client. See the <cmderr> section for the member name of the command processing client, and contact the IBM Support Center.
	X'00005028'	An internal OM error occurred while parsing the command specified in the CMD field. Contact the IBM Support Center.

CSLOMI: API Request

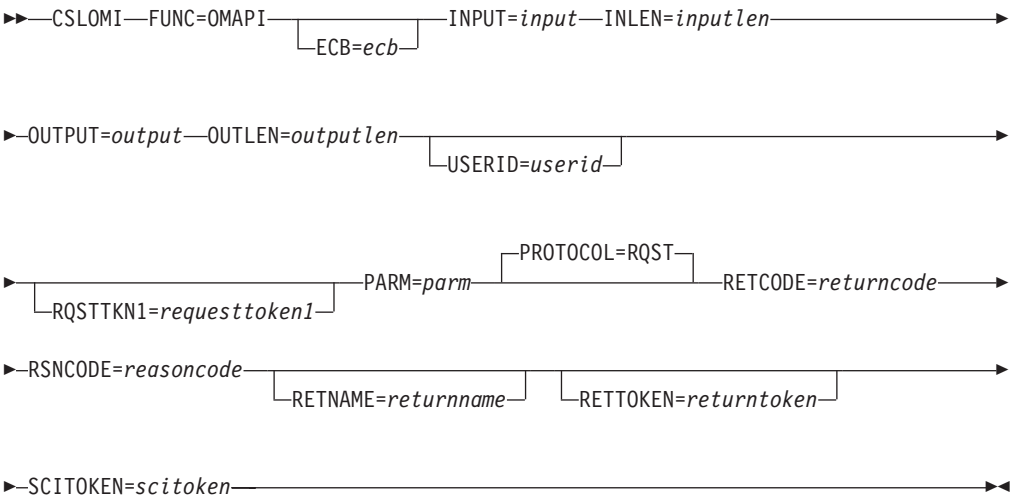
With the CSLOMI request, a z/OS automated operator client can issue an IMS command to or request OM-specific information from an OM. The CSLOMI macro interface is designed for use by system management address spaces that receive

input from a workstation or other z/OS address space and must pass the request to OM. In this case the workstation application builds the input string and passes it to the z/OS address space. The z/OS address space passes the input string to OM on the INPUT= parameter.

CSLOMI Syntax

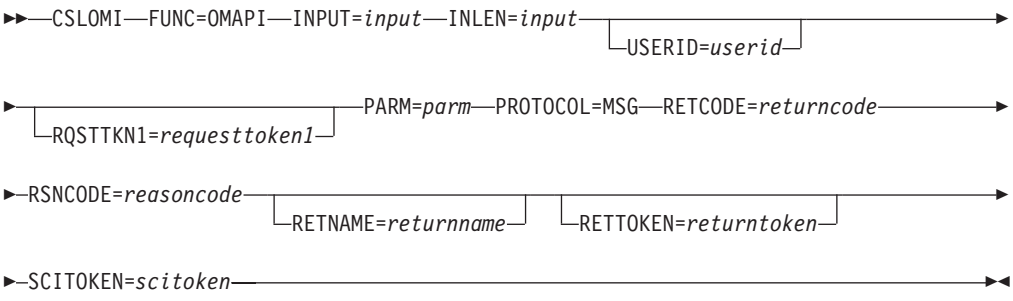
The syntax for CSLOMI can vary, depending on how the automated operator client wants to receive the command response. If the client does not have an input exit and wants to receive the command output as a response, use the request syntax. If the client does have an input exit and wants to receive the command output as a message, use the message syntax. Parameter descriptions for each syntax example are provided in “CSLOMI Request and Message Parameters” on page 65.

CSLOMI Request Protocol Syntax: For automated clients that want to wait for output from the OM request, use this syntax.



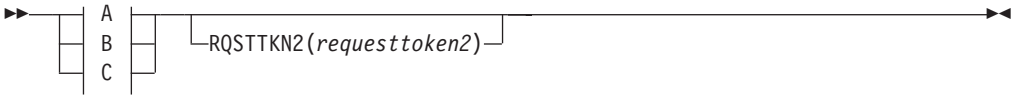
After control is returned to the client (if ECB is not specified), or the ECB is posted (if an ECB is specified), the response is available to the client.

CSLOMI Message Protocol Syntax: For automated clients that want to receive command output through their user exit, use this syntax:

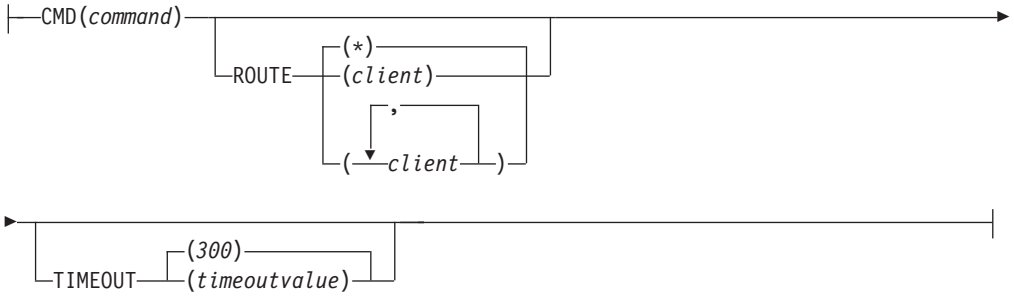


The response is passed back to the client using the SCI Input exit. The client must have specified an SCI Input exit (INPUTEXIT=) on the SCI registration request (CSLSCREG) to receive a response. For more information on the SCI registration request, see “CSLSCREG: Registration Request” on page 187.

CSLOMI Input= Parameter Syntax: For other applications or workstations that do not communicate directly with OM, use this syntax.

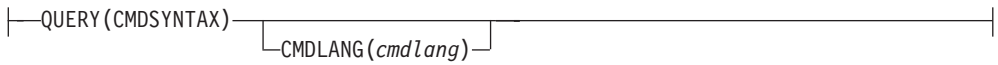


A:



B:

C:



This syntax is used for the INPUT= parameter. The application builds the command or query, and passes it to a z/OS address space that communicates with OM directly.

CSLOMI Request and Message Parameters

The CSLOMI request and message parameters follow. The parameters for the CSLOMI Input Syntax (shown in "CSLOMI Input= Parameter Syntax") are described in "CSLOMI Input= Parameters" on page 67.

ECB=*symbol*
ECB=*(r2-r12)*
(Optional) - Specifies the address of a z/OS event control block (ECB) used for asynchronous requests. When the request is complete, the ECB specified is posted. If an ECB is not specified, the task is suspended until the request is complete. If an ECB is specified, the invoker of the macro must issue a WAIT (or equivalent) after receiving control from CSLOMI before using or examining any data returned by this macro (including the RETCODE and RSNCODE fields).

INLEN=*symbol*
INLEN=*(r2-r12)*
(Required) - Specifies the length of the input buffer.

INPUT=*symbol*
INPUT=*(r2-r12)*
(Required) - Specifies the address of the input buffer.

Figure 16 shows an example of the input buffer that is passed to CSLOMI. The input buffer is the character field MYINPUT and specifies three parameters: a command string of QRY TRAN SHOW(ALL), a timeout value of 360 seconds, and a route list consisting of one element, IMSA:

```
CSLOMI FUNC=OMAPI,INPUT=MYINPUT,INLEN=INPUTLEN
INPUTLEN DC    A(MYINPUTL)
MYINPUT  DC    C'CMD (QRY TRAN SHOW(ALL) TIMEOUT(360) ROUTE(IMSA)'
MYINPUTL EQU   *-MYINPUT
```

Figure 16. Sample Input Buffer Passed to CSLOMI

OUTLEN=*symbol*
OUTLEN=*(r2-r12)*
(Required for RQST) - Specifies a 4-byte field to receive the length of the output returned by the CSLOMI request. OUTLEN contains the length of the output pointed to by the OUTPUT= parameter.

The output length is zero if no output is built, for example, if an error is detected before any output can be built.

OUTPUT=*symbol*
OUTPUT=*(r2-r12)*
(Required) - Specifies a field to receive the variable length output returned by the CSLOMI request. The output contains the command response output. The output length is returned in the OUTLEN= field.

The output address is zero if no output was built, for example, if an error was detected before any output could be built.

The output buffer is not preallocated by the caller. After the request returns it, this word contains the address of a buffer containing the update output. It is the caller's responsibility to release this storage by issuing the CSLSCBFR FUNC=RELEASE request when it is finished with the storage. The length of the output is returned in the OUTLEN= field.

PARM=*symbol*
PARM=*(r1-r12)*
(Required) - Specifies the CSLOMI parameter list. The length of the parameter list must be equal to the parameter list length EQU value defined by OI_PARMLN.

PROTOCOL=RQST
PROTOCOL=MSG
(Optional) - Specifies the SCI protocol for sending the request to OM.

- RQST - Send command to OM using the SCI request protocol.
- MSG - Send command to OM using the SCI message protocol.

RETCODE=*symbol*
RETCODE=*(r2-r12)*
(Required) - Specifies a 4-byte field to receive the return code on output. OM return codes are defined in CSLORR. SCI return codes are defined in CSLSRR. Possible return codes are described in Table 24 on page 69.

The return code can be from OM (CSLOMI) or SCI (CSLSCMSG or CSLSCRQS). If ECB is specified, the RETCODE is not valid until the ECB is posted. All return codes contain the SCI member type indicator for either SCI, OM, or RM in the high order byte (X'01' for SCI, X'02' for OM, X'03' for RM).

RETNAME=*symbol*

RETNAME=*(r2-r12)*
(Optional) - Specifies an 8-byte output field to receive the OM name. This is the CSL member name of the target address space to which SCI sent the request.

RETTOKEN=*symbol*
RETTOKEN=*(r2-r12)*
(Optional) - Specifies a 16-byte output field to receive the OM SCI token returned to the caller. This is the OM SCI token for the target address space to which the request was sent.

RQSTTKN1=*symbol*
RQSTTKN1=*(r2-r12)*
(Optional) - Specifies a 16-byte user generated request token that is used to associate the request response with the request for asynchronous processing. RQSTTKN1 can include A-Z, 0-9, or printable characters (not case sensitive), except &, <, and >. OM returns the request token encapsulated in the <rqsttkn1></rqsttkn1> tags in the XML output. OM converts any invalid data to periods (.) before returning XML output to the client. For PROTOCOL=MSG requests, OM also returns the address of this token in the OM Directive parameter list (mapped by CSLOMDIR macro) in the field ODIR_CQRT1PTR. This parameter must be 16 bytes and, if necessary, padded with blanks.

For information on XML output, see Appendix A, “CSL Operations Manager XML Output,” on page 203. For information on CSLOMDIR, see “CSL OM Directives” on page 94.

RSNCODE=*symbol*
RSNCODE=*(r2-r12)*
(Required) - Specifies a 4-byte field to receive the reason code on output. OM reason codes are defined in CSLORR. SCI reason codes are defined in CSLSRR. Possible reason codes are described in Table 24 on page 69.

SCITOKEN=*symbol*
SCITOKEN=*(r2-r12)*
(Required) - Specifies a 16-byte field containing the SCI token. This token uniquely identifies this connection to SCI. The SCI token is returned by a successful `CSLSCREG FUNC=REGISTER` request.

USERID=*symbol*
USERID=*(r2-r12)*
(Optional) - Specifies the 8-byte user ID that should be used for security checking for the command and keyword combination. This user ID is used only if the client address space is an authorized caller. If the client address space is unauthorized, the user ID is obtained from z/OS control blocks. This user ID is intended for use by authorized system management address spaces that can issue an OM request on behalf of another address space or remote client. In this case, the user ID of the client address space is not the user ID of the actual client, so it must be passed to OM. This parameter must be 8 bytes, left-justified, and, if necessary, padded with blanks.

CSLOMI Input= Parameters

The parameters for the CSLOMI input option are for applications and workstations that do not communicate directly with OM.

CMD(*command***)**
(Required if QUERY is not specified) - Specifies the command input buffer. This can be any IMS command that can be specified through the OM API. The first character of the command does not need to be a command recognition character (for example, /). The command recognition character does not control command routing in OM. The ROUTE keyword is used to control which IMSplex

members receive a command. If a command recognition character is entered in the command string, it is ignored. The first character in the command is considered a command recognition character if it is not a character between A-Z (either uppercase or lowercase).

CMDLANG(*cmdlang***)**
The language to be used for IMS command text that is returned on the request. This value defaults to the default established for the OM system specified on the OM startup parameter CMDLANG=. Currently the only accepted value is ENU for US English. If an invalid language is specified text in the OM default language is returned.

QUERY(*querytype***)**
Type of query to be performed by OM.

CMDCLIENTS
Requests that OM return a list of all clients (for example, IMS control regions) that have registered to OM for command processing.

The list of clients is returned encapsulated in <cmdclients> </cmdclients> tags. *querytype* can be one of the following.

<mbr name=*membername*>

The member name is the name of the client address space.

<typ> </typ>

The member type is the type of the client address space.

<styp> </styp>

The member subtype is the subtype of the client address space.

<vsn> </vsn>

The member version is the version of the client address space.

<jobname> </jobname>

The client jobname is the jobname or the started task for the client address space.

</mbr>

CMDSYNTAX
Requests that OM return a list of the XML representing the command syntax for selected commands registered with OM. Additionally, the translatable text associated with the command syntax is returned.

The command syntax XML is returned encapsulated in <cmdsyntax> </cmdsyntax> tags. The command syntax DTD is returned encapsulated in <cmddtd> </cmddtd> tags. The command syntax translatable text is returned encapsulated in <cmdtext> </cmdtext> tags. For more information on XML output, see Appendix A, “CSL Operations Manager XML Output,” on page 203.

ROUTE(*routelist***)**
(Optional) - Specifies a route list that identifies OM clients (for example, IMS control regions) in the IMSplex to which the command is sent. In the list, the clients are separated by commas. To explicitly route the command to all command processing clients that have registered for and are ready to process commands, specify ROUTE(*). If you do not specify ROUTE, OM routes to all clients that are registered and ready to process commands.

RQSTTKN2(*requesttoken2***)**
(Optional) - Specifies a 16-byte user generated request token that is used to associate the request response with the request for asynchronous processing.

RQSTTKN2 can include A-Z, 0-9, or printable characters (not case sensitive), except &, <, and >. OM returns the request token encapsulated in the <rqsttkn2></rqsttkn2> tags in the XML output. OM converts any invalid data to periods (.) before returning XML output to the originating client. For `PROTOCOL=MSG` requests, OM also returns the address of this token in the OM Directive parameter list (mapped by CSLOMDIR macro) in the field ODIR_CQRT2PTR.

For information on XML output, see Appendix A, "CSL Operations Manager XML Output," on page 203. For information on CSLOMDIR, see "CSL OM Directives" on page 94.

TIMEOUT(*timeoutvalue* **)**

(Optional) - Specifies a 4-byte command timeout value in seconds. If the TIMEOUT value is reached during OM command processing before all clients have responded to the command, OM terminates the command and returns all available responses. If too small a value is specified, an incomplete response is returned. The TIMEOUT value ensures a response is returned even if a client processing the command cannot respond. The TIMEOUT keyword is ignored if no CMD keyword is specified. If a command is requested but no timeout value is specified, a timeout value of 5 minutes is used.

CSLOMI Return and Reason Codes

Table 24 lists the return and reason codes that can be returned on a CSLOMI macro request. Also included is the meaning of a reason code (that is, what possibly caused it).

Table 24. CSLOMI Return and Reason Codes

Return Code	Reason Code	Meaning
X'00000000'	X'00000000'	The request completed successfully.

Table 24. CSLOMI Return and Reason Codes (continued)

Return Code	Reason Code	Meaning
X'02000004'	any code	This return code represents a warning. All or part of the request might have completed successfully. Additional information is returned with the response to the request.
	X'00001000'	The specified command timed out before all of the command response information could be collected. One or more clients might not be responding, or a client might have needed more time to process the command. If a TIMEOUT value is specified, ensure the value is long enough to allow for the command to be processed. All command response information that is collected prior to the time-out is returned.
	X'00001004'	The INPUT exit rejected the command contained in the CMD field. The command was not processed.
	X'00001008'	The client (specified in the corresponding XML <mbr></mbr> tags in the <cmderr> section) was specified in the ROUTE list for the command specified in the CMD field. However, the specified client was overridden with the ANY option. This option enables routing to any client that processes commands.
	X'0000100C'	The client (specified in the corresponding XML <mbr></mbr> tags in the <cmderr> section) was specified in the ROUTE list for the command specified in the CMD field. However, the specified client was overridden with the ALL option. This option enables routing to all clients that processes commands.
	X'00001010'	The text file could not be loaded in the language specified on the CMDLANG parameter. The default language is used.

Table 24. CSLOMI Return and Reason Codes (continued)

Return Code	Reason Code	Meaning
X'02000008'	any code	This return code represents a parameter error. The request was not processed due to the error.
	X'00002000'	The command specified in the CMD field is invalid.
	X'00002004'	The primary keyword specified in the CMD field is invalid with the command specified.
	X'00002028'	An invalid keyword was specified in the CMD field.
	X'0000202C'	BPE detected an unknown positional parameter in the command in the CMD field.
	X'00002030'	A keyword was specified with an equal sign (KEYWORD=) when a sublist was expected (KEYWORD()) in the command in the CMD field.
	X'00002034'	An incomplete keyword or keyword parameter was specified in the command in the CMD field.
	X'00002038'	A keyword is missing from the command in the CMD field.
	X'0000203C'	The value of a keyword parameter specified in the command was invalid.
	X'00002040'	A duplicate keyword was specified in the command in the CMD field.
	X'00002044'	Text containing the syntax error is returned in the XML <message></message> tags.
	X'00002048'	More than one filter was specified.
	X'00002050'	The caller of the service attempted to pass an invalid parameter list. The request is rejected.
X'0200000C'	any code	This return code represents a list error. The request might or might not have been processed due to the error. Refer to the XML tag <cmderr> section and the completion codes for each command processing client listed in the XML tag <cmdrspdata> section.
	X'00003000'	The command was routed to multiple clients. At least one client was able to process the request successfully and return either command response data or a response message. Refer to the completion codes returned on the request for further information.
	X'00003004'	The command was routed to multiple clients. None of the clients were able to process the request successfully. No command response data or response messages were returned by any client.
	X'00003008'	The command was routed to multiple clients. None of the clients that processed the command returned a return code 0 and reason code 0 to OM. At least one command client returned either command response data or a response message.
	X'0000300C'	The command was routed to multiple clients. Not all of the clients that processed the command returned a return code 0 and reason code 0 to the OM. Also, at least one client returned a return code 4. Refer to the completion codes returned on the request for additional information.

Table 24. CSLOMI Return and Reason Codes (continued)

Return Code	Reason Code	Meaning
X'02000010'	any code	This return code represents an environmental error. The request could not be processed at this time due to the current environment. This condition might be temporary.
	X'00004000'	The command contained in the CMD field could not be processed by the client indicated in the corresponding XML <mbr></mbr> tags in the <cmderr> section because the client was not yet ready to process commands.
	X'00004004'	The command contained in the CMD field could not be processed by the client indicated in the corresponding XML <mbr></mbr> tags in the <cmderr> section because the client was not registered for the command.
	X'00004008'	The command contained in the CMD field could not be processed by the client indicated in the corresponding XML <mbr></mbr> tags in the <cmderr> section because the client is not active in the IMSplex.
	X'0000400C'	The command contained in the CMD field could not be processed by the client indicated in the corresponding XML <mbr></mbr> tags in the <cmderr> section because the client registered for the command with invalid PADEF grammar.
	X'00004010'	The command contained in the CMD field could not be processed. The client that issued the command is not authorized. Examine the <cmdsecerr> section in the XML file to determine why the client is not authorized.
	X'00004014'	A data set allocation error occurred; the data set specified by the CMDTEXT= DSN parameter could not be allocated.
	X'00004018'	A data set read error occurred; a member in the data set specified by the CMDTEXT= DSN could not be read. The member name is 'CSLOT' concatenated with the 3-character CMDLANG value.
	X'00004020'	The parameter list version is invalid.

Table 24. CSLOMI Return and Reason Codes (continued)

Return Code	Reason Code	Meaning
X'02000014'	any code	This return code represents a system error. An internal error occurred, and the command was not processed.
	X'00005000'	An OM internal error occurred. Due to a storage shortage, OM was unable to allocate a CMD block to process the command in the CMD field.
	X'00005004'	An OM internal error occurred. Due to a storage shortage, OM was unable to allocate a CRSP block to process the command in the CMD field.
	X'00005008'	An OM internal error occurred. Due to a storage shortage, OM was unable to allocate the command input buffer to process the command in the CMD field.
	X'0000500C'	An OM internal error occurred. OM was unable to obtain the VERB latch while processing the command in the CMD field.
	X'00005010'	An OM internal error occurred. Due to a storage shortage, OM was unable to obtain storage for the parsed output blocks to parse the command in the CMD field.
	X'00005014'	An OM internal error occurred. OM was unable to add the CMD block to the command instance hash table while processing the command in the CMD field.
	X'00005018'	An OM internal error occurred. OM was unable to find the CMD block in the command instance hash table while processing the command in the CMD field.
	X'0000501C'	An OM internal error occurred. OM was unable to scan for the CMD block in the command instance hash table while processing the command in the CMD field.
	X'00005020'	An OM internal error occurred. OM was unable to obtain a system AWE while processing the command in the CMD field. The command was not processed by the command processing client. Refer to the <cmderr> section in the XML file for the member name of the command processing client.

Table 24. CSLOMI Return and Reason Codes (continued)

Return Code	Reason Code	Meaning
X'02000014' *(continued)*	X'00005024'	An OM internal error occurred. OM was unable to queue a system AWE while processing the commanding the CMD field. The command was not processed by the command processing client. Refer to the <cmderr> section of the XML file for the member name of the command processing client.
	X'00005028'	An OM internal error occurred. OM was unable to parse the command contained in the CMD field due to a BPEPARSE internal error.
	X'0000502C'	An OM internal error occurred. The command output header allocation failed.
	X'00005030'	An OM internal error occurred. The command output response allocation failed.
	X'00005034'	An OM internal error occurred. The OUTPUT buffer allocation failed.
	X'00005038'	An OM internal error occurred. The VERB hash table add failed.
	X'0000503C'	An OM internal error occurred. The CLNT block could not be obtained.
	X'00005040'	An OM internal error occurred. The CSLSCQRY request failed.

CSLOMQRY: Query Request

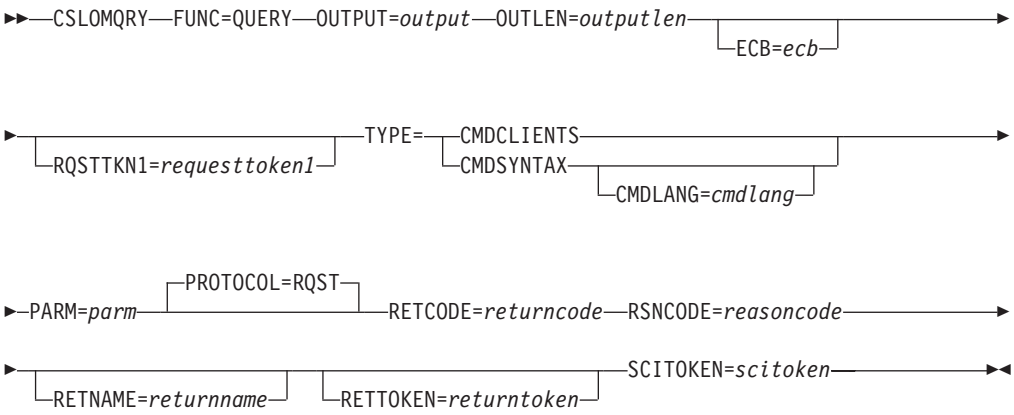
With the CSLOMQRY request, any AOP client running on the host can request OM-specific information.

CSLOMQRY Syntax

The syntax for CSLOMQRY can vary depending on what the automated operator client intends to perform. Parameter descriptions for each syntax example are provided in “CSLOMQRY Parameters” on page 75.

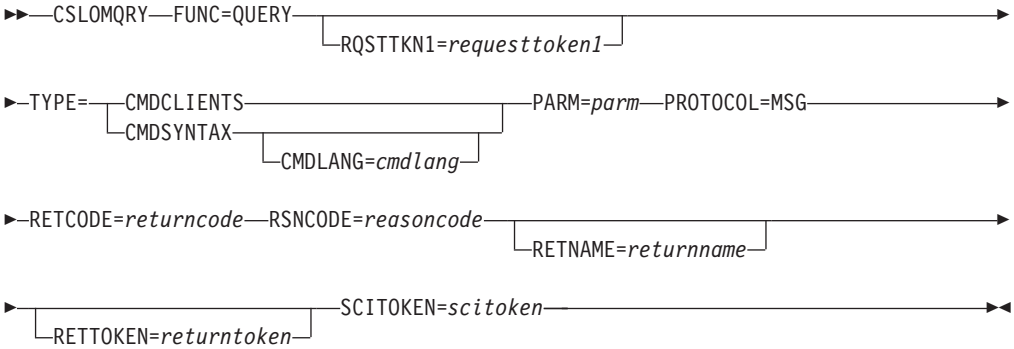
DSECT Syntax: Use the DSECT function of a CSLOMQRY request to include equate (EQU) statements in your program for the CSLOMQRY parameter list length and return and reason codes.

Request Protocol Syntax: For automation clients that want to wait for the output from the OM request, use this syntax.



The response is passed back to the client after the request is completed.

Message Protocol Syntax: For automation clients that want to send a message to OM to process an OM request, use this syntax.



The response is passed back to the client using the SCI Input exit. The client must have specified an SCI Input exit (INPUTEXIT=) on the SCI registration request (CSLSCREG) to receive a response. For more information on the SCI registration request, see “CSLSCREG: Registration Request” on page 187.

CSLOMQRY Parameters

The CSLOMQRY parameters follow.

CMDLANG=*cmdlang*

(Optional) - The language to be used for IMS command text that is returned on the request. This value defaults to the default established for the OM system specified on the OM startup parameter CMDLANG=. Currently the only accepted value is ENU for US English. If an invalid language is specified in OM, the default language is returned.

ECB=*symbol*
ECB=*(r2-r12)*

(Optional) - Specifies the address of a z/OS event control block (ECB) used for asynchronous requests. When the request is complete, the ECB specified is posted. If an ECB is not specified, the task is suspended until the request is complete. If an ECB is specified, the invoker of the macro must issue a WAIT

(or equivalent) after receiving control from CSLOMQRY before using or examining any data returned by this macro (including the RETCODE and RSNCODE fields).

OUTLEN=*symbol*
OUTLEN=*(r2-r12)*
(Required for RQST) - Specifies a 4-byte field to receive the length of the output returned by the CSLOMQRY request. OUTLEN contains the length of the output pointed to by the OUTPUT= parameter.

The output length is zero if no output is built, for example, if an error is detected before any output can be built.

OUTPUT=*symbol*
OUTPUT=*(r2-r12)*
(Required) - Specifies a field to receive the variable length output returned by the CSLOMQRY request. The output contains the command response output. The output length is returned in the OUTLEN= field.

The output address is zero if no output was built, for example, if an error was detected before any output could be built.

The output buffer is not preallocated by the caller. After the request returns it, this word contains the address of a buffer containing the update output. It is the caller's responsibility to release this storage by issuing the `CSLSCBFR FUNC=RELEASE` request when it is finished with the storage. The length of the output is returned in the OUTLEN= field.

PARM=*symbol*
PARM=*(r1-r12)*
(Required) - Four-byte input parameter that specifies the address of the storage used by the request to pass the parameters to SCI. The length of the parameter list must be equal to the parameter list length EQU value defined by OQRY_PARMLN.

PROTOCOL=RQST
PROTOCOL=MSG
(Optional) - Specifies the SCI protocol for sending the request to OM.
- RQST - Send command to OM using the SCI request protocol.
- MSG - Send command to OM using the SCI message protocol.

RETCODE=*symbol*
RETCODE=*(r2-r12)*
(Required) - Specifies a 4-byte field to receive the return code on output. OM return codes are defined in CSLORR. SCI return codes are defined in CSLSRR. Possible return codes are described in Table 25 on page 78.

The return code can be from OM (CSLOMQRY) or SCI (CSLSCMSG or CSLSCRQS). If ECB is specified, the RETCODE is not valid until the ECB is posted. All return codes contain the SCI member type indicator for either SCI, OM, or RM in the high order byte (X'01' for SCI, X'02' for OM, X'03' for RM).

RETNAME=*symbol*
RETNAME=*(r2-r12)*
(Optional) - Specifies an 8-byte output field to receive the OM name. This is the CSL member name of the target address space to which SCI sent the request.

RETTOKEN=*symbol*

RETTOKEN=*(r2-r12)*
(Optional) - Specifies a 16-byte output field to receive the OM SCI token returned to the caller. This is the OM SCI token for the target address space to which the request was sent.

RQSTTKN1=*symbol*
RQSTTKN1=*(r2-r12)*
(Optional) - Specifies a 16-byte user generated request token that is used to associate the request response with the request for asynchronous processing. RQSTTKN1 can include A-Z, 0-9, or printable characters (not case sensitive), except &, <, and >. OM returns the request token encapsulated in the <rqsttkn1></rqsttkn1> tags in the XML output. OM converts any invalid data to periods (.) before returning XML output to the client. For `PROTOCOL=MSG` requests, OM also returns the address of this token in the OM directive parameter list (mapped by CSLOMDIR macro) in the field ODIR_CQRT1PTR. This parameter must be 16 bytes and, if necessary, padded with blanks.

For information on XML output, see Appendix A, "CSL Operations Manager XML Output," on page 203. For information on CSLOMDIR, see "CSL OM Directives" on page 94.

RSNCODE=*symbol*
RSNCODE=*(r2-r12)*
(Required) - Specifies a 4-byte field to receive the reason code on output. OM reason codes are defined in CSLORR. SCI reason codes are defined in CSLSRR. Possible reason codes are described in Table 25 on page 78.

SCITOKEN=*symbol*
SCITOKEN=*(r2-r12)*
(Required) - Specifies a 16-byte field containing the SCI token. This token uniquely identifies this connection to SCI. The SCI token is returned by a successful `CSLSCREG FUNC=REGISTER` request.

TYPE=CMDCLIENTS
TYPE=CMDSYNTAX
(Required) - Four-byte input parameter that specifies the type of query to be performed by OM.

CMDCLIENTS
Requests that OM return a list of all clients (for example, IMS control regions) that have registered to OM for command processing.

The clients are returned encapsulated in <cmdclients> </cmdclients> tags.

<mbr name="*membername*">

The member name is the name of the client address space.

<typ> </typ>

The member type is the type of the client address space.

<styp> </styp>

The member subtype is the subtype of the client address space.

<vsn> </vsn>

The member version is the version of the client address space.

<jobname> </jobname>

The client jobname is the jobname or the started task for the client address space.

</mbr>

For more information on XML output, see Appendix A, “CSL Operations Manager XML Output,” on page 203.

CMDSYNTAX
Requests that OM return a list of the XML representing the command syntax for selected commands registered with OM. Additionally, the translatable text associated with the command syntax is returned.

The command syntax XML is returned encapsulated in <cmdsyntax> </cmdsyntax> tags. The command syntax DTD is returned encapsulated in <cmddtd> </cmddtd> tags. The command syntax translatable text is returned encapsulated in <cmdtext> </cmdtext> tags.

The command syntax and translatable text that is returned as a result of the CSLOMQRY QUERY TYPE(CMDSYNTAX) request includes information for type-2 commands.

CSLOMQRY Return and Reason Codes

Table 25 lists the return and reason code combinations that can be returned on a CSLOMQRY request and that are unique to the CSLOMQRY request. Also included is the meaning of a reason code (that is, what possibly caused it).

Table 25. CLSOMQRY Return and Reason Codes

Return Code	Reason Code	Meaning
X'00000000'	X'00000000'	The request completed successfully.
X'02000004'	any code	This return code represents a warning. All or part of the request might have completed successfully. Additional information is returned with the response to the request.
	X'00001010'	The text file could not be loaded in the language specified on the CMDLANG parameter. The default language is used.
X'02000008'	X'00002050'	The caller of the service attempted to pass an invalid parameter list. The request is rejected.
X'02000010'	any code	This return code represents an environmental error. The request could not be processed at this time due to the current environment. This condition might be temporary.
	X'00004014'	A data set allocation error occurred; the data set specified by the CMDTEXTDSN= parameter in the OM Initialization PROCLIB member (CSLOIxxx) could not be allocated.
	X'00004018'	A data set read error occurred; a member in the data set specified by the CMDTEXTDSN= parameter in the OM Initialization PROCLIB member (CSLOIxxx) could not be read. The member name is ’CSLOT’ concatenated with the 3-character CMDLANG value.
	X'00004020'	The parameter list version is invalid.

CSL OM Command Processing Client Requests

This topic describes the requests made by command processing clients.

If you are writing your own command processing client, ensure that the access authority you provide on the RACF PERMIT command matches the access authority with which the command is registered.

The following topics provide additional information:

CSLOMBLD: Command Registration Build

CSLOMBLD is used to build the command list that is passed to OM on the CSLOMREG request. This list:

- identifies the commands for which the IMS system can be called.
- is comprised of a set of statements starting with a CSLOMBLD FUNC=BEGIN statement and ending with a CSLOMBLD FUNC=END statement

Any number of CSLOMBLD FUNC=DEFVRB statements can be provided, each one defining the command verb. Following each DEFVRB statement are CSLOMBLD FUNC=DEFKEY statements, which identify keywords valid for the previously defined command verb.

The set of CSLOMBLD statements can be defined either in a separate data-only assembler module, or in a static data section of an executable assembler module. Refer to the documentation in the CSLOMBLD macro.

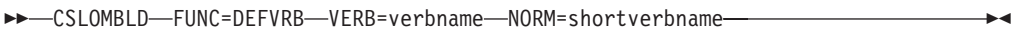
CSLOMBLD is used to build the command registration list; it does not have an input parameter list.

CSLOMBLD Syntax

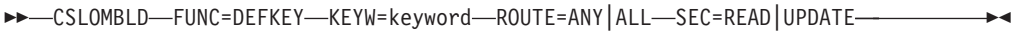
The syntax for the set of CSLOMBLD statements follows.

CSLOMBLD BEGIN: Use the BEGIN function statement to identify the beginning of the set of command statements.

CSLOMBLD DEFVRB: Use the DEFVRB function statement to identify a command that the OM client or IMS system will support. You can specify a short form of the command verb.



CSLOMBLD DEFKEY: Use the DEFKEY function statement to identify a keyword that is valid for the previously defined command. You can also specify command routing and required RACF authorization with this statement.



Refer to "Overriding CSL OM Command Routing with the ROUTE Parameter" on page 81 for information on the ROUTE parameter. Refer to "CSL OM Command Security" on page 39 for information on the SEC parameter.

CSLOMBLD DEFGMR: Use the DEFGMR function statement to identify the beginning of the statements that describe the output parsing grammar.

Note: This function is for internal IBM use only.

```
►►—CSLOMBLD—FUNC=DEFGMR————————————————————————————————►◄
```

CSLOMBLD ENDGMR: Use the ENDGMR function statement to designate the end of the statements that describe the output parsing grammar.

Note: This function is for internal IBM use only.

```
►►—CSLOMBLD—FUNC=ENDGMR————————————————————————————————►◄
```

CSLOMBLD END: Use the END function statement to designate the end of the list of command statements.

```
►►—CSLOMBLD—FUNC=END————————————————————————————————————►◄
```

CSLOMBLD Parameters

The CSLOMBLD parameters follow.

KEYW=*keyword*
Specifies a valid keyword for the command verb that immediately precedes this parameter. For a null keyword, use blanks; for example, 'KEYW= '. This parameter is required for FUNC=DEFKEY.

NORM=*shortverbname*
Specifies the short form of the command being defined. This parameter is required for FUNC=DEFVRB.

ROUTE=*ANY* | *ALL*
Specifies the override routing for the command being defined. This parameter is required for FUNC=DEFKEY. For more information on this parameter, see "Overriding CSL OM Command Routing with the ROUTE Parameter" on page 81.

SEC=*READ* | *UPDATE*
Specifies the required RACF authorization for KEYW. This parameter is required for FUNC=DEFKEY. For more information on this parameter, see "CSL OM Command Security" on page 39.

VERB=*verbname*
Specifies the long form of the command being defined. This parameter is required for FUNC=DEFVRB.

CSLOMBLD Example

Figure 17 on page 81 shows an example of a set of CSLOMBLD statements.

```
CSLOMBLD FUNC=BEGIN
CSLOMBLD FUNC=DEFVRB,VERB=ACTIVATE,NORM=ACT
CSLOMBLD FUNC=DEFKEY,KEYW=LINK,SEC=UPDATE
CSLOMBLD FUNC=DEFKEY,KEYW=NODE,SEC=UPDATE
CSLOMBLD FUNC=END
```

Figure 17. CSLOMBLD Example Statements

Overriding CSL OM Command Routing with the ROUTE Parameter

CSLOMBLD allows the command processing client to override the routing that you specify when you enter a command. There are a few commands that specify command routing overrides. OM overrides command routing when two command processing clients specify different routing overrides for the same command if:

- At least one command processing client specifies an override of ROUTE=ALL, then OM routes the command to all registered command processing clients.
- At least one command processing client specifies an override of ROUTE=ANY, and no command processing client has specified ROUTE=ALL, then OM routes the command to one of the registered command processing clients.
- No command processing clients have specified an override of ROUTE=ALL or ROUTE=ANY, then OM routes the command as specified by the user that entered the command.

For more information on commands, see *IMS Version 9: Command Reference*.

CSLOMDRG: Command Deregistration Request

With the CSLOMDRG request, a command processing client like the IMS control region tells OM that it no longer wants to process commands, for example, when the client address space is terminating. The deregister request removes all client information from the OM command registration tables and stops OM from sending further commands to the client.

CSLOMDRG Syntax

The syntax for the CSLOMDRG request follows.

DSECT Syntax: Use the DSECT function of a CSLOMDRG request to include equate (EQU) statements in your program for the CSLOMDRG parameter list length and return and reason codes.

```
►►—CSLOMDRG—FUNC=DSECT————————————————————————————————————►◄
```

Request Protocol Syntax:

```
►►—CSLOMDRG—FUNC=DEREGISTER—PARM=parm—RETCODE=returncode————————————————►
►—RSNCODE=reasoncode—SCITOKEN=scitoken————————————————————————————————►◄
```

CSLOMDRG Parameters

The parameters for CSLOMDRG follow.

PARM=*symbol*

PARM=*(r1-r12)*
(Required) - Specifies the CSLOMDRG parameter list. The length of the parameter list must be equal to the parameter list length EQU value defined by ODRG_PARMLN.

RETCODE=*symbol*
RETCODE=*(r2-r12)*
(Required) - Specifies a 4-byte field to receive the return code on output. OM return codes are defined in CSLORR. SCI return codes are defined in CSLSRR. Possible return codes are described in Table 26.

The return code can be from OM (CSLOMDRG) or SCI (CSLSCMSG or CSLSCRQS). If ECB is specified, the RETCODE is not valid until the ECB is posted. All return codes contain the SCI member type indicator for either SCI, OM, or RM in the high order byte (X'01' for SCI, X'02' for OM, X'03' for RM).

RSNCODE=*symbol*
RSNCODE=*(r2-r12)*
(Required) - Specifies a 4-byte field to receive the reason code on output. OM reason codes are defined in CSLORR. SCI reason codes are defined in CSLSRR. Possible reason codes are described in Table 26.

SCITOKEN=*symbol*
SCITOKEN=*(r2-r12)*
(Required) - Specifies a 16-byte field containing the SCI token. This token uniquely identifies this connection to SCI. The SCI token is returned by a successful CSLSCREG FUNC=REGISTER request.

CSLOMDRG Return and Reason Codes

The return and reason codes in Table 26 can be returned on a CSLOMDRG macro request. Also included is the meaning of a reason code (that is, what possibly caused it).

Table 26. CLSOMDRG Return and Reason Codes

Return Code	Reason Code	Meaning
X'00000000'	X'00000000'	The request completed successfully.

CSLOMOUT: Unsolicited Output Request

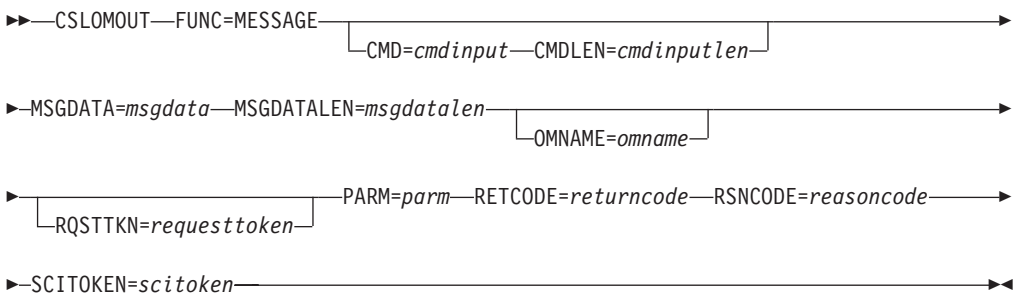
The CSLOMOUT request is issued by a command processing client that wants to send a message not directly in response to a command. The message can be additional information as a result of a command issued after the initial command response is returned to OM; or it can be an informational message as a result of an event in the system. OM sends the unsolicited message to the OM Output user exit.

CSLOMOUT Syntax

The syntax examples for the CSLOMOUT request follow.

DSECT Syntax: Use the DSECT function of a CSLOMOUT request to include equate (EQU) statements in your program for the CSLOMOUT parameter list length and return and reason codes.

Request Protocol Syntax:



CSLOMOUT Parameters

The parameters for the CSLOMOUT request follow.

CMD=*symbol*
CMD=*(r2-r12)*
(Optional) - Specifies the command input buffer. This can be any IMS command that can be specified through the OM API. This parameter represents the original command input.

CMDLEN=*symbol*
CMDLEN=*(r2-r12)*
(Optional) - Specifies the length of the command input buffer.

MSGDATA=*symbol*
MSGDATA=*(r2-r12)*
(Required) - Specifies the command response message buffer.

MSGDATALEN=*symbol*
MSGDATALEN=*(r2-r12)*
(Required) - Specifies the length of the command response message buffer.

OMNAME=*symbol*
OMNAME=*(r2-r12)*
(Optional) - Specifies the 8-byte OM name to which to send the unsolicited output message when the message is an asynchronous response to a command.

RQSTTKN=*symbol*
RQSTTKN=*(r2-r12)*
(Optional) - Specifies the 32-byte request token that was passed to the command processing client on an OM command directive. This represents the RQSTTKN1 and RQSTTKN2 fields that are entered on either or both the CSLOMI and CSLOMCMD requests.

PARM=*symbol*
PARM=*(r1-r12)*
(Required) - Specifies the CSLOMOUT parameter list. The length of the parameter list must be equal to the parameter list length EQU value defined by ORDY_PARMLN.

RETCODE=*symbol*
RETCODE=*(r2-r12)*
(Required) - Specifies a 4-byte field to receive the return code on output. OM return codes are defined in CSLORR. SCI return codes are defined in CSLSRR. Possible return codes are described in Table 27 on page 84.

The return code can be from OM (CSLOMOUT) or SCI (CSLSCMSG or CSLSCRQS). If ECB is specified, the RETCODE is not valid until the ECB is posted. All return codes contain the SCI member type indicator for either SCI, OM, or RM in the high order byte (X'01' for SCI, X'02' for OM, X'03' for RM).

RSNCODE=*symbol*
RSNCODE=*(r2-r12)*
(Required) - Specifies a 4-byte field to receive the reason code on output. OM reason codes are defined in CSLORR. SCI reason codes are defined in CSLSRR. Possible reason codes are described in Table 27.

SCITOKEN=*symbol*
SCITOKEN=*(r2-r12)*
(Required) - Specifies a 16-byte field containing the SCI token. This token uniquely identifies this connection to SCI. The SCI token is returned by a successful `CSLSCREG FUNC=REGISTER` request.

CSLOMOUT Return and Reason Codes

The return and reason codes in Table 27 can be returned on a CSLOMOUT macro request. lists the return and reason codes that can be returned on a CSLOMI macro request. Also included is the meaning of a reason code (that is, what possibly caused it).

Table 27. CLSOMOUT Return and Reason Codes

Return Code	Reason Code	Meaning
X'00000000'	X'00000000'	The request completed successfully.

CSLOMRDY: Ready Request

With the CSLOMRDY request, command processing clients like the IMS control region notify OM that they are ready to process commands. OM does not send commands to a client until this request is processed.

CSLOMRDY Syntax

The syntax for the CSLOMRDY request follows.

CSLOMRDY DSECT Syntax: Use the DSECT function of a CSLOMRDY request to include equate (EQU) statements in your program for the CSLOMRDY parameter list length and return and reason codes.

```
►►—CSLOMRDY—FUNC=DSECT———————————————————————————————————►◄
```

CSLOMRDY Request Protocol Syntax:

```
►►—CSLOMRDY—FUNC=READY—┬———————————————————————————┬—PARM=parm———►
                        └—OMNAME=omname—┬———————————┬—┘
                                        └—MASTER=NO|YES—┘

►—RETCODE=returncode—RSNCODE=reasoncode—SCITOKEN=scitoken———————————►◄
```

CSLOMRDY Parameters

The parameters for CSLOMRDY follow.

MASTER=NO
MASTER=YES
(Optional) - Specifies whether or not the client should be chosen as the

command master. If a client specifies MASTER=YES, OM can select that client to be the command master. If the client specifies MASTER=NO, OM selects it to be the command master only if no other client has specified MASTER=YES.

For more information on how OM chooses an IMS command master, see *IMS Version 9: Command Reference*.

OMNAME=*symbol*
OMNAME=*(r2-r12)*
(Optional) - Specifies the 8-byte OM name to which to send the command ready request. If an OM name is not specified, the ready request is sent to all OM address spaces registered in the IMSplex.

PARM=*symbol*
PARM=*(r1-r12)*
(Required) - Specifies the CSLOMRDY parameter list. The length of the parameter list must be equal to the parameter list length EQU value defined by ORDY_PARMLN.

RETCODE=*symbol*
RETCODE=*(r2-r12)*
(Required) - Specifies a 4-byte field to receive the return code on output. OM return codes are defined in CSLORR. SCI return codes are defined in CSLSRR. Possible return codes are described in Table 28.

The return code can be from OM (CSLOMRDY) or SCI (CSLSCMSG or CSLSCRQS). If ECB is specified, the RETCODE is not valid until the ECB is posted. All return codes contain the SCI member type indicator for either SCI, OM, or RM in the high order byte (X'01' for SCI, X'02' for OM, X'03' for RM).

RSNCODE=*symbol*
RSNCODE=*(r2-r12)*
(Required) - Specifies a 4-byte field to receive the reason code on output. OM reason codes are defined in CSLORR. SCI reason codes are defined in CSLSRR. Possible reason codes are described in Table 28.

SCITOKEN=*symbol*
SCITOKEN=*(r2-r12)*
(Required) - Specifies a 16-byte field containing the SCI token. This token uniquely identifies this connection to SCI. The SCI token is returned by a successful `CSLSCREG FUNC=REGISTER` request.

CSLOMRDY Return and Reason Codes

Table 28 lists the return and reason codes that can be returned on a CSLOMRDY macro request. Also included is the meaning of the reason code (that is, what possibly caused it).

Table 28. CLSOMRDY Return and Reason Codes

Return Code	Reason Code	Meaning
X'00000000'	X'00000000'	The request completed successfully.

CSLOMREG: Command Registration Request

With the CSLOMREG request, a command processing client, such as the IMS control region, can register commands with an OM. The registration tells OM which commands a client can process. CSLOMREG must be the first request that a command processing client issues to OM. A command processing client must register with all OM address spaces in the IMSplex. If a client is registered with only one OM in an IMSplex, and that OM goes down, the client's commands are not

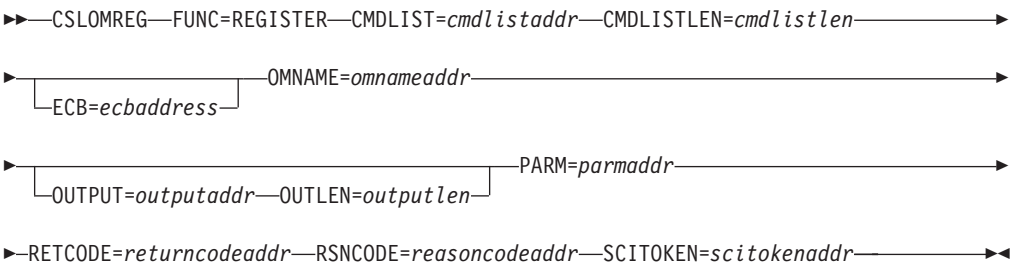
routed to another OM in the IMSplex. Use the CSLSCQRY request to obtain the names of all OMs in the IMSplex. The client must be authorized to issue a CSLOMREG request. This authorization is from SCI, which notifies OM that the client can issue requests.

CSLOMREG Syntax

The syntax examples for CSLOMREG follow.

CSLOMREG DSECT Syntax: Use the DSECT function of a CSLOMREG request to include equate (EQU) statements in your program for the CSLOMREG parameter list length and return and reason codes.

CSLOMREG Request Protocol Syntax: The syntax for the CSLOMREG request follows.



CSLOMREG Parameters

The parameters for CSLOMREG follow:

CMDLIST=*symbol*
CMDLIST=*(r2-r12)*
(Required) - Specifies the command definition list.

The command list is built using the CSLOMBLD macro. Refer to “CSLOMBLD: Command Registration Build” on page 79 for more information.

CMDLISTLEN=*symbol*
CMDLISTLEN=*(r2-r12)*
(Required) - Specifies the length of the command definition list buffer.

ECB=*symbol*
ECB=*(r2-r12)*
(Optional) - Specifies the address of a z/OS event control block (ECB) used for asynchronous requests. When the request is complete, the ECB specified is posted. If an ECB is not specified, the task is suspended until the request is complete. If an ECB is specified, the invoker of the macro must issue a WAIT (or equivalent) after receiving control from CSLOMREG before using or examining any data returned by this macro (including the RETCODE and RSNCODE fields).

OMNAME=*symbol*
OMNAME=*(r2-r12)*
(Required) - Specifies the 8-byte OM name to which to send the command registration request.

OUTLEN=*symbol*

OUTLEN=*(r2-r12)*
(Optional) - Specifies a 4-byte field to receive the length of the output returned by the CSLOMREG request. OUTLEN contains the length of the output pointed to by the OUTPUT= parameter.

The output length is zero if no output is built, for example, if an error is detected before any output can be built.

OUTPUT=*symbol*
OUTPUT=*(r2-r12)*
(Required) - Specifies a field to receive the variable length output returned by the CSLOMREG request. The output length is returned in the OUTLEN= field.

The output is mapped by the CSLOREGO macro and is built only if there was an error registering one or more commands. The output contains a header and one or more list entries. Refer to the CSLOREGO macro for the output fields.

The output address is zero if no output was built, for example, if an error was detected before any output could be built.

The output buffer is not preallocated by the caller. After the request returns it, this word contains the address of a buffer containing the update output. It is the caller's responsibility to release this storage by issuing the `CSLSCBFR FUNC=RELEASE` request when it is finished with the storage. The length of the output is returned in the OUTLEN= field.

PARM=*symbol*
PARM=*(r1-r12)*
(Required) - Four-byte input parameter that specifies the address of the storage used by the request to pass the parameters to SCI. The length of the parameter list must be equal to the parameter list length EQU value defined by OREG_PARMLN.

RETCODE=*symbol*
RETCODE=*(r2-r12)*
(Required) - Specifies a 4-byte field to receive the return code on output. OM return codes are defined in CSLORR. SCI return codes are defined in CSLSRR. Possible return codes are described in Table 29 on page 88.

The return code can be from OM (CSLOMREG) or SCI (CSLSCMSG or CSLSCRQS). If ECB is specified, the RETCODE is not valid until the ECB is posted. All return codes contain the SCI member type indicator for either SCI, OM, or RM in the high order byte (X'01' for SCI, X'02' for OM, X'03' for RM).

RSNCODE=*symbol*
RSNCODE=*(r2-r12)*
(Required) - Specifies a 4-byte field to receive the reason code on output. OM reason codes are defined in CSLORR. SCI reason codes are defined in CSLSRR. Possible reason codes are described in Table 29 on page 88.

SCITOKEN=*symbol*
SCITOKEN=*(r2-r12)*
(Required) - Specifies a 16-byte field containing the SCI token. This token uniquely identifies this connection to SCI. The SCI token is returned by a successful `CSLSCREG FUNC=REGISTER` request.

CLSOMREG Return and Reason Codes

The return and reason codes in Table 29 on page 88 can be returned on a CSLOMREG macro request. Completion codes from CSLOMREG are in Table 30 on page 88.

Table 29. CLSOMREG Return and Reason Codes

Return Code	Reason Code	Meaning
X'00000000'	X'00000000'	The request completed successfully.
X'0200000C'	X'00003000'	The request might or might not have been processed completely. If the OUTPUT parameter is provided on the request, refer to completion codes in the output buffer for error conditions. Completion codes indicate the reason for the error with the resource name. The completion codes that can be returned are described in Table 30.
X'02000010'	X'00004010'	The client that issued the command is not authorized.
	X'00004020'	The parameter list version is invalid.
X'02000014'	X'00005034'	An OM internal error occurred. OM was unable to obtain storage for the output buffer.
	X'00005038'	An OM internal error occurred. OM was unable to add the VERB block to the command verb hash table during command processing.
	X'0000503C'	An OM internal error occurred. OM was unable to allocate a CLNT block for the client during command processing.

The completion codes in Table 30 can be returned on a CSLOMREG request. They are returned in the ORGE_CC field of the CSLOREGO macro, which maps the OUTPUT= area if an error occurred during command registration.

Table 30. CLSOMREG Completion Codes

Completion Code	Meaning
X'00000104'	OM was unable to allocate a VERB block for the resource.
X'00000108'	OM was unable to allocate a KWD block for the resource.
X'0000010C'	OM was unable to allocate an MUID block for the resource.
X'00000160'	OM was unable to obtain a latch for the resource.

CSLOMRSP: Command Response Request

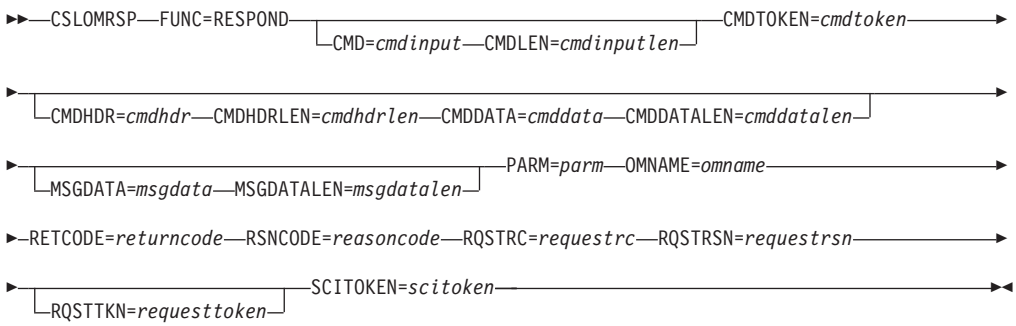
A command processing client issues the CSLOMRSP request in response to a command. All command response information from an individual command processing client is consolidated by the client and sent to OM in one request. OM consolidates the responses from multiple clients into one response for the automated operator program client.

CSLOMRSP Syntax

The syntax examples for the CSLOMRSP request follow.

CSLOMRSP DSECT Syntax: Use the DSECT function of a CSLOMRSP request to include equate (EQU) statements in your program for the CSLOMRSP parameter list length and return and reason codes.

CSLOMRSP Request Protocol Syntax:



CSLOMRSP Parameters

The parameters for the CSLOMRSP request follow.

CMD=*symbol*
CMD=*(r2-r12)*

(Optional) - Specifies the command input buffer. This can be any IMS command that can be specified through the OM API.

This parameter is optional; what you provide here will be included in the input tags that are returned as XML output. See Appendix A, “CSL Operations Manager XML Output,” on page 203 for additional information.

CMDDATA=*symbol*
CMDDATA=*(r2-r12)*
(Optional) - Specifies the command response data buffer.

CMDDATALEN=*symbol*
CMDDATALEN=*(r2-r12)*
(Optional) - Specifies the length of the command response data buffer.

CMDHDR=*symbol*
CMDHDR=*(r2-r12)*
(Optional) - Specifies the command response header buffer.

CMDHDRLEN=*symbol*
CMDHDRLEN=*(r2-r12)*
(Optional) - Specifies the length of the command response header buffer.

CMDLEN=*symbol*
CMDLEN=*(r2-r12)*
(Optional) - Specifies the length of the command input buffer.

CMDTOKEN=*symbol*
CMDTOKEN=*(r2-r12)*
(Required) - Specifies a 32-byte field to contain the command token. This token uniquely identifies the command instance that the client has processed. The command token is passed to the client on an OM command directive. The address of the token is passed to the client in the ODIR_CMDTKPTR field in the OM command directive parameter list.

MSGDATA=*symbol*
MSGDATA=*(r2-r12)*
(Optional) - Specifies the command response message buffer.

MSGDATALEN=*symbol*
MSGDATALEN=*(r2-r12)*
(Optional) - Specifies the length of the command response message buffer.

PARM=*symbol*
PARM=*(r1-r12)*
(Required) - Four-byte input parameter that specifies the address of the storage used by the request to pass the parameters to SCI. The length of the parameter list must be equal to the parameter list length EQU value defined by ORSP_PARMLN.

OMNAME=*symbol*
OMNAME=*(r2-r12)*
(Required) - Specifies the 8-byte OM name to which to send the command registration request.

RETCODE=*symbol*
RETCODE=*(r2-r12)*
(Required) - Specifies a 4-byte field to receive the return code on output. OM return codes are defined in CSLORR. SCI return codes are defined in CSLSRR. Possible return codes are described in Table 31 on page 91.

The return code can be from OM (CSLOMRSP) or SCI (CSLSCMSG or CSLSCRQS). If ECB is specified, the RETCODE is not valid until the ECB is posted. All return codes contain the SCI member type indicator for either SCI, OM, or RM in the high order byte (X'01' for SCI, X'02' for OM, X'03' for RM).

RQSTRC=*symbol*
RQSTRC=*(r2-r12)*
(Required) - Specifies a 4-byte field to contain the reason code to be passed to the originator of the command. This reason code is defined by the command processing client and indicates the result of the command. Non-zero reason codes are passed back to the client in the <cmderr> section of the command response.

RQSTRSN=*symbol*
RQSTRSN=*(r2-r12)*
(Required) - Specifies a 4-byte field to contain the reason code to be passed to the originator of the command. This reason code is defined by the command processor client and indicates the result of the command. Non-zero reason codes are passed back to the client in the <cmderr> section of the command response.

RQSTTKN=*symbol*
RQSTTNK=*(r2-r12)*
(Optional) - Specifies the 32-byte request token that was passed to the command processing client on an OM command directive. This parameter represents the RQSTTKN1 and RQSTTKN2 fields that are entered on either or both the CSLOMI and CSLOMCMD requests.

RSNCODE=*symbol*
RSNCODE=*(r2-r12)*
(Required) - Specifies a 4-byte field to receive the reason code on output. OM reason codes are defined in CSLORR. SCI reason codes are defined in CSLSRR. Possible reason codes are described in Table 31 on page 91.

SCITOKEN=*symbol*
SCITOKEN=*(r2-r12)*
(Required) - Specifies a 16-byte field containing the SCI token. This token uniquely identifies this connection to SCI. The SCI token is returned by a successful `CSLSCREG FUNC=REGISTER` request.

CSLOMRSP Return and Reason Codes

Table 31 lists the return and reason codes that can be returned on a CSLOMRSP macro request. Also included is the meaning of the reason code (that is, what possibly caused it).

Table 31. CLSOMRSP Return and Reason Codes

Return Code	Reason Code	Meaning
X'00000000'	X'00000000'	The request completed successfully.

CSL OM Automated Operator Program Clients

OM provides an application programming interface (API) for application programs that automate operator actions. These programs are called automated operator programs (AOP). An AOP issues commands that are embedded in an OM API request to an OM. The responses to those commands are returned to the AOP embedded in XML tags. See “CSL OM XML Output” on page 93 for more information on XML output.

If you want to use OM to manage commands and command responses in an IMSplex for your own product or service, you can use an AOP client, such as:

- The IMS-supplied AOP client, TSO single point of control (SPOC), which runs on the host. With the TSO SPOC, an automated operator can issue commands to the IMSplex and receive responses to those commands interactively.
- An AOP client that runs on a workstation (called a workstation SPOC).
- The IMS Control Center, an IMS system management application that performs SPOC functions.
- A command processing client, such as IMS.

Each of these clients is described in this topic.

An OM client uses OM requests to communicate with OM. Each OM client must register to SCI before it can issue OM requests.

If you intend to write AOPs, you can write them in either assembler or REXX. Assembler applications issue requests to the OM API; REXX applications issue REXX host commands to communicate with OM.

IMS provides a REXX SPOC API, which is a REXX program interface to a SPOC application. Your existing REXX applications can use this REXX SPOC API to interact with OM.

How AOP Clients that Run on the Host Communicate with the CSL OM

AOP clients that run on the host can communicate directly with OM. After a z/OS AOP is registered with SCI, it can issue OM command (CSLOMCMD) or query (CSLOMQRY) requests. When the z/OS AOP is ready to terminate, it must deregister with SCI using the CSLSCDRG macro. Each of the requests can be sent directly to OM or SCI. For more information on how to issue an SCI request, see “CSL SCI Requests” on page 171.

Table 32 on page 92 lists the sequence of requests that are issued from an AOP that is running on the host. The request is listed with its purpose.

Table 32. Sequence of requests for AOP running on the host

Request	Purpose
CSLSCREG	Registers to SCI, which enables the client to send OM requests to OM through SCI.
CSLSCRDY	Readies the OM client to SCI, which routes messages to the client by client type.
CSLOMxxx	Issues OM requests (CSLOMCMD, CSLOMQRY) to send commands to OM.
CSLSCBFR	Releases the output buffer returned by the request, if any.
CSLSCQSC	Quiesces with SCI.
CSLSCDRG	Deregisters from SCI.

Note: Although not required for an AOP executing on the host, CSLSCRDY and CSLSCQSC are recommended for clients that want to receive messages routed by TYPE.

An OM client uses OM requests to access and use OM services and resources. Some SCI and OM requests must be issued by the client to request OM services. Some of those requests must be issued in a particular sequence, as shown in Table 32. Other requests can be issued multiple times, in any order, based on the processing requirements of the client.

How AOP Clients that Run on a Workstation Communicate with the CSL OM

A workstation AOP client cannot communicate directly with OM. Instead, it must communicate with a z/OS address space that acts as an OM AOP client.

Instead of issuing CSLOMCMD or CSLOMQRY requests, the z/OS address space issues CSLOMI, which passes the prebuilt string that it received from the workstation to OM. For example, if the workstation wants to query the command processing clients to see how many exist in the IMSplex, it can send the string QUERY(CMDCLIENTS) to the z/OS address space, which would then use CSLOMI to send the query to OM for command processing.

If the workstation wants to issue a QRY TRAN command to the IMSplex, it can send the following string to the z/OS address space:

```
CMD(QUERY TRAN NAME) ROUTE(IMSA) TIMEOUT(10) RQSTTKN2(QTRANCMD)
```

The z/OS address space would then use CSLOMI to send the string to OM for command processing. The z/OS address space should pass the user ID associated with the workstation application to ensure correct authorization processing by OM.

Table 33 illustrates the sequence of requests issued from an AOP executing on z/OS and communicating directly with OM for the workstation. The request is listed with its purpose.

Table 33. Sequence of requests for AOP running on the workstation

Request	Purpose
CSLSCREG	Registers to SCI, which enables the client to send OM requests to OM through SCI.

Table 33. Sequence of requests for AOP running on the workstation (continued)

Request	Purpose
CSLSCRDY	Readies the OM client to SCI, which routes messages to the client by client type.
CSLOMI	Issues OM requests (CMD(), QUERY()) to send commands to OM.
CSLSCBFR	Releases the output buffer returned by the request, if any.
CSLSCQSC	Quiesces with SCI.
CSLSCDRG	Deregisters from SCI.

Note: Although not required for an AOP executing on the workstation, CSLSCRDY and CSLSCQSC are recommended for clients that want to receive messages routed by TYPE.

Command Processing Clients and the CSL OM

A command processing client, such as an IMS control region, is a system that provides a command processor to accept and process commands entered by an AOP.

A command processing client must register to OM in addition to registering with SCI. The command processing client registers with OM by passing a list of commands to OM that it can process.

After successful command registration, the client must inform OM that it is ready to process commands.

Like the AOP clients, command processing clients must issue requests in a particular sequence. This sequence, and the purpose of the request, is listed in Table 34.

Table 34. Sequence of requests for a command processing client

Request	Purpose
CSLSCREG	Registers to SCI, which enables the client to send OM requests to OM through SCI.
CSLSCRDY	Readies the OM client to SCI, which routes messages to the client by client type.
CSLOMREG	Registers the command list to OM.
CSLOMRDY	Readies OM client to OM. Client is now ready to process commands.
CSLOMRSP	Sends the command response output back to OM after receiving and processing a command from OM.
CSLOMDRG	Deregisters from OM. The client no longer wants to process commands.
CSLSCQSC	Quiesces with SCI.
CSLSCDRG	Deregisters from SCI.

CSL OM XML Output

Command responses that are returned through the OM API are embedded in XML tags. XML output is generated for responses to the CSLOMI, CSLOMCMD, and CSLOMQRY requests.

For example, with the CSLOMI request, the QUERY parameter allows you to query all clients that are registered to OM. The clients are returned embedded in <cmdclients></cmdclients> tags.

The list of XML tags and the descriptions of each tag are provided in Appendix A, “CSL Operations Manager XML Output,” on page 203.

CSL OM Directives

An OM directive is a function that OM defines that can be sent as a message to OM clients, informing the OM clients of work to be processed. Any command processing client that has registered commands to OM can be selected to perform an OM directive.

OM directives are always issued in message protocol (PROTOCOL=MSG), that is, asynchronously; OM therefore expects no response from the OM client, and it continues processing without waiting for a response. The OM client is responsible for determining whether or not to take any action in response to the directive.

When a client issues PROTOCOL=MSG, SCI sends the XML output from OM to the client’s SCI Input exit. The SCI Input exit routine’s INXP_MBRPLPTR field points to the CSLOMDIR parameter list. For more information on the SCI Input exit parameter list, see “CSL SCI Input Exit Parameter List” on page 164.

When a client issues CSLOMI PROTOCOL=RQST, the XML output stream from OM is sent directly to the client in the OUTPUT= parameter.

The SCI Input exit routine is responsible for notifying the client of the directive. The client should code their SCI Input exit routine to support OM directives. The client is responsible for determining where the function and function code are to be defined. After the client is finished using the CSLOMDIR parameter list, it must release the storage by issuing CSLSCBFR.

OM directives are defined in the CSLOMDIR macro, which includes:

- Command directive (ODIR_CMDD)
- CSLOMI response directive (ODIR_OMIRESPD)
- Command response directive (ODIR_CMDRESPD)
- Query response directive (ODIR_QRYRESPD)

The directives and their parameters are described in this topic.

CSL OM Command Directive

The OM command directive, ODIR_CMDD, is sent to a command processing client when a command is available for processing.

The parameters for the OM command directive follow. They are passed to the SCI Input Exit:

ODIR_COMMAND
Identifies the start of the command directive.

ODIR_CMDTKLEN=*length*
Contains the length of the OM command token. It is used only by OM to identify the command instance.

ODIR_CMDTKPTR=*address*
Contains the address of the OM command token.

ODIR_INPUTLEN=*length*
Contains the length of the command input string that you enter.

ODIR_INPUTPTR=*address*
Contains the address of the command input string.

ODIR_VERBLEN=*length*
Contains the length of the command verb in normalized form.

ODIR_VERBPTR=*address*
Contains the address of the command verb.

ODIR_KWDLEN=*length*
Contains the length of the command keyword.

ODIR_KWDPTR=*address*
Contains the address of the command keyword.

ODIR_PARSELEN=*length*
Contains the length of the parsed command block.

ODIR_PARSEPTR=*address*
Contains the address of the parsed command block.

ODIR_CUIDLEN=*length*
Contains the length of the user ID that originated the command.

ODIR_CUIDPTR=*address*
Contains the address of the user ID that originated the command.

ODIR_CNAMELEN=*length*
Contains the length of the name of the client that originated the command (that is, the name that was registered to SCI).

ODIR_CNAMEPTR=*address*
Contains the address of the name of the client that originated the command (that is, the name that was registered to SCI).

ODIR_CTYPE=*client type*
Contains the type of client that originated the command. This is the value from the TYPE= parameter as defined to SCI. This parameter is passed by value; the length field is always zero.

ODIR_CSTYPLEN=*length*
Contains the subtype of the client that originated the command. This is the value from the SUBTYPE= parameter as defined to SCI.

ODIR_CSTYPPTR=*address*
Contains the address of the subtype of the client that originated the command.

ODIR_CFLAGS=*flags*
Contains the OM command processing flags. These parameters are passed by value; the length field is always zero.

ODIR_CRQTKLEN=*length*
Contains the length of the user request token; this parameter is used only by the program that originated the command to identify the command instance.

ODIR_CRQTKPTR=*address*
Contains the address of the user request token; this parameter is used only by the program that originated the command to identify the command instance.

ODIR_TIMEOUT=*timeoutvalue*
Contains the command timeout value as specified on the command. This parameter is passed by value; the length field is always zero.

ODIR_CMDLN
The command directive length EQU.

CSL OM Response Directives

There are three response directives in CSLOMDIR:

- CSLOMI response (ODIR_OMIRESPD)

  The CSLOMI response directive returns a response to a client regarding a CSLOMI call. The response is sent when an error occurs and it is unclear if the response is for a CSLOMI CMD or CSLOMI QUERY call.
- Command response (ODIR_CMDRESPD)

  The command response directive returns a command response to a client that results from a CSLOMI CMD or CSLOMCMD call.
- Query response (ODIR_QRYRESPD)

  The query response directive returns a query response to a client that results from a CSLOMI QUERY or CSLOMQRY call.

The parameters for the OM response directives are identical.

ODIR_CQRESP
Identifies the start of the command or query response.

ODIR_CQRSPRC=*returncode*
Contains the return code of the command or query response.

ODIR_CQRSPRSN=*reasoncode*
Contains the reason code of the command or query response.

ODIR_CQXMLLEN=*length*
Contains the length of the XML output being returned.

ODIR_CQXMLPTR=*address*
Contains the address of the XML output being returned.

ODIR_CQRT1LEN=*length*
Contains the length of request token 1 (RQSTTKN1).

ODIR_CQRT1PTR=*address*
Contains the address of request token 1 (RQSTTKN1).

ODIR_CQRT2LEN=*length*
Contains the length of request token 2 (RQSTTKN2).

ODIR_CQRT2PTR=*address*
Contains the address of request token 2 (RQSTTKN2).

ODIR_CQRSPLN
The response directive length EQU.

Chapter 4. CSL Resource Manager

This topic describes the role of RM in a CSL:

Overview of the CSL Resource Manager

RM is an IMS address space that manages global resources and IMSplex-wide processes in a sysplex on behalf of its clients. IMS is one example of an RM client that uses RM to manage:

- global message destination and terminal resources
- global online change

Clients communicate with RM using RM requests that are described in this topic. Requests used to manage global resources include:

- "CSLRMDEL: Delete Resources" on page 113
- "CSLRMDRG: Deregister Clients" on page 117
- "CSLRMQRY: Query Resources" on page 131
- "CSLRMREG: Register Clients" on page 136
- "CSLRMUPD: Update Resources" on page 139

Requests used to manage IMSplex-wide processes in a sysplex include:

- "CSLRMPRI: Process Initiate" on page 118
- "CSLRMPRR: Process Respond" on page 121
- "CSLRMPRS: Process Step" on page 123
- "CSLRMPRT: Process Terminate" on page 129

Using these requests, RM clients can either communicate with RM and manipulate global resources, or participate in IMSplex-wide processes.

With RM, the system administrator can manage resources that are shared by multiple IMS systems in an IMSplex. RM provides an infrastructure for managing global resources and coordinating processes across the IMSplex.

RM uses Common Queue Server (CQS) to maintain global resource information in a *resource structure*, which is a coupling facility list structure that all CQSs in the IMSplex can access.

Recommendation: Although a resource structure is optional in an IMSplex, it is recommended that you define a resource structure for improved recovery capabilities.

RM also supports coordinated processes across the IMSplex. For example, IMS uses this support to coordinate global online change across the IMSplex. For this activity, OM is also required.

At least one RM must be defined in an IMSplex to use RM functions. You can have one or more RMs on each z/OS image if a resource structure is defined. If no resource structure is defined, you can have only one RM. If you do not require RM functions, you can configure your IMS system without an RM, specifying RMENV=N on the DFSCGxxx PROCLIB member. Any RM can process work from any z/OS image within an IMSplex. For more information on CSL configurations, see "CSL Configuration Examples" on page 7.

Maintaining Global Resource Information with the CSL RM

RM uses CQS to store global resource information on a coupling facility list structure that all RMs can access. This structure is called a *resource structure*. You can use RM to create, update, query, or delete resources in the resource structure. To display information about the resource structure managed by RM, use the QUERY STRUCTURE command. More information on this command is in *IMS Version 9: Command Reference*.

RM works with CQS to access the resource structure for the client. RM issues the requests to query and maintain the resources, and it notifies the client if there is a resource structure change that affects the client.

The IMSs in the IMSplex still contain the resource definitions. RM does not ensure resource definition consistency across the IMSplex. You can use global online change to update the resource definitions in the IMSs.

RM's Stored Resource Information

RM maintains the following information about resources on a resource structure:

Resource Name
A client-defined resource name that is 11 bytes in length.

Name Type
A resource attribute that ensures that all resources within the same type have unique names. RM enforces this rule. Resources within a name type can have different resource types.

Resource Type
A client-defined resource attribute that allows CQS to physically group resources on a coupling facility list structure. RM supports up to 255 resource types.

RM uses resource type 253 to store its global information. Resource type 253 hashes to the same physical list headers as resource types 11, 22, 33, 44,...253. A client can define resource types that map to the same physical list headers as RM's global information.

Resource Version
The number of times that the resource has been updated. RM uses the version to serialize updates on the resource structure.

Resource Owner
A resource attribute that signifies the owner of the resource. This attribute is optional.

If the owner is identified, the client is responsible for enforcing a single owner of a resource. For example, IMS can set owners for the following resources to enforce a single active instance of the resources: nodes, lterms, users, and user IDs. A user or user ID that is signed on to one IMS in the IMSplex cannot

be signed on to another IMS in the IMSplex at the same time. An lterm that is active on one IMS in the IMSplex cannot be active on another IMS in the IMSplex at the same time.

Resource Data
A resource attribute that contains any additional data about the resource.

Resource Structure Duplexing Requirements for CSL RM

CQS does not log resource updates or support a checkpoint of the resource structure. However, CQS supports automatic duplexing of the resource structure for backup in case of structure failure. You must define the resource structure in the coupling facility resource management (CFRM) policy to automatically be duplexed if you want the resource structure to be recoverable.

How the CSL RM Repopulates a Resource Structure

If the resource structure and its duplex (if applicable) fail, and CQS can allocate a new structure, CQS notifies RM to repopulate the structure. RM repopulates the structure from information in its local control blocks. RM then issues a directive to its clients to populate the structure. For information on RM directives, see "CSL RM Directives" on page 145.

If the resource structure fails, and CQS cannot allocate a new structure, CQS notifies RM that the structure failed. RM then issues a directive to its clients that the structure failed. Any RM or IMS resource that existed only on the resource structure is lost. When a new resource structure is allocated, the clients can choose to coordinate the repopulation of the resource structure.

How z/OS Rebuilds a Resource Structure

Resource structures are defined with system-managed rebuild, so z/OS automatically rebuilds the structure when no CQS is up to build the structure. z/OS cannot rebuild the structure if the structure fails or if z/OS loses connectivity to the structure. If any CQS is up and rebuild is initiated with the `SETXCF START,REBUILD` command, then CQS copies the structure. If the structure fails, no structure recovery is initiated because resource structures do not support structure checkpoint.

CSL RM Definition and Tailoring

This topic describes the how to define and tailor RM in an IMSplex. You can tailor the following procedures:

- "CSL RM Startup Procedure"
- "CSL RM Execution Parameters" on page 100
- "CSL RM Initialization Parameters PROCLIB Member" on page 101
- "BPE Considerations for the CSL RM" on page 103

You can also use the BPE user exit list PROCLIB member to define the BPE user exit routines to include in RM. See *IMS Version 9: Base Primitive Environment Guide and Reference* for information on the BPE user exit list PROCLIB member.

CSL RM Startup Procedure

Use the RM startup procedure to dynamically override the settings in the RM initialization parameters PROCLIB member. You can start RM as a started

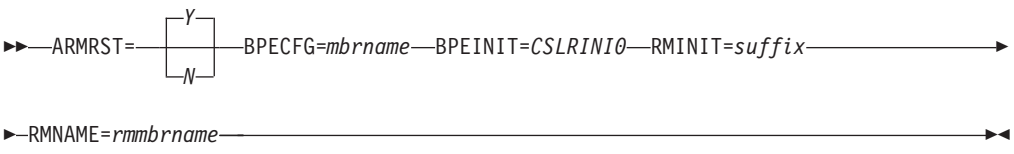
procedure or with JCL. A sample startup procedure, shown in Figure 18, is called CSLRM and can be found in IMS.PROCLIB.

```
//*****************************************************************
//*     RM Procedure
//*
//*
//*     Parameters:
//*     BPECFG  - Name of BPE member
//*     RMINIT  - Suffix for your CSLRIxxx member
//*     PARM1   - other override parameters:
//*               ARMRST - Indicates if ARM should be used
//*               RMNAME - Name of RM being started
//*
//*               example:
//*               PARM1='ARMRST=Y,RMNAME=RM1'
//*
//*************************************************************@SCPYRT**
//*                                                                   *
//*       Licensed Materials - Property of IBM                        *
//*                                                                   *
//*       "Restricted Materials of IBM"                               *
//*                                                                   *
//*       5655-J38 (C) Copyright IBM Corp. 2003                       *
//*                                                                   *
//*************************************************************@ECPYRT**
//*
//CSLRM    PROC RGN=3000K,SOUT=A,
//             RESLIB='IMS.SDFSRESL',
//             BPECFG=BPECONFG,
//             RMINIT=000,
//             PARM1=
//*
//RMPROC   EXEC PGM=BPEINI00,REGION=&RGN,
//  PARM='BPECFG=&BPECFG,BPEINIT=CSLRINI0,RMINIT=&RMINIT,&PARM1'
//*
//STEPLIB  DD  DSN=&RESLIB,DISP=SHR
//         DD  DSN=SYS1.CSSLIB,DISP=SHR
//PROCLIB  DD  DSN=IMS.PROCLIB,DISP=SHR
//SYSPRINT DD  SYSOUT=&SOUT
//SYSUDUMP DD  SYSOUT=&SOUT
//*
```

Figure 18. Sample Resource Manager Startup Procedure

CSL RM Execution Parameters

This topic describes the parameters that can be specified as execution parameters on the RM startup procedure. Some parameters that are required for RM initialization can also be specified in the RM initialization parameters PROCLIB member.



ARMRST= Y | N
Specifies whether or not the z/OS Automatic Restart Manager (ARM) should be used to restart RM after an abend. **Y** (yes) specifies that ARM should be used.

The RM address space is restarted by ARM after most system failures. **N** (no) specifies that ARM should not be used. RM is not restarted by ARM after any failures.

This is an optional execution parameter. If specified, it overrides the value specified in the CSLRIxxx PROCLIB member. If not specified, the value in the CSLRIxxx PROCLIB member is used.

For information on ARM, see "Using the z/OS Automatic Restart Manager with the CSL" on page 29.

BPECFG=
Specifies an 8-character name for the BPE configuration parameters PROCLIB member. This parameter can be specified only as an execution parameter. The parameter is optional. If it is not specified, the BPE defaults are used:

- No BPE user exits
- A BPE trace level of ERROR
- The language used for BPE messages is US English

For more information on this parameter, see *IMS Version 9: Base Primitive Environment Guide and Reference*.

BPEINIT=CSLRINI0
Specifies the name of the module that contains the RM start up values required by BPEINI00 to start an RM address space. For RM, this value must be CSLRINI0. This parameter can be specified only as an execution parameter. This is a required parameter.

RMINIT=
Specifies a 3-character suffix for the RM initialization parameters PROCLIB member, CSLRIxxx. This parameter can be specified only as an execution parameter. The default suffix is 000.

RMNAME=*rmmbrname*
Specifies the name for the RM address space. This is an optional 1-6 character name. If specified, it overrides the value specified in the CSLRIxxx PROCLIB member. You must specify this parameter either as an execution parameter or in the CSLRIxxx PROCLIB member. This name is used to create the RMID, which is used in RM processing. The 8-character RMID is the RMNAME followed by the characters "RM". Trailing blanks in the RMNAME are deleted, and the RMID is padded with blanks. For example, if RMNAME=ABC then RMID="ABCRM ".

CSL RM Initialization Parameters PROCLIB Member

Use the CSLRIxxx PROCLIB member to specify parameters that initialize RM. Some parameters within CSLRIxxx can be overridden with RM execution parameters.

The CSLRIxxx PROCLIB member consists of one or more fixed-length character records (the configuration data set can be of any LRECL greater than eight, but it must be fixed record format). The rightmost-eight columns are ignored but can be used for sequence numbers or any other notation. Keyword parameters can be coded in the remaining columns in free format, and can contain leading and trailing blanks. You can specify multiple keywords in each record; use commas or spaces to delimit keywords. Statements that begin with a "*" or "#" in column 1 are comment lines and are ignored. Additionally, comments can be included anywhere within a statement by enclosing them between " /* "and "*/", for example, /*

PROCLIB comments */. Values coded in this PROCLIB member are case-sensitive. In general, you should use upper case for all parameters.

ARMRST= Y | N
Specifies whether or not the z/OS Automatic Restart Manager (ARM) should be used to restart RM after an abend. **Y** (yes) specifies that ARM should be used. The RM address space is restarted by ARM after most system failures. **N** (no) specifies that ARM should not be used. RM is not restarted by ARM after any failures. For information on ARM, see "Using the z/OS Automatic Restart Manager with the CSL" on page 29.

CQSSSN=
Specifies the one- to four-character subsystem name of the CQS. RM uses this name to connect to the proper CQS. When connecting RM to CQS, you must specify the same value on CQSSSN= and on the SSN= parameter of the CQSIPxxx PROCLIB member for the target CQS. The parameter is optional, and no default exists. Both CQSSSN and RSRCSTRUCTURE must be specified together, or neither must be specified. CQSSSN and RSRCSTRUCTURE must be specified to make use of RM's global resource services.

IMSPLEX()
Specifies definitions for an IMSplex managed by RM. IMSPLEX is a required parameter. There is no default. Only one IMSPLEX keyword can be specified. The IMSPLEX keyword must precede the left parenthesis. The IMSPLEX definition parameters follow:

NAME=
Specifies a 1-5 character identifier that specifies the IMSplex group name. This defines the IMSplex to which the resource structure is defined. NAME is required and no default exists. RM concatenates this identifier to "CSL" to create the IMSplex group name. All OM, RM, SCI, IMS, and IMSplex members that are in the same IMSplex sharing group sharing either databases or message queues must specify the same identifier. The same identifier must also be used for the IMSPLEX= parameter in the CSLSIxxx, CSLOIxxx and DFSCGxxx PROCLIB members.

RSRCSTRUCTURE()
Specifies definitions for a resource structure managed by RM. This keyword construct is optional. RSRCSTRUCTURE must be specified to make use of RM's global resource services. Only one resource structure can be defined. The resource structure definitions must be enclosed within parentheses and separated by commas. The RSRCSTRUCTURE keyword must precede the left parenthesis.

STRNAME=
Specifies the 1- to 16-character name of a resource structure that IMS connects to, which contains IMS resource information. If the RSRSTRUCTURE construct is specified, then STRNAME is required within the RSRCSTRUCTURE construct.

The installation must have defined the structure name in the CFRM administrative policy. The structure name must follow the naming rules as allowed by the CFRM. The structure name must be from 1 to 16 characters long. For names with less than 16 characters, CQS pads the name with blanks. The valid characters are A-Z, 0-9 and the characters $, &, # or _. Names must be uppercase and start with an alphabetic character. To

avoid using names IBM uses for its structures, do not begin structure names with the letters A-I, or the character string SYS. This resource structure must also be defined in the CQS global structure definition PROCLIB member (CQSSGxxx) of the CQS in the same IMSplex sharing group. This resource structure must also be defined in the CFRM policy.

RMNAME=*rmmbrname*
Specifies the name for the RM address space. This is an optional 1- to 6-character name. You must specify this parameter either as an execution parameter or in the CSLRIxxx PROCLIB member. This name is used to create the RMID, which is used in RM processing. The 8-character RMID is the RMNAME followed by the characters “RM”. Trailing blanks in the RMNAME are deleted, and the RMID is padded with blanks. For example, if RMNAME=ABC then RMID=“ABCRM ”.

A sample CSLRIxxx PROCLIB member is shown in Figure 19.

```
*-------------------------------------------------------------------*
* Sample RM Initialization PROCLIB Member.                          *
*-------------------------------------------------------------------*
ARMRST=Y,                      /* ARM should restart RM on failure  */
CQSSSN=CQS1,                   /* CQS to manage Resource Structure  */
IMSPLEX(
  NAME=PLEX1,                  /* IMSplex name (CSLPLEX1)           */
  RSRCSTRUCTURE(
    STRNAME=IMSRSRC01)),       /* RESOURCE STRUCTURE NAME           */
RMNAME=RM1                     /* RM Name (RMID = RM1RM)            */
*-------------------------------------------------------------------*
* End of Member CSLRI000                                            *
*-------------------------------------------------------------------*
```

Figure 19. CSLRIxxx PROCLIB Member

BPE Considerations for the CSL RM

Use the RM BPE user exit list PROCLIB member to define RM user exits to BPE. The member is the PROCLIB member specified by the EXITMBR= parameter in the BPE configuration parameter PROCLIB member.

Use the user exit list PROCLIB member to specify the modules to be called for specific exit types. Each user exit type can have one or more exit modules associated with it. Use the EXITDEF statement to define the user exit modules to be called for a given exit type.

The BPE user exit PROCLIB member and BPE configuration PROCLIB member are described in *IMS Version 9: Base Primitive Environment Guide and Reference*.

A sample RM user exit list PROCLIB member is shown in Figure 20 on page 104.

```
*********************************************************************
* RM USER EXIT LIST PROCLIB MEMBER                                  *
*********************************************************************

#-------------------------------------------------------------------#
# DEFINE 1 RM CLIENT CONNECTION USER EXIT: ZRCLNCN0                 #
# WITH AN ABEND LIMIT OF 8.                                         #
#-------------------------------------------------------------------#
EXITDEF(TYPE=CLNTCONN,EXITS=(ZRCLNCN0),ABLIM=8,COMP=RM)

#-------------------------------------------------------------------#
# DEFINE 1 RM INIT/TERM USER EXIT: ZRINTM00                         #
#-------------------------------------------------------------------#
EXITDEF(TYPE=INITTERM,EXITS=(ZRINTM00),COMP=RM)
```

Figure 20. RM User Exit PROCLIB Member

CSL RM Administration

The tasks associated with administering RM are typically performed by the system administrator or system operator. The tasks include:

- "Starting the CSL RM"
- "Shutting Down the CSL RM"

Starting the CSL RM

The system operator can start RM in two ways:

- As a started procedure.
- As JCL.

To start an RM address space with a started procedure, issue the z/OS `START` command as follows:

```
S rmjobname
```

In this example, `rmjobname` is the job name of the RM address space to be started. For more information on the initialization parameters used with the RM startup procedures, see "CSL RM Startup Procedure" on page 99.

After RM is started, if it is abnormally terminated, it can be restarted using the z/OS Automatic Restart Manager (ARM). RM must complete initialization for ARM to restart the address space if an abend occurs. Use of ARM to restart RM is the default.

Shutting Down the CSL RM

Recommendation: Although you can shut down RM by itself, IBM recommends that you shut down RM by shutting down the CSL as one unit. For information about shutting down the CSL, see "Shutting Down the CSL" on page 26.

To shut down RM by itself, issue one of the following:

- The CSLZSHUT request, described in "CSLZSHUT: Shut Down Request" on page 26
- The z/OS STOP command:

  ```
  P rmjobname
  ```

In this example, `rmjobname` is the job name of the RM address space to stop. If no clients are connected to RM, RM shuts down. If clients are connected to RM,

message CSL0300I is issued, and RM quiesces in-flight work. After all work is quiesced, the RM address space terminates.

Before shutting down an RM, consider the reasons for shutting down and how shutting down OM can impact other IMSplex members. For more information, see "Shutting Down the CSL" on page 26.

CSL RM User Exit Routines

You can write RM user exits to customize and monitor the RM environment. No sample exits are provided.

RM uses BPE services to call and manage its user exits. BPE enables you to externally specify the user exit modules to be called for a particular user exit type by using EXITDEF= statements in the BPE user exit list PROCLIB members. BPE also provides a common user exit runtime environment for all user exits. This environment includes a standard user exit parameter list, callable services, static and dynamic work areas for the exits, and a recovery environment for user exit abends. For more information about the BPE user exit interface, see the *IMS Version 9: Base Primitive Environment Guide and Reference*.

CSL RM Client Connection User Exit

This exit is called when a client connects (registers) to RM or disconnects (deregisters) from RM. This exit is optional.

This exit is called for the following events:

- After a client has successfully connected to RM.
- After a client has successfully disconnected normally or abnormally from RM.

This exit is defined as TYPE=CLNTCONN in the EXITDEF statement in the BPE user exit list PROCLIB member. You can specify one or more user exits of this type. When this exit is invoked, all user exits of this type are called in the order specified by the EXITS= keyword. For more information on how to define user exit module names, see the RM BPE user exit List PROCLIB member information in *IMS Version 9: Base Primitive Environment Guide and Reference*.

This exit is invoked amode 31 and should be reentrant.

Contents of Registers on Entry

Register	Contents
1	Address of BPE user exit parameter list (mapped by macro BPEUXPL).
13	Address of the first of 2 prechained 72-byte save areas. These save areas are chained according to standard z/OS save area linkage convention. The first save area can be used by the exit to save registers on entry. The second save area is for use by routines called from the user exit.
14	Return address.
15	Entry point of exit routine.

On entry to the Client Connection exit, register 1 points to a standard BPE user exit parameter list. Field UXPL_EXITPLP in this list contains the address of the RM Client Connection user exit parameter list, which is mapped by macro CSLRCLX. Field UXPL_COMPTYPEP in this list points to the character string "RM" indicating an RM address space.

RM Client Connection User Exit Parameter List--Client Connect: Table 35 lists the user exit parameter list for client connect. Included are the field name, the offset value and length, both in hexadecimal, how the field is used, and a brief description of the field.

Table 35. RM Client Connection User Exit Parameter List--Client Connect

Field Name	Offset	Length	Field Usage	Description
RCLX_PVER	X'00'	X'04'	Input	Parameter list version number (00000001).
RCLX_FUNC	X'04'	X'04'	Input	Function code **1** Client Connect.
RCLX_MBRNAME	X'08'	X'08'	Input	Client (IMSplex member) name.
RCLX_MBRTYPE	X'10'	X'02'	Input	IMSplex member type (mapped by CSLSTPIX).
	X'12'	X'02'	None	Reserved.
RCLX_MBRSTYPE	X'14'	X'08'	Input	IMSplex member subtype
	X'1C'	X'04'	None	Reserved.

RM Client Connection User Exit Parameter List--Client Disconnect: Table 36 lists the user exit parameter list for client disconnect. Included are the field name, the offset value and length, both in hexadecimal, how the field is used, and a brief description of the field.

Table 36. RM Client Connection User Exit Parameter List--Client Disconnect

Field Name	Offset	Length	Field Usage	Description
RCLX_PVER	X'00'	X'04'	Input	Parameter list version number (00000001).
RCLX_FUNC	X'04'	X'04'	Input	Function code **2** Client Disconnect.
RCLX_MBRNAME	X'08'	X'08'	Input	Client (IMSplex member) name.
RCLX_MBRTYPE	X'10'	X'02'	Input	IMSplex member type (mapped by CSLSTPIX).
RCLX_FLAG1	X'12'	X'01'	Input	Flag byte indicates whether the client disconnect is normal or abnormal. **X'80'** Client disconnect is abnormal.
	X'13'	X'01'	None	Reserved.
RCLX_MBRSTYPE	X'14'	X'08'	Input	IMSplex member subtype
	X'1C'	X'08'	None	Reserved.

Contents of Registers on Exit

Register	Contents	
15	**Return Code**	**Meaning**
	0	Always zero

All other registers must be restored.

CSL RM Initialization/Termination User Exit

This exit is called for the following events:

- After RM has completed initialization
- After each IMSplex has initialized
- When RM is terminating normally
- When an IMSplex is terminating normally

This exit is not called during RM address space abnormal termination or IMSplex abnormal termination. This exit is optional.

This exit is defined as TYPE=INITTERM in the EXITDEF statement in the BPE user exit list PROCLIB member. You can specify one or more user exits of this type. When this exit is invoked, all user exits of this type are called in the order specified by the EXITS= keyword. For more information on how to define user exit module names, see the RM BPE user exit List PROCLIB member information in *IMS Version 9: Base Primitive Environment Guide and Reference*.

This exit is invoked amode 31 and should be reentrant.

Contents of Registers on Entry

Register	Contents
1	Address of BPE user exit parameter list (mapped by macro BPEUXPL).
13	Address of the first of 2 prechained 72-byte save areas. These save areas are chained according to standard z/OS save area linkage convention. The first save area can be used by the exit to save registers on entry. The second save area is for use by routines called from the user exit.
14	Return address.
15	Entry point of exit routine.

On entry to the Initialization/Termination exit, register 1 points to a standard BPE user exit parameter list. Field UXPL_EXITPLP in this list contains the address of the RM Initialization/Termination user exit parameter list, which is mapped by macro CSLRITX. Field UXPL_COMPTYPEP in this list points to the character string “RM,” indicating an RM address space.

RM Init/Term User Exit Parameter List--RM Initialization: Table 37 lists the user exit parameter list for RM initialization. Included are the field name, the offset value and length, both in hexadecimal, how the field is used, and a brief description of the field.

Table 37. RM Init/Term User Exit Parameter List--RM Initialization

Field Name	Offset	Length	Field Usage	Description
RITX_PVER	X’00’	X’04’	Input	Parameter list version number (00000001).
RITX_FUNC	X’04’	X’04’	Input	Function code: **1** RM Initialization

RM Init/Term User Exit Parameter List--RM Termination: Table 38 on page 108 lists the user exit parameter list for RM termination. Included are the field name, offset value and length, both in hexadecimal, how the field is used, and a brief description of the field.

Table 38. RM Init/Term User Exit Parameter List--RM Termination

Field Name	Offset	Length	Field Usage	Description
RITX_PVER	X'00'	X'04'	Input	Parameter list version number (00000001).
RITX_FTERM	X'04'	X'04'	Input	Function code **2** RM normal termination.

RM Init/Term User Exit Parameter List--IMSplex Initialization: Table 39 lists the user exit parameter list for IMSplex initialization. Included are the field name, the offset value and length, both in hexadecimal, how the field is used, and a brief description of the field.

Table 39. RM Init/Term User Exit Parameter List--IMSplex Initialization

Field Name	Offset	Length	Field Usage	Description
RITX_PVER	X'00'	X'04'	Input	Parameter list version number (00000001).
RITX_FPLXINIT	X'04'	X'04'	Input	Function code **3** IMSplex normal initialization.
RITX_IPLEXNM	X'08'	X'08'	Input	IMSplex name.
RITX_ISTRNM	X'10'	X'10'	Input	Resource structure name.

RM Init/Term User Exit Parameter List--IMSplex Termination: Table 40 lists the user exit parameter list for IMSplex termination. Included are the field name, the offset value and length, both in hexadecimal, how the field is used, and a brief description of the field.

Table 40. RM Init/Term User Exit Parameter List--IMSplex Termination

Field name	Offset	Length	Field Usage	Description
RITX_PVER	X'00'	X'04'	Input	Parameter list version number (00000001).
RITX_FUNC	X'04'	X'04'	Input	Function code **4** IMSplex normal termination.
RITX_TPLEXNM	X'08'	X'08'	Input	IMSplex name.
RITX_TSTRNM	X'10'	X'10'	Input	Resource structure name.

Contents of Registers on Exit

Register	Contents	
15	**Return Code**	**Meaning**
	0	Always zero

All other registers must be restored.

CSL RM Statistics Available through BPE Statistics User Exit

The BPE Statistics user exit can be used to gather both BPE and RM statistics. Refer to the BPE user exit information in *IMS Version 9: Base Primitive Environment Guide and Reference* for details on the exit and when it is called.

The following describes the RM statistics that are available to the BPE Statistics user exit and are returned on a CSLZQRY FUNC=STATS request directed to the RM address space. When the user exit is called, field BPESTXP_COMPSTATS_PTR in the BPE Statistics user exit parameter list, BPESTXP, contains the pointer to the RM statistics header. When the CSLZQRY FUNC=STATS request is called, the OUTPUT= buffer points to the output area mapped by CSLZQRYO. The output area field ZQYO_STXOFF contains the offset to the RM statistics header. The header is mapped by CSLRSTX.

CSL RM Statistics Header

Table 41 lists the RM statistics header. Included are the offset value and length, both in hexadecimal, how the field is used, and a brief description of the field.

Table 41. RM Statistics Header

Field Name	Offset	Length	Field Usage	Description
RSTX_ID	X'00'	X'08'	Input	Eyecatcher "CSLRSTX".
RSTX_LEN	X'08'	X'04'	Input	Length of header.
RSTX_PVER	X'0C'	X'04'	Input	Header version number (0000001).
RSTX_PLEXCNT	X'10'	X'04'	Input	Number of IMSplexes for which statistics are available.
RSTX_STATCNT	X'14'	X'04'	Input	Number of statistics areas available for each IMSplex.
RSTX_STATLEN	X'18'	X'04'	Input	Length of all statistics areas for each IMSplex.
RSTX_STATOFF	X'1C'	X'04'	Input	Offset to statistics area for first IMSplex. This is the offset from the beginning of CSLRSTX. The offset points to the CSLRST1 area.
RSTX_RST1OFF	X'20'	X'04'	Input	Offset to the RM request statistics record for activity performed by RM requests (mapped by macro CSLRST1). The offset is from the start of the statistics area for this IMSplex. Refer to Table 42 for a description of the RM request statistics record.
RSTX_RST2OFF	X'24'	X'04'	Input	Offset to RM IMSplex statistics record for activity performed by RM for an IMSplex (mapped by macro CSLRST2). The offset is from the start of the statistics area for this IMSplex. Refer to Table 43 on page 110 for a description of the RM IMSplex statistics record.
	X'28'	X'04'	None	Reserved.
	X'2C'	X'04'	None	Reserved.

CSL RM Statistics Record CSLRST1

CSLRST1 contains statistics that are related to specific requests processed by RM. Table 42 lists the RM statistics record CSLRST1. Included are the offset value and length, both in hexadecimal, how the field is used, and a brief description of the field.

Table 42. RM Statistics Record CSLRST1

Field Name	Offset	Length	Field Usage	Description
RST1_ID	X'00'	X'08'	Input	Eyecatcher "CSLRST1".
RST1_LEN	X'08'	X'04'	Input	Length of valid data.
RST1_PVER	X'0C'	X'04'	Input	Parameter list version number (00000001).

Table 42. RM Statistics Record CSLRST1 (continued)

Field Name	Offset	Length	Field Usage	Description
RST1_RMUPD	X'10'	X'04'	Input	Number of CSLRMUPD FUNC=UPDATE requests.
RST1_RMQRY	X'14'	X'04'	Input	Number of CSLRMQRY FUNC=QUERY requests.
RST1_RMDEL	X'18'	X'04'	Input	Number of CSLRMDEL FUNC=DELETE requests.
RST1_RMREG	X'20'	X'04'	Input	Number of CSLRMREG FUNC=REGISTER requests.
RST1_RMDRG	X'24'	X'04'	Input	Number of CSLRMDRG FUNC=DEREGISTER requests.
RST1_RMDRGIN	X'28'	X'04'	Input	Number of internal deregister requests for client normal termination.
RST1_RMDRGIA	X'2C'	X'04'	Input	Number of internal deregister requests for client abnormal termination.
	X'30'	X'10'	Input	Not used.
RST1_RMPRCI	X'40'	X'04'	Input	Number of CSLRMPRI FUNC=INITIATE initiate IMSplex-wide process requests.
RST1_RMPRCT	X'44'	X'04'	Input	Number of CSLRMPRT FUNC=TERMINATE terminate IMSplex-wide process requests.
RST1_RMPRCS	X'48'	X'04'	Input	Number of CSLRMPRS FUNC=PROCESS IMSplex-wide step requests.
RST1_RMPRCR	X'4C'	X'04'	Input	Number of CSLRMPRR FUNC=RESPOND IMSplex-wide step response requests.
RST1_ZQRY	X'50'	X'04'	Input	Number of CSLZQRY requests.
	X'54'	X'04'	Input	Number of BPESTATS requests.

CSL RM Statistics Record CSLRST2

CSLRST2 contains statistics that are related to an IMSplex, but not to specific requests. Table 43 lists the RM statistics record CSLRST2. Included are the offset value and length, both in hexadecimal, how the field is used, and a brief description of the field.

Table 43. RM Statistics Record CSLRST2

Field Name	Offset	Length	Field Usage	Description
RST2_ID	X'00'	X'08'	Input	Eyecatcher "CSLRST2".
RST2_LEN	X'08'	X'04'	Input	Length of valid data.
RST2_PVER	X'0C'	X'04'	Input	Parameter list version number (00000001).
RST2_PLEXNAME	X'10'	X'08'	Input	IMSplex name.
RST2_STRNAME	X'18'	X'10'	Input	Resource structure name.
RST2_STRVER	X'28'	X'08'	Input	Resource structure version.
RST2_CQSID	X'30'	X'08'	Input	CQS ID.
RST2_CLIENTS	X'38'	X'04'	Input	Number of registered clients.
RST2_CREATES	X'3C'	X'04'	Input	Number of resource creates.
RST2_UPDATES	X'40'	X'04'	Input	Number of resource updates.
RST2_DELETES	X'44'	X'04'	Input	Number of resource deletes.

Writing a CSL RM Client

If you want to use RM to manage global resources in an IMSplex for your own product or service, you have to write one or more RM clients. An RM client uses RM requests, some issued in a particular sequence, to communicate with RM.

To write an RM client, you can use the set of client requests provided by RM. These requests allow a client to access RM or resources on a resource structure, or to coordinate an IMSplex-wide process. One example of an RM client is IMS. You can write an RM client in assembler language.

An RM client uses RM requests to make use of RM services and resources. A client issues SCI and RM requests to request RM services. Some of the requests must follow a particular sequence. Other requests can be issued multiple times, in any order, based on the processing requirements of the client.

Before an RM client can issue RM requests, it must register:

- To SCI
- Its own resource types and associated name types to RM
- To each active RM in the IMSplex, so any RM can process an RM request

See “CSLRMREG: Register Clients” on page 136 for more information on RM client registration.

Table 44 lists the sequence of requests issued by an RM client. The request is listed with its purpose.

Table 44. Sequence of Requests for RM Client

Request	Purpose
CSLSCREG	Registers to SCI, which enables the client to send RM requests to RM through SCI.
CSLSCRDY	Readies the RM client to SCI, which routes messages to the client by client type
CSLRMREG	Registers client to RM to enable communication with RM. The client should register to each active RM in the IMSplex, so any RM can process an RM request. The client can also register its own resource types and associated name types to RM.
CSLRMxxx	Issues RM resource requests such as CSLRMUPD, CSLRMDEL, CSLRMQRY to manipulate resources on a resource structure.
CSLRMPxx	Issues RM process requests such as CSLRMPRI, CSLRMPRS, CSLRMPRR, and CSLRMPRT to participate in an IMSplex-wide process.
CSLSCBFR	Releases the output buffer returned by the request, if any.
CSLRMDRG	Deregisters client from RM to end communications with RM.
CSLSCDRG	Deregisters from SCI.

Table 45 lists the sequence of requests issued by an RM client that is participating in IMSplex-wide processes. The request is listed with its purpose.

Table 45. Sequence of Requests for RM Client Participating in IMSplex-wide Process

Request	Purpose
CSLRMPRI	Initiate an IMSplex-wide process.

Table 45. Sequence of Requests for RM Client Participating in IMSplex-wide Process (continued)

Request	Purpose
CSLRMPRS	Process a step in an IMSplex-wide process. A process can have zero, one, or more process steps. The client that initiates the process step is the master of the step.
CSLRMPRR	Respond to a process step.
CSLRMPRT	Terminate an IMSplex-wide process.

CSL RM Requests

This topic describes the requests associated with RM:

- "Using CSL RM Requests to Manage Global Resources"
- "Using CSL RM Requests to Coordinate IMSplex-wide Processes"

Using CSL RM Requests to Manage Global Resources

You can use the following requests to manage sysplex processes and maintain global resource information:

- "CSLRMDEL: Delete Resources" on page 113
- "CSLRMDRG: Deregister Clients" on page 117
- "CSLRMQRY: Query Resources" on page 131
- "CSLRMREG: Register Clients" on page 136
- "CSLRMUPD: Update Resources" on page 139

Before a client can access or change global resource information, it must register to SCI using the CSLSCREG request (see "CSLSCREG: Registration Request" on page 187). The client must issue an SCI registration request for every IMSplex with which it intends to communicate.

After the client registers to SCI, it must register to RM using the CSLRMREG request.

When the client is ready to terminate, it must deregister from RM using the CSLRMDRG request and then deregister from SCI using the CSLSCDRG request.

Using CSL RM Requests to Coordinate IMSplex-wide Processes

You can use RM-supplied requests to coordinate IMSplex-wide processes. All clients that are to participate in the process register to RM using the RM client registration request (CSLRMREG), if the clients have not registered already. One client initiates the process using the RM process initiate request (CSLRMPRI). The same or a different client initiates a step using RM process step request (CSLRMPRS). The initiating client is called the *master* of the step. One RM processes the request and sends RM directives to the other clients to perform the process step. All the other clients process the step, build output, and then respond to the step using the RM process respond request (CSLRMPRR). RM consolidates the responses from all the clients into one output, and then returns the output to the master of the process step. If there are more steps in the process, a client initiates a step, and the clients perform processing and respond. Any client terminates the process using the RM process terminate request (CSLRMPRT). Clients can deregister using the RM client deregistration request (CSLRMDRG) if required.

Some failures can cause RM to lose all knowledge of an IMSplex-wide process. These include resource structure failure (and its duplex, if applicable) and failure of all RMs. If this type of failure occurs, each RM client should clean up knowledge of the process locally, and a master RM should terminate the process. The first RM client to detect a problem can initiate a clean up process step by issuing the CSLRMPRS request with the force option; this enables RM to force the process step regardless of the error. The clients participating in the process step clean up the process locally. The master of this process step then terminates the process with the CSLRMPRT request.

The requests that can be used to coordinate IMSplex-wide processes include:

- "CSLRMPRI: Process Initiate" on page 118
- "CSLRMPRR: Process Respond" on page 121
- "CSLRMPRS: Process Step" on page 123
- "CSLRMPRT: Process Terminate" on page 129

CSLRMDEL: Delete Resources

Use the CSLRMDEL request to delete one or more uniquely named resources on a resource structure.

This request is supported in assembler language.

CSLRMDEL Syntax

The syntax for the CSLRMDEL request follows.

CSLRMDEL DSECT Syntax: Use the DSECT function of a CSLRMDEL request to include the following in your program:

- Equate (EQU) statement for the CSLRMDEL parameter list length
- The CSLRMDEL return codes, reason codes, and completion codes
- The CSLRDELL DSECT to map the input delete list
- The CSLRDELO DSECT to map the delete output

```
►►──CSLRMDEL──FUNC=DSECT──────────────────────────────────────────────►◄
```

CSLRMDEL DELETE Syntax: Use the DELETE function of a CSLRMDEL request to delete one or more uniquely named resources on a resource structure.

```
►►──CSLRMDEL──FUNC=DELETE──PARM=parm──LIST=deletelist─────────────────►

►─LISTLEN=deletelistlength──OUTPUT=output──OUTLEN=outputlength────────►

►─┬──────────┬──┬──────────────────────┬──────────────────────────────►
  └─ECB=ecb──┘  └─RETNAME=returnname───┘

►─┬─────────────────────────┬─RETCODE=returncode──────────────────────►
  └─RETTOKEN=returntoken────┘

►─RSNCODE=reasoncode──SCITOKEN=scitoken───────────────────────────────►◄
```

CSLRMDEL Parameters

The parameters for the CSLRMDEL request follow.

ECB=*symbol*

ECB=*(r2-r12)*

(Optional) - Specifies the address of a z/OS ECB used for asynchronous requests. When the request is complete, the ECB specified is posted. If an ECB is not specified, the task is suspended until the request is complete. If an ECB is specified, the invoker of the request must issue a WAIT (or equivalent) after receiving control from CSLRMDEL before using or examining any data returned by this request (including the RETCODE and RSNCODE fields).

LIST=*symbol*
LIST=*(r2-r12)*

(Required) - Specifies the delete resource list built by the caller. Each list entry is a separate delete request. The list length can vary, depending on the number of list entries.

CSLRDELL maps the delete resource list entry. The list contains a header and one or more list entries. The list entries must reside in contiguous storage. Each delete list entry contains the following:

- Resource name - the client-defined name of the resource.
- Resource type - a client-defined physical grouping of resources on the resource structure. Valid values are 1-255.

LISTLEN=*symbol*
LISTLEN=*(r2-r12)*

(Required) - Specifies the 4-byte delete resource list length.

OUTLEN=*symbol*
OUTLEN=*(r2-r12)*

(Required) - Specifies a 4-byte field to receive the length of the output returned by the CSLRMDEL request. OUTLEN contains the length of the output pointed to by the OUTPUT= parameter.

The output length is zero if no output is built, for example, if an error is detected before any output can be built.

OUTPUT=*symbol*
OUTPUT=*(r2-r12)*

(Required) - Specifies a 4-byte field to receive the address of the variable length output returned by the CSLRMDEL request. The output contains a header and one or more delete entries for resource deletes that were attempted. The output length is returned in the OUTLEN= field.

The output address is zero if no output was built, for example, if an error was detected before any output could be built.

The CSLRDELO macro maps the output that is returned. The output contains a header and one or more list entries.

The output header contains the following:

- Eyecatcher
- Output length
- CSLRDELO version
- CSLRDELO header length (offset to start of entries)
- CSLRDELO entry length
- Resource entry count

Each output entry represents a resource delete that failed. Each entry contains the following:

- Output entry length - the list entry length

- Name type - a client-defined value associated with a resource type that ensures uniqueness of client-defined resource names within a name type. Valid values are 1-255.
- Resource name
- Resource type
- Delete type
- Version - resource version of an existing resource if the delete request failed because of a version mismatch.
- Owner - resource owner of an existing resource if the delete failed because of a version mismatch and the option to read the owner was set.
- Completion code for the delete request. Completion codes are mapped by CSLRRR.

Possible completion codes are:

X'00000008'
Invalid resource type.

X'00000010'
Version mismatch. The version specified on input does not match the resource's version, so delete fails.

X'00000018'
Resource type is not registered. The resource type must be registered by using a CSLRMREG request.

X'00000024'
Resource structure is unavailable.

X'00000038'
Delete failed because of CQS internal error.

X'0000003C'
Delete failed because RM incorrectly built the CQSDEL list entry.

The output buffer is not preallocated by the caller. After being returned from the request, this word contains the address of a buffer containing the delete output. It is the caller's responsibility to release this storage by issuing the `CSLSCBFR FUNC=RELEASE` request when it is through with the storage. The length of the output is returned in the OUTLEN= field.

PARM=*symbol*
PARM=*(r2-r12)*
(Required) - specifies the CSLRMDEL parameter list. The length of the parameter list must be equal to the parameter list length EQU value defined by RDEL_PARMLN.

RETCODE=*symbol*
RETCODE=*(r2-r12)*
(Required) - specifies a 4-byte field to receive the return code on output. RM return codes are defined in CSLRRR. SCI return codes are defined in CSLSRR. Possible return codes are described in Table 46 on page 116.

RETNAME=*symbol*
RETNAME=*(r2-r12)*
(Optional) - Specifies an 8-byte field to receive the RM name returned to the caller. This is the CSL member name of the target RM address space to which SCI sent the request.

RETTOKEN=*symbol*

RETTOKEN=*(r2-r12)*
(Optional) - Specifies a 16-byte field to receive RM's SCI token returned to the caller. This is the SCI token for the target RM address space to which SCI sent the request.

RSNCODE=*symbol*
RSNCODE=*(r2-r12)*
(Required) - Specifies a 4-byte field to receive the reason code on output. RM reason codes are defined in CSLRRR. SCI reason codes are defined in CSLSRR. Possible reason codes are described in Table 46.

SCITOKEN=*symbol*
SCITOKEN=*(r2-r12)*
(Required) - Specifies a 16-byte field containing the SCI token. This token uniquely identifies this connection to SCI. The SCI token was returned by a successful `CSLSCREG FUNC=REGISTER` request.

CSLRMDEL Return and Reason Codes

Table 46 lists the return and reason codes that can be returned on a CSLRMDEL request. Also included is the meaning of a reason code (that is, what possibly caused it).

Table 46. CSLRMDEL Return and Reason Codes

Return Code	Reason Code	Meaning
X'00000000'	X'00000000'	The request completed successfully.
X'03000008'	X'00002000'	The client is not registered.
	X'00002100'	The delete list length is invalid.
	X'00002108'	The delete list address is invalid.
	X'00002110'	The version in the list header (DELL_PVER) is zero, which is invalid. The list version must be set in the list header to the maximum list version (DELL_PVERMAX).
	X'00002114'	The list header length is invalid. The list header length cannot be zero or greater than the list length that was passed in. The list header length (DELL_HDRLEN) must be set in the list header to the list header length.
	X'00002200'	One of the list entries contains an invalid resource type, such as zero. RM assumes that the rest of the list is invalid.
	X'0000220C'	One of the list entries contains one or more invalid delete options. RM assumes that the rest of the list is invalid.
	X'00002210'	A resource name or owner is required.
	X'00002214'	The version is invalid.
	X'00002404'	No resource structure is defined.

Table 46. CSLRMDEL Return and Reason Codes (continued)

Return Code	Reason Code	Meaning
X'0300000C'	X'00003000'	The request succeeded for at least one, but not all, list entries. Check the completion code in each list entry in the OUTPUT buffer for individual errors.
	X'00003004'	The request failed for all entries. Check the completion code in each list entry in the OUTPUT buffer for individual errors.
	X'00003008'	The request failed for one or more list entries and all failures were version mismatches. Check the completion code in each list entry in the OUTPUT buffer for individual errors.
X'03000010'	X'00004000'	The CQS address space is unavailable. Retry the request again, which attempts to route the request to a different RM with an available CQS.
	X'00004100'	The requested version is not supported. The client compiled with a version of CSLRMDEL that is not supported by RM. All RMs must be migrated to a new release before IMS is migrated to a new release that uses a new CSLRMDEL function.
	X'00004104'	The list version is not supported. The client created the delete list at a version that is not supported by RM. All RMs must be migrated to a new release before the client is migrated to a new release that uses a new CSLRMDEL function.
X'03000014'	X'00005000'	Storage allocation for the delete output buffer failed.
	X'00005120'	Storage allocation for the CQSDEL buffer failed.
	X'00005200'	The CQS request resulted in unexpected error.
	X'00005204'	The CQS request failed because RM incorrectly built the request input.

CSLRMDRG: Deregister Clients

The deregister request is issued by a client when the client no longer wants to process resource requests or IMSplex-wide process requests from RM. The deregister request removes client information from RM and stops RM from sending new resource requests to the client. Some information about the client is retained that can affect IMSplex-wide processes.

This request is issued by resource processing clients such as the IMS control region.

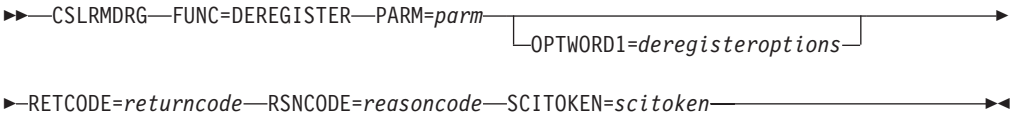
This request is supported in assembler.

CSLRMDRG Syntax

The syntax for the CSLRMDRG request follows.

CSLRMDRG DSECT Syntax: Use the DSECT function of a CSLRMDRG request to include equate (EQU) statements in your program for the CSLRMDRG parameter list length and the deregister options.

CSLRMDRG Deregister Syntax: Use the DEREGISTER function of a CSLRMDRG request to deregister from RM.



CSLRMDRG Parameters

OPTWORD1=*symbol*
OPTWORD1=*(r2-r12)*
(Optional) - Specifies a 4-byte field containing deregistration options. `CSLRMDRG FUNC=DSECT` generates the equates for deregistration options.

X'80000000'
Remove client from IMSplex. Delete all knowledge of the client.

PARM=*symbol*
PARM=*(r2-r12)*
(Required) - specifies the CSLRMDRG parameter list. The length of the parameter list must be equal to the parameter list length EQU value defined by RDRG_PARMLN.

RETCODE=*symbol*
RETCODE=*(r2-r12)*
(Required) - specifies a 4-byte field to receive the return code on output. RM return codes are defined in CSLRRR. RM does not return a response to the CSLRMDRG request.

RSNCODE=*symbol*
RSNCODE=*(r2-r12)*
(Required) - Specifies a 4-byte field to receive the reason code on output. SCI reason codes are defined in CSLSRR. RM does not return a response to the CSLRMDRG request.

SCITOKEN=*symbol*
SCITOKEN=*(r2-r12)*
(Required) - Specifies a 16-byte field containing the SCI token. This token uniquely identifies this connection to SCI. The SCI token was returned by a successful `CSLSCREG FUNC=REGISTER` request.

CSLRMPRI: Process Initiate

Use the CSLRMPRI request for a client to initiate a process across the IMSplex.

RM ensures that only one IMSplex-wide process of a type can be in progress at one time. The process initiation fails if any other IMSplex-wide process of the type is in progress.

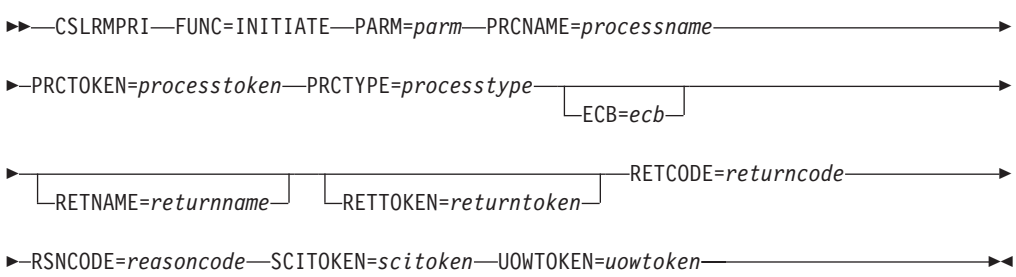
This request is supported in assembler language.

CSLRMPRI Syntax

The syntax for the CSLRMPRI request follows.

CSLRMPRI DSECT Syntax: Use the DSECT function of a CSLRMPRI request to include equate (EQU) statements in your program for the length of the CSLRMPRI parameter list.

CSLRMPRI INITIATE Syntax: Use the INITIATE function of a CSLRMPRI request to initiate an IMSplex-wide process.



CSLRMPRI Parameters

The parameters for the CSLRMPRI request follow.

ECB=*symbol*
ECB=*(r2-r12)*
(Optional) - Specifies the address of a z/OS ECB used for asynchronous requests. When the request is complete, the ECB specified is posted. If an ECB is not specified, the task is suspended until the request is complete. If an ECB is specified, the invoker of the request must issue a WAIT (or equivalent) after receiving control from CSLRM PRI before using or examining any data returned by this request (including the RETCODE and RSNCODE fields).

PARM=*symbol*
PARM=*(r2-r12)*
(Required) - Specifies the CSLRMPRI parameter list. The length of the parameter list must be equal to the parameter list length EQU value defined by RPRI_PARMLN.

PRCNAME=*symbol*
PRCNAME=*(r2-r12)*
(Required) - specifies an 8-byte field containing the process name. The process name is client defined and has no meaning to RM. RM uses the process name and the process type to insure that only one instance of a process of a particular process type is in progress at one time.

PRCTOKEN=*symbol*
PRCTOKEN=*(r2-r12)*
(Required) - specifies a 16-byte field to receive the process token returned to the caller. The process token uniquely identifies the process instance. The process token returned is zero, if the IMSplex is defined with a resource structure. The process token is non-zero, if the IMSplex is not defined with a resource structure. The process token must be specified as input on any subsequent CSLRMPRS, CSLRMPRR, or CSLRMPRT request.

PRCTYPE=*symbol*
PRCTYPE=*(r2-r12)*
(Required) - specifies a 1-byte client-defined process type. Only one process of a particular type can be in progress at any one time. The process type can be 1 through 255.

RETCODE=*symbol*
RETCODE=*(r2-r12)*
(Required) - specifies a 4-byte field to receive the return code on output. RM

return codes are defined in CSLRRR. SCI return codes are defined in CSLSRR. Possible return codes are described in Table 47.

RETNAME=*symbol*
RETNAME=*(r2-r12)*
(Optional) - Specifies an 8-byte field to receive the RM name returned to the caller. This is the CSL member name of the target RM address space to which SCI sent the request.

RETTOKEN=*symbol*
RETTOKEN=*(r2-r12)*
(Optional) - Specifies a 16-byte field to receive RM's SCI token returned to the caller. This is the SCI token for the target RM address space to which SCI sent the request.

RSNCODE=*symbol*
RSNCODE=*(r2-r12)*
(Required) - Specifies a 4-byte field to receive the reason code on output. RM reason codes are defined in CSLRRR. SCI reason codes are defined in CSLSRR. Possible reason codes are described in Table 47.

SCITOKEN=*symbol*
SCITOKEN=*(r2-r12)*
(Required) - Specifies a 16-byte field containing the SCI token. This token uniquely identifies this connection to SCI. The SCI token was returned by a successful `CSLSCREG FUNC=REGISTER` request.

UOWTOKEN=*symbol*
UOWTOKEN=*(r2-r12)*
(Required) - specifies a 16-byte field containing the unit of work token. The UOW token uniquely identifies an instance of this process and ties all of the process steps together. The UOW token must be specified on the RM process step request, CSLRMPRS. The UOW token is client-defined and has no meaning to RM.

CSLRMPRI Return and Reason Codes

Table 47 lists the return and reason codes that can be returned on a CSLRMPRI request. Also included is the meaning of a reason code (that is, what possibly caused it).

Table 47. CSLRMPRI Return and Reason Codes

Return Code	Reason Code	Meaning
X'00000000'	X'00000000'	The request completed successfully.
X'03000008'	X'00002000'	The client is not registered.
	X'00002208'	The process type is invalid.
	X'00002310'	The UOW token is invalid.

Table 47. CSLRMPRI Return and Reason Codes (continued)

Return Code	Reason Code	Meaning
X'03000010'	X'00004000'	The CQS address space is unavailable. Retry the request to attempt routing the request to another RM with an available CQS.
	X'00004100'	The requested version is not supported. The client compiled with a version of CSLRMPRI that is not supported by RM. All RMs must be migrated to a new release before IMS is migrated to a new release that uses a new CSLRMPRI function.
	X'00004120'	A process of the same type is already in progress. This process initiation request is rejected. Try the process again later.
	X'0000412C'	A different process of the same type is already in progress. This process initiation request is rejected. Try the process again later.
X'03000014'	X'00005114'	The process block allocation failed.
	X'00005200'	The CQS request resulted in unexpected error.
	X'00005204'	The CQS request failed because RM incorrectly built the request input.
	X'00005208'	The resource structure is not available.
	X'0000520C'	The resource structure is full.
	X'00005210'	RM is unable to add the process block to hash table.
	X'00005218'	RM is unable to scan the process block in hash table.
	X'00005220'	RM is unable to get the process latch.

CSLRMPRR: Process Respond

Use the CSLRMPRR request for a client to respond to a step in an IMSplex-wide process.

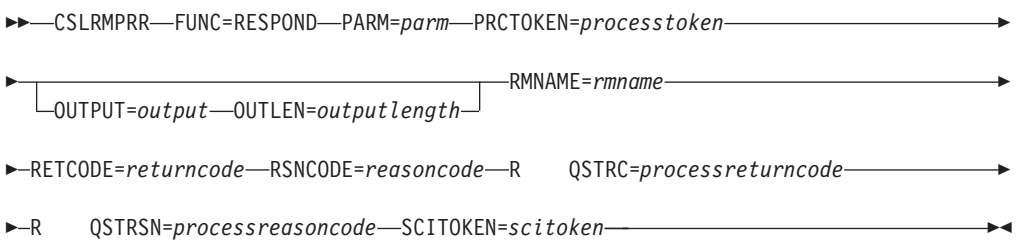
This request is supported in assembler language.

CSLRMPRR Syntax

The syntax for the CSLRMPRR request follows.

CSLRMPRR DSECT Syntax: Use the DSECT function of a CSLRMPRR request to include equate (EQU) statements in your program for the length of the CSLRMPRR parameter list.

CSLRMPRR RESPOND Syntax: Use the RESPOND function of a CSLRMPRR request to respond to a step in an IMSplex-wide process.



CSLRMPRR Parameters

The parameters for the CSLRMPRR request follow.

OUTLEN=*symbol*
OUTLEN=*(r2-r12)*
(Required) - specifies a 4-byte input field that contains the length of the process step output buffer. OUTLEN= contains the length of the output pointed to by the OUTPUT= parameter.

OUTPUT=*symbol*
OUTPUT=*(r2-r12)*
(Required) - Specifies a 4-byte field that contains the address of the output buffer built by the caller. The output is client-defined and contains the results from this client's processing of the step. The output length is returned in the OUTLEN= field.

PARM=*symbol*
PARM=*(r2-r12)*
(Required) - specifies the CSLRMPRR parameter list. The length of the parameter list must be equal to the parameter list length EQU value defined by RPRR_PARMLN.

PRCTOKEN=*symbol*
PRCTOKEN=*(r2-r12)*
(Required) - specifies a 16-byte field that contains the process token that uniquely identifies the process. This token was returned on a successful CSLRMPRI FUNC=INITIATE request.

If the IMSplex is defined with a resource structure, the process token is zero.

RETCODE=*symbol*
RETCODE=*(r2-r12)*
(Required) - specifies a 4-byte field to receive the return code on output. SCI return codes are defined in CSLSRR. RM does not return a response to CSLRMPRR.

RMNAME=*symbol*
RMNAME=*(r2-r12)*
(Required) - specifies an 8-byte field containing the RM name to which to send the process step response. This is the RM that originated the process step.

RQSTRC=*symbol*
RQSTRC=*(r2-r12)*
(Required) - specifies a 4-byte field that contains the return code to be passed to the originator of the process step on output. The return code is defined by the process step originating client and indicates the result of the process step.

RQSTRSN=*symbol*

RQSTRSN=*(r2-r12)*
(Required) - specifies a 4-byte field that contains the reason code to be passed to the originator of the process step on output. The reason code is defined by the process step originating client and indicates the result of the process step.

RSNCODE=*symbol*
RSNCODE=*(r2-r12)*
(Required) - Specifies a 4-byte field to receive the reason code on output. RM reason codes are defined in CSLRRR. RM does not return a response to CSLRMPRR.

SCITOKEN=*symbol*
SCITOKEN=*(r2-r12)*
(Required) - Specifies a 16-byte field containing the SCI token. This token uniquely identifies this connection to SCI. The SCI token was returned by a successful `CSLSCREG FUNC=REGISTER` request.

CSLRMPRR Return and Reason Codes

CSLRMPRR is sent to the target client address space using the SCI message protocol; RM does not return codes to CSLRMPRR. All return and reason codes that are applicable to the CSLSCMSG request can be returned on a CSLRMPRR request.

CSLRMPRS: Process Step

Use the CSLRMPRS request for a client to perform a step in a process. An IMSplex-wide process can consist of zero, one, or more steps.

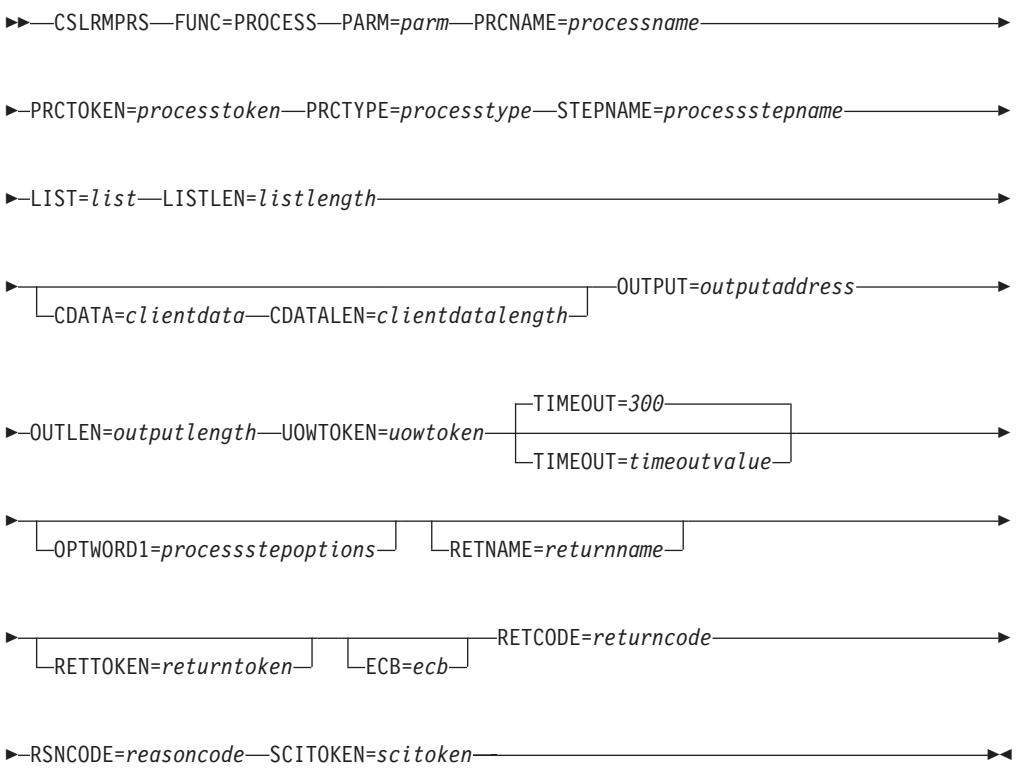
This request is supported in assembler language.

CSLRMPRS Syntax

The syntax for the CSLRMPRS request follows.

CSLRMPRS DSECT Syntax: Use the DSECT function of a CSLRMPRS request to include equate (EQU) statements in your program for the length of the CSLRMPRS parameter list and the process step request options.

CSLRMPRS PROCESS Syntax: Use the PROCESS function of a CSLRMPRS request to perform a step in an IMSplex-wide process.



CSLRMPRS Parameters

The parameters for the CSLRMPRS request follow.

CDATA=*symbol*
CDATA=*(r2-r12)*
(Optional) - specifies a variable length area that contains client data to send to clients participating in the IMSplex-wide process step. The client data has meaning to clients, not to RM.

CDATALEN=*symbol*
CDATALEN=*(r2-r12)*
(Optional) - specifies a 4-byte input field that contains the client data length. If this parameter is specified, CDATA= must also be specified.

ECB=*symbol*
ECB=*(r2-r12)*
(Optional) - Specifies the address of a z/OS ECB used for asynchronous requests. When the request is complete, the ECB specified is posted. If an ECB is not specified, the task is suspended until the request is complete. If an ECB is specified, the invoker of the request must issue a WAIT (or equivalent) after receiving control from CSLRM PRS before using or examining any data returned by this request (including the RETCODE and RSNCODE fields).

LIST=*symbol*
LIST=*(r2-r12)*
(Required) - specifies the variable length input list that contains the list of clients to which to send the process step.

The process step list contains a list header and one or more list entries. The list header contains the list header length, the parameter list version, the list entry

length, the list entry count, and a user data area. The list header user data area is passed back to the requestor in the list header of the process step output. Each list entry contains the client name and an optional user data area. The user data area is passed back to the requestor in a list entry in the process step output. The list entries must reside in contiguous storage.

The CSLRPRSL macro maps the process step list.

LISTLEN=*symbol*
LISTLEN=*(r2-r12)*
(Required) - specifies a 4-byte input field that contains the process step list length.

OPTWORD1=*symbol*
OPTWORD1=*(r2-r12)*
(Optional) - specifies a 4-byte field containing the process step options. `CSLRMPRS FUNC=DSECT` maps the process step options.

X'80000000'
Force process step after error. Take over a process step in progress, if a process step is already in progress for an IMSplex member that is not active. Initiate a process and perform a process step if no process is known to be in progress due to an error such as resource structure failure.

OUTLEN=*symbol*
OUTLEN=*(r2-r12)*
(Required) - specifies a 4-byte field to receive the length of the output buffer returned by the CSLRMPRS request. After being returned by request, this word contains the length of the buffer pointed to by the OUTPUT= parameter. If no output is built, the output buffer length is zero. This can occur if an error is detected before any output can be built.

It is the caller's responsibility to release this storage by issuing the `CSLSCBFR FUNC=RELEASE` request when it is through with the storage.

OUTPUT=*symbol*
OUTPUT=*(r2-r12)*
(Required) - specifies a 4-byte field to receive the address of the variable length output buffer returned by the CSLRMPRS request. The output buffer contains the client-defined data from each participating client and indicates the results of the process step. The output buffer length is returned in the OUTLEN= field.

If no output is built, the output buffer address is zero. This can occur if an error is detected before any output can be built.

The CSLRPRSO macro maps the output buffer that is returned. The output buffer header contains an eyecatcher, the output buffer length, the CSLRPRSO version, the header length (offset to start of the process list entries), the list entry minimum size, the process list entry count, a user data area, and the CSLRPRSO create timestamp. The user data area contains the user data passed in the input process step list header.

Each output buffer entry represents the results from a client that participated in a process step. Each entry contains the following:

- Entry length
- Client name
- User data - the user data passed in the input process step list
- Process step response length
- Process step response

- Completion code (CSLRRR) - possible completion codes are:

  X'00000000'
  Client processes step successfully.

  X'00000044'
  Client did not respond before the process step timed out.

  X'00000048'
  The client was not sent the process step request because the client is not registered to RM.

  This buffer is not preallocated by the caller. After the request returns it, this word contains the address of a buffer containing information from the IMSplex members participating in the process. It is the caller's responsibility to release this storage by issuing the `CSLSCBFR FUNC=RELEASE` request when it is through with the storage.

PARM=*symbol*
PARM=*(r2-r12)*
(Required) - specifies the CSLRMPRS parameter list. The length of the parameter list must be equal to the parameter list length EQU value defined by RPRS_PARMLN.

PRCNAME=*symbol*
PRCNAME=*(r2-r12)*
(Required) - specifies an 8-byte field containing the process name. The process name is client defined and has no meaning to RM. RM uses the process name and type to insure that only one instance of a process, with a particular process type, is in progress at one time.

PRCTOKEN=*symbol*
PRCTOKEN=*(r2-r12)*
(Required) - specifies a 16-byte field that contains the process token that uniquely identifies the process. This token was returned on a successful `CSLRMPRI FUNC=INITIATE` request.

If the IMSplex is defined with a resource structure, the process token is zero.

PRCTYPE=*symbol*
PRCTYPE=*(r2-r12)*
(Required) - specifies a 1-byte client-defined process type. Only one process of a particular type can be in progress at any one time. The process type can be 1 through 255.

RETCODE=*symbol*
RETCODE=*(r2-r12)*
(Required) - specifies a 4-byte field to receive the return code on output. SCI return codes are defined in CSLSRR. SCI return codes are defined in CSLSRR. Possible return codes are described in Table 48 on page 128.

RETNAME=*symbol*
RETNAME=*(r2-r12)*
(Optional) - specifies an 8-byte field to receive the RM name returned to the caller. This is the CSL member name of the target RM address space to which SCI sent the request.

RETNAME=*symbol*

RETNAME=*(r2-r12)*
(Optional) - specifies an 8-byte field to receive the RM name returned to the caller. This is the CSL member name of the target RM address space to which SCI sent the request.

RSNCODE=*symbol*
RSNCODE=*(r2-r12)*
(Required) - Specifies a 4-byte field to receive the reason code on output. RM reason codes are defined in CSLRRR. SCI reason codes are defined in CSLSRR. Possible reason codes are described in Table 48 on page 128.

SCITOKEN=*symbol*
SCITOKEN=*(r2-r12)*
(Required) - Specifies a 16-byte field containing the SCI token. This token uniquely identifies this connection to SCI. The SCI token was returned by a successful `CSLSCREG FUNC=REGISTER` request.

STEPNAME=*symbol*
STEPNAME=*(r2-r12)*
(Required) - Specifies an 4-byte field containing the process step name. The process step name is client-defined and has no meaning to RM. Each process step must have a different name.

TIMEOUT=*timeoutvalue*
TIMEOUT=*symbol*
TIMEOUT=*(r2-r12)*
(Optional) - Specifies a 4-byte field containing the process step timeout value in seconds. If the timeout value is reached during the processing of the step, before all of the participants have responded to the process step, RM terminates the process step and returns the available responses. If the specified timeout value is too small, an incomplete response is returned. The TIMEOUT value ensures a response is returned even if a client processing the step is unable to respond.

The default timeout value is 5 minutes (300 seconds). Specify a negative one (-1) value if no timeout is required for the request.

The TIMEOUT value is the shortest possible time value that can cause the process step to time out. RM internally sets a timer to pop every 5 seconds. When the RM timer pops, RM checks to see if any process step timeout value has expired. When the process step timeout value is less than the RM timer value, the actual length of step processing can be longer than the user specified TIMEOUT value.

UOWTOKEN=*symbol*
UOWTOKEN=*(r2-r12)*
(Required) - Specifies a 16-byte field containing the unit of work token. The UOW token uniquely identifies an instance of this process and ties all of the process steps together. The UOW token must match the UOW token specified on the `CSLRMPRI FUNC=INITIATE` request. The UOW token is client-defined and has no meaning to RM.

CSLRMPRS Return and Reason Codes

Table 48 on page 128 lists the return and reason codes that can be returned on a CSLRMPRS request. Also included is the meaning of a reason code (that is, what possibly caused it).

Table 48. CSLRMPRS Return and Reason Codes

Return Code	Reason Code	Meaning
X'00000000'	X'00000000'	The request completed successfully.
X'03000008'	X'00002000'	The client is not registered.
	X'00002110'	The list version in the list header (PRSL_PVER) is zero, which is invalid. The list version must be set in the list header to the maximum list version (PRSL_PVERMAX).
	X'00002114'	The list header length cannot be zero or greater than the list length that was passed in. The list header length (PRSL_HDRLEN) must be set in the list header to the list header length.
	X'00002140'	The client data length cannot be zero or greater than 256.
	X'00002208'	The process type is invalid.
	X'0000220C'	The process step options are invalid.
	X'00002300'	The process token is invalid.
	X'00002310'	The UOW token is invalid.
X'0300000C'	X'00003000'	The process step succeeded for at least one client, but not all. Check the completion code in each list entry in the OUTPUT buffer for individual errors.
	X'00003004'	The request failed for all clients. Check the completion code in each list entry in the OUTPUT buffer for individual errors.
X'03000010'	X'00004000'	The CQS address space is unavailable. Retry the request to attempt routing the request to another RM with an available CQS.
	X'00004100'	The requested version is not supported. The client compiled with a version of CSLRMPRS that is not supported by RM. All RMs must be migrated to a new release before IMS is migrated to a new release that uses a new CSLRMPRS function.
	X'00004104'	The version of the list is not supported. The client created the process step list at a version that is not supported by RM. All RMs must be migrated to a new release before the client is migrated to a new release that uses a new CSLRMPRS function.
	X'00004108'	The SCI address space is unavailable. SCI was available to send the CSLRMPRS request to RM. RM tried coordinating the process step by sending SCI messages to the active clients. The SCI request to send a message to SCI failed for one or more active clients that did not have an SCI active on the system. Some of the clients might have successfully processed the step.
	X'00004124'	A process is not in progress. The process step is rejected.
	X'00004128'	A process step is already in progress. The process step is rejected. If a process step is already in progress because an error occurred while a previous process step was in progress, and the owner of that process step is still active, the next process step must be specified by the owner of the process step with the FORCE option.

Table 48. CSLRMPRS Return and Reason Codes (continued)

Return Code	Reason Code	Meaning
X'03000014'	X'00005000'	Storage allocation for the output buffer failed. The process step might or might not have succeeded.
	X'00005114'	The process block allocation failed.
	X'00005118'	The process step response block allocation failed.
	X'00005200'	The CQS request resulted in an unexpected error.
	X'00005204'	The CQS request failed because RM incorrectly built the request input.
	X'00005208'	The resource structure is not available.
	X'00005210'	RM is unable to add the process block to hash table.
	X'00005214'	RM is unable to find the process block in hash table.
	X'00005218'	RM is unable to scan the process block in hash table.
	X'00005300'	An SCI error was encountered. SCI was available to send the CSLRMPRS request to RM. RM tried coordinating the process step by sending SCI messages to the active clients. The SCI request to send a message to SCI failed with an error for one or more active clients. Some of the clients might have successfully processed the step.

CSLRMPRT: Process Terminate

Use the CSLRMPRT request to terminate an IMSplex-wide process. Any client that is participating in the process can issue a CSLRMPRT FUNC= TERMINATE request to terminate the process.

This request is supported in assembler language.

CSLRMPRT Syntax

The syntax for the CSLRMPRT request follows.

DSECT Syntax: Use the DSECT function of a CSLRMPRT request to include equate (EQU) statements in your program for the length of the CSLRMPRT parameter list.

```
►►—CSLRMPRT—FUNC=DSECT———————————————————————————————————►◄
```

TERMINATE Syntax: Use the TERMINATE function of a CSLRMPRT request to terminate an IMSplex-wide process.

```
►►—CSLRMPRT—FUNC=TERMINATE—PARM=parm—PRCNAME=processname————————————►

►—PRCTOKEN=processtoken—PRCTYPE=processtype—UOWTOKEN=uowtoken————————►

►———————————————————————————————————RETCODE=returncode————————►
  └RETNAME=returnname┘  └RETTOKEN=returntoken┘

►—RSNCODE=reasoncode—SCITOKEN=scitoken———————————————————————————►◄
```

CSLRMPRT Parameters

The parameters for the CSLRMPRT request follow.

PARM=*symbol*
PARM=*(r2-r12)*
(Required) - specifies the CSLRMPRT parameter list. The length of the parameter list must be equal to the parameter list length EQU value defined by RPRT_PARMLN.

PRCNAME=*symbol*
PRCNAME=*(r2-r12)*
(Required) - specifies an 8-byte field containing the process name. The process name is client defined and has no meaning to RM. RM uses the process name and the process type to insure that only one instance of a process of a particular process type is in progress at one time.

PRCTOKEN=*symbol*
PRCTOKEN=*(r2-r12)*
(Required) - specifies a 16-byte field that contains the process token that uniquely identifies the process. This token was returned on a successful `CSLRMPRI FUNC=INITIATE` request.

If the IMSplex is defined with a resource structure, the process token is zero.

PRCTYPE=*symbol*
PRCTYPE=*(r2-r12)*
(Required) - specifies a 1-byte client-defined process type. Only one process of a particular type can be in progress at any one time. The process type can be 1 through 255.

RETCODE=*symbol*
RETCODE=*(r2-r12)*
(Required) - specifies a 4-byte field to receive the return code on output. SCI return codes are defined in CSLSRR. RM does not return a response to CSLRMPRT.

RETNAME=*symbol*
RETNAME=*(r2-r12)*
(Optional) - Specifies an 8-byte field to receive the name of the RM address space to which SCI sent the process terminate request.

RETTOKEN=*symbol*
RETTOKEN=*(r2-r12)*
(Optional) - Specifies a 16-byte field to receive the SCI token of the RM address space to which SCI sent the process terminate request.

RSNCODE=*symbol*
RSNCODE=*(r2-r12)*
(Required) - Specifies a 4-byte field to receive the reason code on output. SCI reason codes are defined in CSLSRR. RM does not return a response to the CSLRMPRT request.

SCITOKEN=*symbol*
SCITOKEN=*(r2-r12)*
(Required) - Specifies a 16-byte field containing the SCI token. This token uniquely identifies this connection to SCI. The SCI token was returned by a successful `CSLSCREG FUNC=REGISTER` request.

UOWTOKEN=*symbol*
UOWTOKEN=*(r2-r12)*
(Required) - Specifies a 16-byte field containing the unit of work token. The

UOW token uniquely identifies an instance of this process and ties all of the process steps together. The UOW token must match the UOW token specified on the `CSLRMPRI FUNC=INITIATE` request. The UOW token is client-defined and has no meaning to RM.

CSLRMPRT Return and Reason Codes

CSLRMPRT is sent to the target client address space using the SCI message protocol. All return and reason codes that are applicable to the CSLSCMSG request can be returned on a CSLRMPRT request. CSLRMPRT does not issue any additional return and reason codes.

CSLRMQRY: Query Resources

Use the CSLRMQRY request to query one or more uniquely named resources on a resource structure.

This request is supported in assembler language.

CSLRMQRY Syntax

The syntax for the CSLRMQRY request follows.

CSLRMQRY DSECT Syntax: Use the DSECT function of a CSLRMQRY request to include the following inputs and outputs in your program:

- Equate (EQU) statements for the length of the CSLRMQRY parameter list
- The CSLRMQRY return codes, reason codes, and completion codes
- The CSLRQRYL DSECT to map the input query list
- The CSLRQRYO DSECT to map the query output

```
►►—CSLRMQRY—FUNC=DSECT———————————————————————————————————►◄
```

CSLRMQRY QUERY Syntax: Use the QUERY function of a CSLRMQRY request to query one or more uniquely named resources on a resource structure.

```
►►—CSLRMQRY—FUNC=QUERY—PARM=parm—LIST=querylist———————————————►

►—LISTLEN=querylistlength———————————————————————————————————►
                          └—RETNAME=returnname—┘

►———————————————————————OUTPUT=output—OUTLEN=outputlength———————►
  └—RETTOKEN=returntoken—┘

                  ┌—PROTOCOL=RQST—┐
►———————————————————————————————RETCODE=returncode———————————►
  └—ECB=ecb—┘

►—RSNCODE=reasoncode—SCITOKEN=scitoken———————————————————————►◄
```

CSLRMQRY Parameters

The parameters for the CSLRMQRY request follow.

ECB=*symbol*

ECB=*(r2-r12)*

(Optional) - Specifies the address of a z/OS ECB used for asynchronous requests. When the request is complete, the ECB specified is posted. If an ECB is not specified, the task is suspended until the request is complete. If an ECB is specified, the invoker of the request must issue a WAIT (or equivalent) after receiving control from CSLRM QRY before using or examining any data returned by this request (including the RETCODE and RSNCODE fields).

LIST=*symbol*
LIST=*(r2-r12)*

(Required) - Specifies the query resource list built by the caller. Each list entry is a separate query request. The list length can vary, depending upon the number of list entries.

The list contains a header and one or more list entries. The list entries must reside in contiguous storage. Each query list entry contains the following:

- Resource name - client-defined name of the resource.
- Resource type - the resource type is a client-defined physical grouping of resources on the resource structure. Valid values are 1-255.
- Query options - options that indicate special processing to perform for the query.
- Owner - owner of the resource.
- Optional user field - optional 4-byte user field, set by the caller, that will be passed back in the output list entry associated with the input list entry.

LISTLEN=*symbol*
LISTLEN=*(r2-r12)*

(Required) - specifies the 4-byte query resource list length.

OUTLEN=*symbol*
OUTLEN=*(r2-r12)*

(Required) - Specifies a 4-byte field to receive the length of the output buffer returned by the CSLRMQRY request. OUTLEN contains the length of the output buffer pointed to by the OUTPUT parameter. The length of the output data (header and entries) is passed in the output header data, mapped by CSLRQRYO.

OUTPUT=*symbol*
OUTPUT=*(r2-r12)*

(Required) - Specifies a 4-byte field to receive the address of the variable length output returned by the CSLRMQRY request. The output contains a header and one or more query entries for resource queries that were attempted. The output length is returned in the OUTLEN field.

The output address is zero if no output was built, for example, if an error was detected before any output could be built.

The CSLRQRYO macro maps the output that is returned. The output contains a header and one or more list entries. The header contains the following:

- an eyecatcher
- the output length
- CSLRQRYO version
- CSLRQRYO header length (offset to start of entries)
- minimum entry length (offset to DATA2)
- resource entry count
- timestamp

Each output entry represents a resource query that was attempted. Each entry contains the following:

- Output entry length - the list entry length can vary, depending upon whether DATA2 is returned.
- Name type - the name type is a client-defined value associated with a resource type that ensures uniqueness of client-defined resource names within a name type. Valid values are 1-255.
- Resource name - client-defined name of the resource.
- Resource type - the resource type is a client-defined physical grouping of resources on the resource structure. Valid values are 1-255.
- Version - the resource version, which is the number of times the resource has been updated.
- DATA2 flag byte - flag byte indicating if DATA2 was read.
- Resource name status flag - the resource name status indicates how the resource name in the query output list entry is associated with the input resource parameter. This enables you to tie the input resource parameter to the output query list entries that are generated. The following resource name status are possible:

  Specific parameter
  A specific resource name was specified. This query list entry contains the resource name that matches the input parameter.

  Wildcard Parameter
  A wildcard parameter was specified. This query list entry contains the wildcard parameter and a completion code. This query list entry does not contain information about a specific resource. If the completion code is zero, one or more wildcard match list entries follow.

  Wildcard match
  A wildcard parameter was specified. This entry contains information about one resource that matches the input wildcard parameter. All wildcard match list entries follow contiguously after a wildcard parameter list entry.
- Owner - owner of a resource.
- DATA1- a small piece of client data (fixed length, contained in the adjunct area of a data entry) associated with an existing resource.
- DATA2 length - length of a large piece of client data associated with an existing resource, if DATA2 exists and the option to read DATA2 was set.
- Optional User field - optional 4 byte user field passed back to the caller in the output list entry associated with the input list entry.
- DATA2 - a large piece of client data (variable length, contained in one or more data elements of a data entry) associated with an existing resource, if DATA2 exists, and the option to read DATA2 was set. The maximum size of DATA2 is 61312 bytes (X'EF80').
- Completion code for the query request - completion codes are mapped by CSLRRR. Possible completion codes are:

X'00000000'
Query request succeeded. At least one resource matching the query parameters is returned in the output buffer specified by OUTPUT=.

X'00000004'
No resources found.

X'00000008'
Invalid resource type.

X'0000000C'
Invalid name type.

X'00000024'
Resource structure is unavailable.

X'00000034'
Invalid options specified.

X'00000038'
Query failed because of CQS internal error.

X'0000003C'
Query failed because RM incorrectly built the CQSBRWSE list entry.

The output buffer is not preallocated by the caller. After the request returns it, this word contains the address of a buffer containing the query output. It is the caller's responsibility to release this storage by issuing the `CSLSCBFR FUNC=RELEASE` request when it is through with the storage. The length of the buffer is returned in the OUTLEN= field.

PARM=*symbol*
PARM=*(r2-r12)*
(Required) - Specifies the CSLRMQRY parameter list. The length of the parameter list must be equal to the parameter list length EQU value defined by RQRY_PARMLN.

PROTOCOL=RQST
(Optional) - SCI protocol for sending the request to RM. RQST sends the query request using SCI request interface.

RETCODE=*symbol*
RETCODE=*(r2-r12)*
(Required) - specifies a 4-byte field to receive the return code on output. RM return codes are defined in CSLRRR. SCI return codes are defined in CSLSRR. Possible return codes are described in Table 49 on page 135.

RETNAME=*symbol*
RETNAME=*(r2-r12)*
(Optional) - Specifies an 8-byte field to receive the RM name returned to the caller. This is the CSL member name of the target RM address space to which SCI sent the request.

RETTOKEN=*symbol*
RETTOKEN=*(r2-r12)*
(Optional) - Specifies a 16-byte field to receive RM's SCI token returned to the caller. This is the SCI token for the target RM address space to which SCI sent the request.

RSNCODE=*symbol*
RSNCODE=*(r2-r12)*
(Required) - Specifies a 4-byte field to receive the reason code on output. RM reason codes are defined in CSLRRR. SCI reason codes are defined in CSLSRR. Possible reason codes are described in Table 49 on page 135.

SCITOKEN=*symbol*
SCITOKEN=*(r2-r12)*
(Required) - Specifies a 16-byte field containing the SCI token. This token

uniquely identifies this connection to SCI. The SCI token was returned by a successful `CSLSCREG FUNC=REGISTER` request.

CSLRMQRY Return and Reason Codes

Table 49 lists the return and reason codes that can be returned on a CSLRMQRY request. Also included is the meaning of a reason code (that is, what possibly caused it).

Table 49. CSLRMQRY Return and Reason Codes

Return Code	Reason Code	Meaning
X'00000000'	X'00000000'	The request completed successfully.
X'03000004'	X'00001000'	No resources were found.
X'03000008'	X'00002000'	The client is not registered.
	X'00002100'	The query-list length is invalid.
	X'00002108'	The query-list address is invalid.
	X'00002110'	The list version in the list header (QRYL_PVER) is zero, which is invalid. The list version must be set in the list header to the maximum list version (QRYL_PVERMAX).
	X'00002114'	The list header length cannot be zero or greater than the list length that was passed in. The list header length (QRYL_HDRLEN) must be set in the list header to the list header length.
	X'00002404'	No resource structure is defined.
X'0300000C'	X'00003000'	The request succeeded for at least one list entry, but not all. Check the completion code in each query list entry in the OUTPUT buffer for individual errors.
	X'00003004'	The request failed for all entries. Check the completion code in each query list entry in the OUTPUT buffer for individual errors.
X'03000010'	X'00004000'	The CQS address space is unavailable. Retry the request again to attempt routing the request to another RM with an available CQS.
	X'00004100'	The requested version is not supported. The client compiled with a version of CSLRMQRY that is not supported by RM. All RMs must be migrated to a new release before IMS is migrated to a new release that uses a new CSLRMQRY function.
	X'00004104'	The list version is not supported. The client created the query list at a version that is not supported by RM. All RMs must be migrated to a new release before the client is migrated to a new release that uses a new CSLRMQRY function.
X'03000014'	X'00005000'	Storage allocation for the query output buffer failed.
	X'00005108'	Storage allocation for the CQSBRWSE buffer failed.
	X'00005200'	The CQS request resulted in an unexpected error.
	X'00005204'	The CQS request failed because RM incorrectly built the request input.

CSLRMREG: Register Clients

Use the CSLRMREG request to register a client to RM and, optionally, to register the client's resource types and associated name types. The client must be authorized to issue a CSLRMREG request. You cannot register a client if an IMSplex-wide process is in progress.

You must register a client to RM before the client can issue any other RM requests. After the client is registered, it must participate in any IMSplex-wide processes that are performed. You must register the client to all RMs that are active in the IMSplex. If registration to an RM fails, you must deregister the client from any RMs to which the client had successfully registered. If an RM fails, register with it when it comes back up.

You can register the same client multiple times. For example, you might need to specify the resource list for the client after the client is already registered. Optionally, register resource types to RM along with the client to define the resource types to RM and associate a name type with each resource type. You must register resource types before you can specify them in other requests. You cannot register the client if the resource type and name type associations do not match those already registered previously.

Resource-processing clients, such as the IMS control region, issue this request.

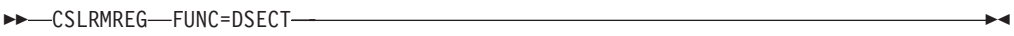
This request is supported in assembler language.

CSLRMREG Syntax

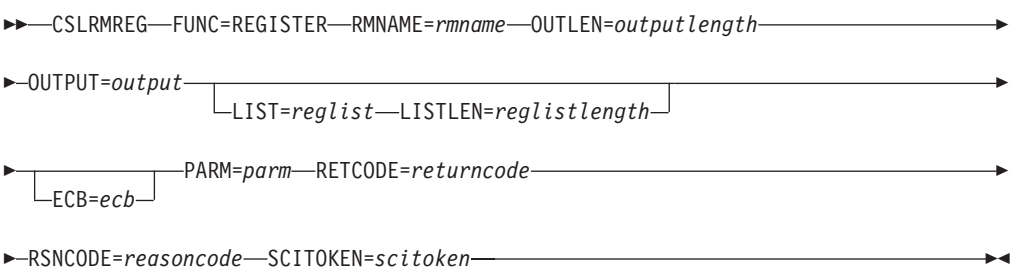
The syntax for the CSLRMREG request follows.

CSLRMREG DSECT Syntax: Use the DSECT function of a CSLRMREG request to include the following inputs and outputs in your program:

- Equate (EQU) statements for the length of the CSLRMREG parameter list
- The CSLRMREG return codes, reason codes, and completion codes
- The CSLRREGL DSECT to map the input registration list
- The CSLRREGO DSECT to map the register output



CSLRMREG REGISTER Syntax: Use the CSLRMREG request to register a client to RM and, optionally, to register the client's resource types and associated name types to RM.



CSLRMREG Parameters

The parameters for the CSLRMREG request follow.

ECB=*symbol*
ECB=*(r2-r12)*

(Optional) - Specifies the address of a z/OS ECB used for asynchronous requests. When the request is complete, the ECB specified is posted. If an ECB is not specified, the task is suspended until the request is complete. If an ECB is specified, the invoker of the request must issue a WAIT (or equivalent) after receiving control from CSLRM REG before using or examining any data returned by this request (including the RETCODE and RSNCODE fields).

LIST=*symbol*
LIST=*(r2-r12)*

(Optional) - Specifies the registration list built by the caller. Each list entry is a separate resource type registration. If a registration list is specified when no resource structure is defined, it is ignored.

The CSLRREGL macro maps the registration list entry. The list contains a header and one or more list entries. The list entries must reside in contiguous storage. Each registration list entry contains the following:

- Resource type
- Name type

LISTLEN=*symbol*
LISTLEN=*(r2-r12)*

(Optional) - Specifies the 4-byte registration list length. LISTLEN is required if LIST is specified.

OUTLEN=*symbol*
OUTLEN=*(r2-r12)*

(Required) - Specifies a 4-byte field to receive the length of the output returned by the CSLRMREG request. OUTLEN contains the length of the output pointed to by the OUTPUT= parameter.

The output length is zero if no output is built, for example, if an error is detected before any output can be built.

OUTPUT=*symbol*
OUTPUT=*(r2-r12)*

(Required) - Specifies a 4-byte field to receive the address of the variable length output returned by the CSLRMREG request. The output contains a header and zero, one, or more registration entries for registrations that were attempted. The output length is returned in the OUTLEN= field.

The output address is zero if no output was built, for example, if an error was detected before any output could be built.

The CSLRREGO macro maps the output that is returned. The output contains a header and zero, one, or more list entries. The output header contains the following:

- Eyecatcher
- Output length
- CSLRREGO version
- CSLRREGO header length (offset to start of entries)
- CSLRREGO entry length
- Registration list count
- Timestamp

- Registration status
- Structure version

Each output entry represents a registration request that was attempted. Each entry contains the following:

- Resource type
- Name type
- Completion code for the registration request. Completion codes are mapped by CSLRRR. Possible completion codes are:

X'00000000'
Register succeeded.

X'00000008'
Invalid resource type. The resource type cannot be zero.

X'0000000C'
Invalid name type. The name type cannot be zero, or the resource type is already defined with a different name type.

PARM=*symbol*
PARM=*(r2-r12)*
(Required) - Specifies the CSLRMREG parameter list. The length of the parameter list must be equal to the parameter list length EQU value defined by RREG_PARMLN.

RETCODE=*symbol*
RETCODE=*(r2-r12)*
(Required) - specifies a 4-byte field to receive the return code on output. RM return codes are defined in CSLRRR. SCI return codes are defined in CSLSRR. Possible return codes are described in Table 50.

RMNAME=*symbol*
RMNAME=*(r2-r12)*
(Required) - Specifies an 8-byte RM name to which to send the registration request.

RSNCODE=*symbol*
RSNCODE=*(r2-r12)*
(Required) - Specifies a 4-byte field to receive the reason code on output. RM reason codes are defined in CSLRRR. SCI reason codes are defined in CSLSRR. Possible reason codes are described in Table 50.

SCITOKEN=*symbol*
SCITOKEN=*(r2-r12)*
(Required) - Specifies a 16-byte field containing the SCI token. This token uniquely identifies this connection to SCI. The SCI token was returned by a successful `CSLSCREG FUNC=REGISTER` request.

CSLRMREG Return and Reason Codes

Table 50 lists the return and reason codes that can be returned on a CSLRMREG request. Also included is the meaning of a reason code (that is, what possibly caused it).

Table 50. CSLRMREG Return and Reason Codes

Return Code	Reason Code	Meaning
X'00000000'	X'00000000'	The request completed successfully.

Table 50. CSLRMREG Return and Reason Codes (continued)

Return Code	Reason Code	Meaning
X'03000004'	X'00001100'	The request completed successfully but the LIST is ignored. No resource structure is defined.
X'03000008'	X'00002100'	The registration-list length is invalid.
	X'00002108'	The registration-list address is invalid.
	X'00002110'	The list version in the list header (REGL_PVER) is zero, which is invalid. The list version must be set in the list header to the maximum list version (REGL_PVERMAX).
	X'00002114'	The list header length cannot be zero or greater than the list length that was passed in. The list header length (REGL_HDRLEN) must be set in the list header to the list header length.
X'0300000C'	X'00003000'	The request is valid for at least one list entry, but not all. The registration for the valid list entries is not performed and the client registration is rejected. Check the completion code in each list entry in the OUTPUT buffer for individual errors.
	X'00003004'	The request failed for all entries. Check the completion code in each list entry in the OUTPUT buffer for individual errors.
X'03000010'	X'00004010'	The client is not authorized.
	X'00004100'	The requested version is not supported. The client compiled with a version of CSLRMREG that is not supported by RM. All RMs must be migrated to a new release before IMS is migrated to a new release that uses a new CSLRMREG function.
	X'00004104'	The list version is not supported. The client created the registration list at a version that is not supported by RM. All RMs must be migrated to a new release before the client is migrated to a new release that uses a new CSLRMREG function.
X'03000014'	X'00005000'	Storage allocation for the register output buffer failed.
	X'00005100'	Storage allocation for CQSUPD buffer failed.
	X'00005200'	CQS request resulted in an unexpected error.
	X'00005204'	CQS request failed because RM incorrectly built request input.
	X'00005110'	The client block allocation failed.

CSLRMUPD: Update Resources

CSLRMUPD creates a resource if it does not exist, or updates a resource if it does exist (as long as the version specified matches the version of the resource). A resource can be created or updated with or without client data.

This request is supported in assembler language.

CSLRMUPD Syntax

The syntax for the CSLRMUPD request follows.

CSLRMUPD DSECT Syntax: Use the DSECT function of a CSLRMUPD request to include the following inputs and outputs in your program:

- Equate (EQU) statements for the length of the CSLRMUPD parameter list
- The CSLRMUPD return codes, reason codes, and completion codes
- The CSLRUPDL DSECT to map the input update list
- The CSLRUPDO DSECT to map the update output

```
►►—CSLRMUPD—FUNC=DSECT———————————————————————————————————————►◄
```

CSLRMUPD UPDATE Syntax: Use the CSLRMUPD request to create or update a uniquely named resource on a resource structure.

```
►►—CSLRMUPD—FUNC=UPDATE—PARM=parm—LIST=updlist—LISTLEN=updlistlength———————►
►—OUTPUT=output—OUTLEN=outputlength———————————————————————————————————►
                                    └ECB=ecb┘  └RETNAME=returnname┘
►———————————————————RETCODE=returncode—RSNCODE=reasoncode———————————►
  └RETTOKEN=returntoken┘
►—SCITOKEN=scitoken———————————————————————————————————————————————►◄
```

CSLRMUPD Parameters

The parameters for CSLRMUPD follow.

ECB=*symbol*
ECB=*(r2-r12)*

(Optional) - Specifies the address of a z/OS ECB used for asynchronous requests. When the request is complete, the ECB specified is posted. If an ECB is not specified, the task is suspended until the request is complete. If an ECB is specified, the invoker of the request must issue a WAIT (or equivalent) after receiving control from CSLRM UPD before using or examining any data returned by this request (including the RETCODE and RSNCODE fields).

LIST=*symbol*
LIST=*(r2-r12)*

(Required) - Specifies the update resource list built by the caller. Each list entry is a separate update request. The list length can vary, depending upon the number of list entries and whether they contain DATA2.

The CSLRUPDL macro maps the update resource list entry. The list contains a header and one or more list entries. The list entries must reside in contiguous storage. Each update list entry contains the following:

- Entry length - the update list entry length. The list entry length can vary, depending upon whether DATA2 is specified.
- Resource name - client-defined name of the resource.
- Resource type - the resource type is a client-defined physical grouping of resources on the resource structure. Valid values are 1-255.
- Update options - options that indicate special processing to perform for the update.
- Version - the resource version, which is the number of times the resource has been updated. The version must match the resource's version for an existing resource for the update to succeed. The version must be zero to create a resource.
- Owner - owner of the resource.

- DATA1 - a small piece of client data (fixed length, contained in the adjunct area of a data entry) for the resource to be updated.
- DATA2 length - DATA2 length, if DATA2 is specified.
- DATA2 - a large piece of client data (variable length, contained in one or more data elements of a data entry) associated with the resource to be updated. DATA2 is optional. The maximum size of DATA2 is 61312 bytes (X'EF80').

LISTLEN=*symbol*
LISTLEN=*(r2-r12)*
(Optional) - Specifies the 4-byte update resource list length. LISTLEN is required if LIST is specified.

OUTLEN=*symbol*
OUTLEN=*(r2-r12)*
(Required) - Specifies a 4-byte field to receive the length of the output returned by the CSLRMUPD request. OUTLEN contains the length of the output pointed to by the OUTPUT= parameter.

The output length is zero if no output is built, for example, if an error is detected before any output can be built.

OUTPUT=*symbol*
OUTPUT=*(r2-r12)*
(Required) - Specifies a 4-byte field to receive the address of the variable length output returned by the CSLRMUPD request. The output contains a header and one or more update entries for resource updates that were attempted. The output length is returned in the OUTLEN= field.

The output address is zero if no output was built, for example, if an error was detected before any output could be built.

The CSLRUPDO macro maps the output that is returned. The output contains a header and one or more list entries. The output header contains the following:

- Eyecatcher
- Output length
- CSLRUPDO version
- Timestamp
- Resource entry count
- CSLRUPDO header length (offset to start of entries)
- Minimum entry length (offset to DATA2)

Each output entry represents a resource update that was attempted. Each entry contains the following:

- Output entry length - the list entry length can vary, depending upon whether DATA2 is returned.
- Resource type
- Name type - the name type is a client-defined value associated with a resource type that ensures uniqueness of client-defined resource names within a name type. Valid values are 1-255.
- Resource name
- Version - new resource version, if update succeeded, or the resource version of an existing resource, if the failed because of a version mismatch.
- Owner - resource owner of an existing resource, if the update failed because of a version mismatch and the option to read the owner was set.

- DATA1 - a small piece of client data (fixed length, contained in the adjunct area of a data entry) associated with an existing resource, if the update failed because of a version mismatch and the option to read DATA1 was set.
- DATA2 length - length of large piece of client data associated with an existing resource, if the update failed because of a version mismatch, DATA2 exists, and the option to read DATA2 was set.
- DATA2 - a large piece of client data (variable length, contained in one or more data elements of a data entry) associated with an existing resource, if the update failed because of a version mismatch, DATA2 exists, and the option to read DATA2 was set. The maximum size of DATA2 is 61312 bytes (X'EF80').
- Completion code for the update request - completion codes are mapped by CSLRRR. Possible completion codes are:

X'00000000'
Update request succeeded.

X'00000008'
Invalid resource type.

X'00000010'
Version mismatch. Resource already exists and version specified on input did not match.

X'00000014'
Resource already exists as a different resource type.

X'00000018'
Resource type is not registered. The resource type must be registered using a CSLRMREG request.

X'0000001C'
Resource structure is full.

X'00000024'
Resource structure is unavailable.

X'00000038'
Update failed because of CQS internal error.

X'0000003C'
Update failed because RM incorrectly built the CQSUPD list entry.

X'00000040'
Version mismatch. The resource already exists and the version specified on input did not match. The requestor requested that DATA2 be passed back, but RM encountered an error reading DATA2.

The output buffer is not preallocated by the caller. After the request returns it, this word contains the address of a buffer containing the update output. It is the caller's responsibility to release this storage by issuing the `CSLSCBFR FUNC=RELEASE` request when it is through with the storage. The length of the output is returned in the OUTLEN= field.

PARM=*symbol*
PARM=*(r2-r12)*
(Required) - Specifies the CSLRMUPD parameter list. The length of the parameter list must be equal to the parameter list length EQU value defined by RUPD_PARMLN.

RETCODE=*symbol*

RETCODE=*(r2-r12)*
(Required) - specifies a 4-byte field to receive the return code on output. RM return codes are defined in CSLRRR. SCI return codes are defined in CSLSRR. Possible return codes are described in Table 51.

RETNAME=*symbol*
RETNAME=*(r2-r12)*
(Optional) - Specifies an 8-byte field to receive the RM name returned to the caller. This is the CSL member name of the target RM address space to which SCI sent the request.

RETTOKEN=*symbol*
RETTOKEN=*(r2-r12)*
(Optional) - Specifies a 16-byte field to receive RM's SCI token returned to the caller. This is the SCI token for the target RM address space to which SCI sent the request.

RSNCODE=*symbol*
RSNCODE=*(r2-r12)*
(Required) - Specifies a 4-byte field to receive the reason code on output. RM reason codes are defined in CSLRRR. SCI reason codes are defined in CSLSRR. Possible reason codes are described in Table 51.

SCITOKEN=*symbol*
SCITOKEN=*(r2-r12)*
(Required) - Specifies a 16-byte field containing the SCI token. This token uniquely identifies this connection to SCI. The SCI token was returned by a successful `CSLSCREG FUNC=REGISTER` request.

CSLRMUPD Return and Reason Codes

Table 51 lists the return and reason codes that can be returned on a CSLRMUPD request. Also included is the meaning of a reason code (that is, what possibly caused it).

Table 51. CSLRMUPD Return and Reason Codes

Return Code	Reason Code	Meaning
X'00000000'	X'00000000'	The request completed successfully.

Table 51. CSLRMUPD Return and Reason Codes (continued)

Return Code	Reason Code	Meaning
X'03000008'	X'00002000'	The client is not registered.
	X'00002100'	The update-list length is invalid.
	X'00002108'	The update-list address is invalid.
	X'0000210C'	One of the list entries contains one of the following invalid list entry lengths: • Zero length • Smaller than the minimum list entry length • Beyond the end of the list passed in • Not on a fullword boundary RM assumes that the rest of the list is invalid.
	X'00002110'	The list version in the list header (UPDL_PVER) is zero, which is invalid. The list version must be set in the list header to the maximum list version (UPDL_PVERMAX).
	X'00002114'	The list header length cannot be zero or greater than the list length that was passed in. The list header length (UPDL_HDRLEN) must be set in the list header to be the list header length.
	X'00002200'	One of the list entries contains an invalid resource type, such as zero. RM assumes the rest of the list is invalid.
	X'0000220C'	One of the entries in the list contains one or more invalid update options. RM assumes the rest of the list is invalid.
	X'00002404'	No resource structure is defined.
X'0300000C'	X'00003000'	The request succeeded for at least one list entry, but not all. Check the completion code in each list entry in the OUTPUT buffer for individual errors.
	X'00003004'	The request failed for all entries. Check the completion code in each list entry in the OUTPUT buffer for individual errors.
	X'00003008'	The request failed for one or more list entries and all failures were version mismatches. Check the completion code in each list entry in the OUTPUT buffer for individual errors.
X'03000010'	X'00004000'	The CQS address space is unavailable. Retry the request to attempt routing the request to another RM with an available CQS.
	X'00004100'	The requested version is not supported. The client compiled with a version of CSLRMUPD that is not supported by RM. All RMs must be migrated to a new release before IMS is migrated to a new release that uses a new CSLRMUPD function.
	X'00004104'	The list version is not supported. The client created the update list at a version level that is not supported by RM. All RMs must be migrated to a new release before the client is migrated to a new release that uses a new CSLRMUPD function.

Table 51. CSLRMUPD Return and Reason Codes (continued)

Return Code	Reason Code	Meaning
X'03000014'	X'00005000'	Storage allocation for the output buffer failed. The resource updates might or might not have succeeded.
	X'00005100'	Storage allocation for CQSUPD buffer failed.
	X'00005200'	CQS request resulted in unexpected error.
	X'00005204'	The CQS request failed because RM incorrectly built the request input.

CSL RM Directives

An RM directive is a function that RM defines that can be sent as a message to RM clients, informing the RM clients of work to be processed. After a *resource processing client* is registered to RM, RM can direct that client to perform RM functions, or directives. RM issues the CSLSCMSG request to send a directive to a client. A resource processing client is any system that manages resources and uses RM to manage global information about those resources.

RM directives are always issued in message protocol (PROTOCOL=MSG), that is, asynchronously; RM therefore expects no response from the RM client, and it continues processing without waiting for a response. The RM client is responsible for determining whether or not to take any action in response to the directive. If the client does not respond, the directive times out.

The CSLRMDIR macro maps the RM directives. The SCI Input exit routine's INXP_MBRPLPTR field points to the CSLRMDIR parameter list. For more information on the SCI Input exit parameter list, see "CSL SCI Input Exit Parameter List" on page 164.

RM directives are defined in the CSLRMDIR macro, which includes the following:

- Repopulate structure (RDIR_STRPOPD)
- Structure failed (RDIR_STRFAILD)
- Process step (RDIR_PRSTEPD)
- Process step response (RDIR_PRRESPD)

The directives and their parameters are described in these topics:

- "CSL RM Repopulate Structure Directive"
- "CSL RM Structure Failed Directive" on page 146
- "CSL RM Process Step Directive" on page 146
- "CSL RM Process Step Response Directive" on page 148

CSL RM Repopulate Structure Directive

If an RM detects a structure failure, it sends the Repopulate Structure directive to all resource processing clients after the structure fails and is reallocated. The client then repopulates the structure. A client can receive this directive from all RMs to which it is registered. If it receives directives from multiple RMs to repopulate the structure after having already done so, it can ignore those requests after confirming that the directives apply to the same structure name and version.

The parameters for the Repopulate Structure directive follow.

RDIR_STRPOP
Identifies the start of the repopulate structure directive.

RDIR_STNAMLEN=*length*
Contains the length of the structure name.

RDIR_STNAMPTR=*address*
Contains the address of the structure name.

RDIR_STVERLEN=*length*
Contains the length of the structure version.

RDIR_STVERPTR=*address*
Contains the address of the structure version.

RDIR_STRPOPLN=*length*
Contains the length of the repopulate structure.

CSL RM Structure Failed Directive

The Structure Failed directive is sent to a resource processing client when the resource structure fails and cannot be reallocated. In this situation, the client cannot make any more resource requests until the problem is corrected. A client can receive this directive from all RMs to which it is registered. If it receives directives from multiple RMs, it can ignore duplicate requests after confirming that the directives apply to the same structure name and version.

The parameters for the Structure Failed directive follow.

RDIR_STRFAIL
Identifies the start of the structure failed directive.

RDIR_SFNAMLEN=*length*
Contains the length of the structure name.

RDIR_SFNAMPTR=*address*
Contains the address of the structure name.

RDIR_SFVERLEN=*length*
Contains the length of the structure version.

RDIR_SFVERPTR=*address*
Contains the address of the structure version.

RDIR_STRFAILN=*length*
Contains the length of the structure failed directive.

CSL RM Process Step Directive

The Process Step directive is sent to a resource processing client when a process step needs to be performed.

The parameters for the Process Step directive follow.

RDIR_PRSTEP
Identifies the start of the Process Step directive.

RDIR_PSTKNLEN=*length*
Contains the length of the process token (PRCTOKEN), which uniquely identifies the IMSplex-wide process. PRCTOKEN is returned after the `CSLRMPRI FUNC=INITIATE` request successfully completes. PRCTOKEN can be specified on `CSLRMPRS FUNC=PROCESS`, `CSLRMPRR FUNC=RESPOND`, and `CSLRMPRT FUNC=TERMINATE` requests.

RDIR_PSTKNPTR=*address*
Contains the address of the PRCTOKEN.

RDIR_PSUOWLEN=*length*
Contains the length of the UOWTOKEN, a client-defined UOW that uniquely identifies a process instance. UOWTOKEN also unites the PROCESS INITIATE, PROCESS RESPOND, and PROCESS TERMINATE steps. UOWTOKEN is defined by the `CSLRMPRI FUNC=INITIATE` request and can be specified on `CSLRMPRS FUNC=PROCESS` requests.

RDIR_PSUOWPTR=*address*
Contains the address of the UOWTOKEN.

RDIR_PRCNMLEN=*length*
Contains the length of the process name (PRCNAME), which is defined by the `CSLRMPRI FUNC=INITIATE` request. It can also be specified on the `CSLRMPRS FUNC=PROCESS` and `CSLRMPRT FUNC=TERMINATE` requests.

RDIR_PRCNMPTR=*address*
Contains the address of the PRCNAME.

RDIR_PRCTYPE
The process type is defined by the `CSLRMPRI FUNC=INITIATE` request. It can be specified on the `CSLRMPRS FUNC=PROCESS` and `CSLRMPRT FUNC=TERMINATE` requests. This parameter is passed by value; the length field is always zero.

RDIR_PSNAME
Contains the process step name, which is defined by the `CSLRMPRS FUNC=PROCESS` request. This parameter is passed by value; the length field is always zero.

RDIR_PSDATLEN=*length*
Contains the length of the process step client data (CDATALEN). The client data is passed to the participants in the process step. CDATALEN is specified on the `CSLRMPRS FUNC=PROCESS` request.

RDIR_PSDATPTR=*address*
Contains the address of the process step client data (CDATA).

RDIR_CNAMLEN=*length*
Contains the length of the client name that was registered to SCI by the client that originated the process step (the process step master).

RDIR_CNAMPTR=*address*
Contains the address of the client name that was registered to SCI by the client that originated the process step (the process step master).

RDIR_CTYPE
Identifies the client type that was registered to SCI by the client that originated the process step (the process step master). This parameter is passed by value; the length field is always zero.

RDIR_CSTYPLEN
Contains the length of the client subtype that was registered to SCI by the client that originated the process step (the process step master).

RDIR_CSTYPPTR
Contains the address of the client subtype that was registered to SCI by the client that originated the process step (the process step master).

RDIR_PRSTEPLN
Contains the length of the process step directive.

CSL RM Process Step Response Directive

The Process Step Response directive is sent to RM by a client that is responding to a process step with a CSLRMPRR request.

The parameters for the Process Step Response directive follow.

RDIR_PRRESP
Identifies the start of the process step response directive.

RDIR_PRTKNLEN=*length*
Contains the length of the process token (PRCTOKEN), which uniquely identifies the IMSplex-wide process. PRCTOKEN is returned after the `CSLRMPRI FUNC=INITIATE` request successfully completes. PRCTOKEN can be specified on `CSLRMPRS FUNC=PROCESS`, `CSLRMPRR FUNC=RESPOND`, and `CSLRMPRT FUNC=TERMINATE` requests.

RDIR_PRTKNPTR=*address*
Contains the address of the of the PRCTOKEN.

RDIR_PROUTLEN=*length*
Contains the length of the process step response output (OUTPUT). The response output is passed back to the originator of the process step. OUTPUT is specified on the `CSLRMPRR FUNC=RESPOND` request.

RDIR_PROUTPTR=*address*
Contains the address of the response output (OUTPUT).

RDIR_PRRCLEN=*length*
Contains the process step response return code (RQSTRC). The return code is specified by the `CSLRMPRR FUNC=RESPOND` request.

RDIR_PRRCPTR=*address*
Contains the address of the process step response return code (RQSTRC).

RDIR_PRRSNLEN=*length*
Contains the length of the process step response reason code (RQSTRSN), which is specified by the `CSLRMPRR FUNC=RESPOND` request.

RDIR_PRRSNPTR=*address*
Contains the address of the process step response reason code (RQSTRSN).

Chapter 5. CSL Structured Call Interface

This topic describes the operations and administrative tasks associated with SCI, one of three CSL managers:

Overview of the CSL SCI

SCI allows IMSplex members to communicate with one another. The communication between IMSplex members can happen within a single z/OS image or among multiple z/OS images. Individual IMS components do not need to know where the other components reside or what communication interface to use.

SCI:

- Routes any requests or messages between the IMS control region, OM, RM, and other members of the IMSplex.
- Registers and deregisters IMSplex members.
- Notifies IMSplex members when a member joins or leaves the IMSplex.

Any IMSplex member that requires SCI services must have an SCI active on its z/OS image.

CSL SCI Definition and Tailoring

This topic describes the how to define and tailor SCI in an IMSplex. You can tailor the following procedures:

You can also use the BPE user exit list PROCLIB member to define the BPE user exit routines to include in the SCI.

CSL SCI Startup Procedure

You can use the SCI startup procedure to dynamically override the settings in the SCI initialization parameters PROCLIB member. The startup procedure is required, but setting values for the execution parameters is optional. A sample startup procedure, shown in Figure 21 on page 150, is called CSLSCI and can be found in IMS.PROCLIB.

```
//*************************************************************
//*     SCI Procedure
//*
//*
//*     Parameters:
//*     BPECFG  - Name of BPE member
//*     SCIINIT - Suffix for your CSLSIxxx member
//*     PARM1   - other override parameters:
//*               ARMRST  - Indicates if ARM should be used
//*               SCINAME - Name of SCI being started
//*
//*               example:
//*               PARM1='ARMRST=Y,SCINAME=SCI1'
//*
//*************************************************************@SCPYRT**
//*                                                             *
//*       Licensed Materials - Property of IBM                  *
//*                                                             *
//*       "Restricted Materials of IBM"                         *
//*                                                             *
//*       5655-C56 (C) Copyright IBM Corp. 2000                 *
//*                                                             *
//*************************************************************@ECPYRT**
//*
//CSLSCI   PROC RGN=3000K,SOUT=A,
//             RESLIB='IMS.SDFSRESL',
//             BPECFG=BPECONFG,
//             SCIINIT=000,
//             PARM1=
//*
//SCIPROC  EXEC PGM=BPEINI00,REGION=&RGN,
//  PARM='BPECFG=&BPECFG,BPEINIT=CSLSINI0,SCIINIT=&SCIINIT,&PARM1'
//*
//STEPLIB  DD  DSN=&RESLIB,DISP=SHR
//         DD  DSN=SYS1.CSSLIB,DISP=SHR
//PROCLIB  DD  DSN=IMS.PROCLIB,DISP=SHR
//SYSPRINT DD  SYSOUT=&SOUT
//SYSUDUMP DD  SYSOUT=&SOUT
//*
```

Figure 21. SCI Sample Startup Procedure

CSL SCI Execution Parameters

You can specify the following parameters as execution parameters on the EXEC statement in the SCI startup procedure. Certain parameters that are required for SCI address space initialization can also be specified in the SCI initialization parameters PROCLIB member.

ARMRST= Y | N

Specifies whether or not the z/OS Automatic Restart Manager (ARM) should be used to restart the SCI address space after an abend. **Y** (yes) specifies that ARM should be used. The SCI address space is restarted by ARM after most system failures. **N** (no) specifies that ARM should not be used. The SCI address space is not restarted by ARM after any failures.

This is an optional execution parameter. If specified, it overrides the value specified in the CSLSIxxx PROCLIB member. If not specified, the value in the CSLSIxxx PROCLIB member is used.

For more information on ARM, see “Using the z/OS Automatic Restart Manager with the CSL” on page 29.

BPECFG=
Specifies an 8-character name for the BPE configuration parameters PROCLIB member. This parameter can be specified only as an execution parameter. If a PROCLIB member is not specified, BPE uses default values for all parameters. This parameter is optional. If not specified, the BPE defaults are no user exits, a trace level of error, and US English as the language.

BPEINIT=CSLSINI0
Specifies the name of the module that contains SCI start up values required by BPEINI00 to start an SCI address space. For SCI, this value must be CSLSINI0. This parameter can only be specified as an execution parameter. This is a required parameter.

FORCE=()
Specifies that SCI is to clean up the global interface storage. FORCE is an optional parameter and has no default. The keywords are:

ALL SCI should delete all of the global storage, including control blocks and routines. This keyword is required.

SHUTDOWN
SCI should shut down after cleaning the global storage. This keyword is optional.

No local IMSplex members can be active when the FORCE keyword is used. If a member is active, results are unpredictable. Use the FORCE keyword in the following situations:

- When an IMSplex managed by an SCI on one image will be managed by a different SCI. For example, PLEX1 is managed by SCI1. If SCI1 becomes inactive, PLEX1 will be managed by SCI2. Before SCI2 is started, use FORCE(ALL,SHUTDOWN) on SCI1 to clean the global storage.
- When an SCI will not be reactivated on an image. To clean the global storage, reactive that SCI one final time using the FORCE(ALL, SHUTDOWN) keyword.

SCIINIT=
Specifies a 3-character suffix for the SCI initialization parameters PROCLIB member, CSLSIxxx. This parameter can be specified only as an execution parameter. The default suffix is 000.

SCINAME=*scimbrname*
Specifies the name for the SCI address space. This is an optional 1- to 6-character name. If specified, it overrides the value specified in the CSLSIxxx PROCLIB member. You must specify this parameter either as an execution parameter or in the CSLSIxxx PROCLIB member. This name is used to create the SCIID which is used in SCI processing. The 8-character SCIID is the SCINAME followed by the characters “SC”. Trailing blanks in the SCINAME are deleted and the SCIID is padded with blanks. For example, if SCINAME=ABC then SCIID=“ABCSC ”.

BPE Considerations for the CSL SCI

Use the SCI BPE user exit list PROCLIB member to define SCI user exits to BPE. The member is the PROCLIB member specified by the EXITMBR= parameter in the BPE configuration parameter PROCLIB member.

Use the user exit list PROCLIB member to specify the modules to be called for specific exit types. Each user exit type can have one or more exit modules associated with it. Use the EXITDEF statement to define the user exit modules to be called for a given exit type.

The BPE user exit PROCLIB member and BPE configuration PROCLIB member are described in *IMS Version 9: Base Primitive Environment Guide and Reference*.

A sample SCI user exit list PROCLIB member is shown in Figure 22.

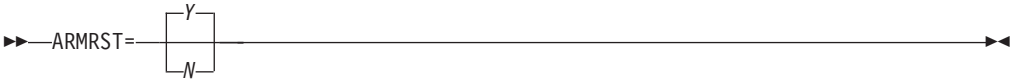
```
********************************************************************
* SCI USER EXIT LIST PROCLIB MEMBER                                *
********************************************************************
#------------------------------------------------------------------#
# DEFINE 1 SCI CLIENT CONNECTION USER EXIT: ZSCLNCN0               #
# WITH AN ABEND LIMIT OF 8.                                        #
#------------------------------------------------------------------#
EXITDEF(TYPE=CLNTCONN,EXITS=(ZSCLNCN0),ABLIM=8,COMP=SCI)

#------------------------------------------------------------------#
# DEFINE 1 SCI INIT/TERM USER EXIT: ZSINTM00                       #
#------------------------------------------------------------------#
EXITDEF(TYPE=INITTERM,EXITS=(ZSINTM00),COMP=SCI)
```

Figure 22. Sample SCI User Exit List PROCLIB Member

CSL SCI Initialization Parameters PROCLIB Member

Use the CSLSIxxx PROCLIB member to specify parameters related to initialization of the SCI address space. Certain parameters within CSLSIxxx can be overridden using SCI execution parameters.

The PROCLIB member consists of one or more fixed-length character records (the configuration data set can be of any LRECL greater than eight, but it must be fixed record format). The rightmost-eight columns are ignored but can be used for sequence numbers or any other notation. Keyword parameters can be coded in the remaining columns in free format, and can contain leading and trailing blanks. You can specify multiple keywords in each record; use commas or spaces to delimit keywords. Statements that begin with a "*" or "#" in column 1 are comment lines and are ignored. Additionally, comments can be included anywhere within a statement by enclosing them between " /* "and "*/", for example, `/* PROCLIB comments */`. Values coded in this PROCLIB member are case-sensitive. In general, you should use upper case for all parameters.



ARMRST= Y | N

Specifies whether or not the z/OS Automatic Restart Manager (ARM) should be used to restart the SCI address space after an abend. **Y** (yes) specifies that ARM should be used. The SCI address space is restarted by ARM after most system failures. **N** (no) specifies that ARM should not be used. The SCI address space is not restarted by ARM after any failures. For more information on ARM, see "Using the z/OS Automatic Restart Manager with the CSL" on page 29.

IMSPLEX()
Specifies definitions for an IMSplex managed by SCI. IMSPLEX is a required parameter. There is no default. Only one IMSPLEX keyword can be specified. The IMSPLEX keyword must precede the left parenthesis. The IMSPLEX definition parameters follow:

NAME=
Specifies a 1- to 5-character name that specifies the IMSplex group name. SCI concatenates this name to "CSL" to create the IMSplex group name. All OM, RM, SCI, IMS, and other address spaces that are in the same IMSplex must specify the same name. This is done by specifying the same name for the IMSPLEX= parameter in the CSLOIxxx, CSLRIxxx, CSLSIxxx, and DFSCGxxx PROCLIB members

SCINAME=*scimbrname*
Specifies the name for the SCI address space. This is an optional 1-6 character name. You must specify this parameter either as an execution parameter or in the CSLSIxxx PROCLIB member. This name is used to create the SCIID which is used in SCI processing. The 8-character SCIID is the SCINAME followed by the characters "SC". Trailing blanks in the SCINAME are deleted and the SCIID is padded with blanks. For example, if SCINAME=ABC then SCIID="ABCSC ".

FORCE=()
Specifies that SCI is to clean up the global interface storage. FORCE is an optional parameter and has no default. The keywords are:

ALL SCI should delete all of the global storage, including control blocks and routines. This keyword is required.

SHUTDOWN
SCI should shut down after cleaning the global storage. This keyword is optional.

No local IMSplex members can be active when the FORCE keyword is used. If a member is active, results are unpredictable. Use the FORCE keyword in the following situations:

- When an IMSplex managed by an SCI on one image will be managed by a different SCI. For example, PLEX1 is managed by SCI1. If SCI1 becomes inactive, PLEX1 will be managed by SCI2. Before SCI2 is started, use FORCE(ALL,SHUTDOWN) on SCI1 to clean the global storage.
- When an SCI will not be reactivated on an image. To clean the global storage, reactive that SCI one final time using the FORCE(ALL, SHUTDOWN) keyword.

A sample CSLSIxxx PROCLIB member is shown in Figure 23 on page 154.

```
*********************************************************************
* SCI INITIALIZATION PROCLIB MEMBER                                 *
*********************************************************************

ARMRST=Y                /* ARM should restart SCI on failure        */
IMSPLEX(NAME=PLEX1)     /* IMSplex name (CSLPLEX1)                  */
SCINAME=SCI1            /* SCI name (SCIID = SCI1SC)                */
```

Figure 23. Sample CSLSIxxx PROCLIB Member

CSL SCI Administration

This topic describes the administrative tasks associated with SCI:

- "Starting the CSL SCI"
- "Shutting Down the CSL SCI"
- "CSL SCI Security" on page 155

Starting the CSL SCI

SCI is started as a started procedure or with JCL. To start an SCI address space with a started procedure, issue the z/OS `START` command as follows:

```
S scijobname
```

In this example, `scijobname` is the job name of the SCI address space to be started. For information on how to start SCI as a started procedure, see "CSL SCI Startup Procedure" on page 149.

After SCI is started, if it is abnormally terminated, it can be restarted using the z/OS Automatic Restart Manager (ARM). SCI must complete initialization for ARM to restart the address space if an abend occurs. Use of ARM to restart SCI is the default.

Shutting Down the CSL SCI

Recommendation: Although you can shut down SCI by itself, IBM recommends that you shut down SCI by shutting down the CSL as one unit. For information about shutting down the CSL, see "Shutting Down the CSL" on page 26.

To shut down RM by itself, issue one of the following:

- the CSLZSHUT request, described in "CSLZSHUT: Shut Down Request" on page 26
- the z/OS STOP command:

  ```
  P scijobname
  ```

In this example, `scijobname` is the job name of the SCI address space to stop. If no clients are connected to SCI, SCI shuts down. However, if there are registered members on the local z/OS image, message CSL0300I is issued, and SCI does not process any new requests or messages from local members. After all in-flight requests have completed or timed out, the SCI address space terminates.

Before shutting down an RM, consider the reasons for shutting down and how shutting down OM can impact other IMSplex members. For more information, see "Shutting Down the CSL" on page 26.

CSL SCI Security

When a client issues the CSLSCREG request to register with SCI, SCI first determines whether the address space is authorized to register with SCI. It does this by issuing a `RACROUTE REQUEST=AUTH` call. RACF (or an equivalent security product) checks the user ID of the address space issuing the CSLSCREG request; the user ID must have at least update authority to register to SCI.

The security administrator can define profiles in the FACILITY class to control SCI registration. Profile names must be of the form `CSL.imsplex_name`, where `imsplex_name` is:

- The name of the IMSplex with RACF (or an equivalent security product) protection.
- The IMSplex name as defined on the IMSPLEX parameter in the CSLSIxxx PROCLIB member, with "CSL" added as a prefix to the name.

Figure 24 defines a profile for SCI to prevent users other than SCIUSER1 and SCIUSER2 from registering to SCI.

```
RDEFINE FACILITY CSL.CSLPLEX1 UACC(NONE)
PERMIT CSL.CSLPLEX1 CLASS(FACILITY) ID(SCIUSER1) ACCESS(UPDATE)
PERMIT CSL.CSLPLEX1 CLASS(FACILITY) ID(SCIUSER2) ACCESS(UPDATE)
SETROPTS CLASSACT(FACILITY)
```

Figure 24. FACILITY Profile Example

CSL SCI User Exit Routines

SCI user exits allow you to customize and monitor the SCI environment. They are written and supplied by the user. No sample exits are provided.

SCI uses BPE services to call and manage its user exits. BPE enables you to externally specify the user exit modules to be called for a particular user exit type by using EXITDEF= statements in the BPE user exit list PROCLIB members. BPE also provides a common user exit runtime environment for all user exits. This environment includes a standard user exit parameter list, callable services, static and dynamic work areas for the exits, and a recovery environment for user exit abends. For more information about the BPE user exit interface, see the *IMS Version 9: Base Primitive Environment Guide and Reference*.

CSL SCI Client Connection User Exit

This exit is called when a client connects (registers) or disconnects (deregisters) from SCI. It is also called when a client issues the CSLSCRDY (ready) and the CSLSCQSC (quiesce) requests. This exit is optional.

This exit is called for the following events:

- After a client has successfully connected to SCI.
- After a client has successfully completed the Ready request to SCI.
- After a client has successfully completed the Quiesce request to SCI.
- After a client has successfully disconnected normally or abnormally from SCI.

This exit is defined as TYPE=CLNTCONN in the EXITDEF statement in the BPE user exit list PROCLIB member. You can specify one or more user exits of this type. When this exit is invoked, all user exits of this type are driven in the order specified

by the EXITS= keyword. For more information on how to define user exit module names, see the SCI BPE user exit list PROCLIB member information in the *IMS Version 9: Base Primitive Environment Guide and Reference*.

This exit is invoked amode 31 and should be reentrant.

Contents of Registers on Entry

Register	Contents
1	Address of BPE user exit parameter list (mapped by macro BPEUXPL).
13	Address of the first of 2 prechained 72-byte save areas. These save areas are chained according to standard z/OS save area linkage convention. The first save area can be used by the exit to save registers on entry. The second save area is for use by routines called from the user exit.
14	Return address.
15	Entry point of exit routine.

On entry to the Client Connection exit, register 1 points to a standard BPE user exit parameter list. Field UXPL_EXITPLP in this list contains the address of the SCI Client Connection user exit parameter list, which is mapped by macro CSLSCLX. Field UXPL_COMPTYPEP in this list points to the character string “SCI”, indicating an SCI address space.

Table 52 describes the user exit parameter lists for SCI:

- client connection
- client disconnect
- client ready
- client quiesce

SCI Client Connection User Exit Parameter List: Table 52 lists the user exit parameter list for SCI client connection. Included are the field name, the offset value and length, both in hexadecimal, how the field is used, and a brief description of the field.

Table 52. SCI Client Connection User Exit Parameter List

Field Name	Offset	Length	Field Usage	Description
SCLX_PVER	X'00'	X'04'	Input	Parameter list version number (00000001).
SCLX_FUNC	X'04'	X'04'	Input	Function code: • **1** Client connect. • **2** Client disconnect. • **3** Client ready. • **4** Client quiesce.
SCLX_MBRNAME	X'08'	X'08'	Input	Client (IMSplex member) name.
SCLX_MBRTYPE	X'10'	X'02'	Input	IMSplex member type (mapped by CSLSTPIX).
SCLX_FLAG1	X'12'	X'01'	Input	Flag byte: • X'80' - Client disconnect is abnormal. • X'40' - Client is authorized.
	X'13'	X'01'	None	Reserved.
SCLX_MBRSTYPE	X'14'	X'08'	Input	IMSplex member subtype.
SCLX_MBRVSN	X'1C'	X'04'	Input	Member version number.
SCLX_JOBNAME	X'20'	X'08'	Input	Member jobname.

Table 52. SCI Client Connection User Exit Parameter List (continued)

Field Name	Offset	Length	Field Usage	Description
SCLX_USERID	X'28'	X'08'	Input	Member userid.
SCLX_OSNAME	X'30'	X'08'	Input	Name of the member's operating system.
SCLX_SCITOKEN	X'38'	X'16'	Input	Member SCI token.
	X'48'	X'04'	None	Reserved.
	X'4C'	X'04'	None	Reserved.

Contents of Registers on Exit

Register	Contents	
15	**Return Code**	**Meaning**
	0	Always zero

All other registers must be restored.

CSL SCI Initialization/Termination User Exit

This exit is called during SCI address space initialization, IMSplex initialization, SCI address space normal termination, or IMSplex normal termination. This exit is not called during SCI address space abnormal termination or IMSplex abnormal termination. This exit is optional.

This exit is called for the following events:

- After SCI has completed initialization
- After each IMSplex has initialized
- When SCI is terminating normally
- When an IMSplex is terminating normally

This exit is defined as TYPE=INITTERM in the EXITDEF statement in the BPE user exit list PROCLIB member. You can specify one or more user exits of this type. When this exit is invoked, all user exits of this type are driven in the order specified by the EXITS= keyword. For more information on how to define user exit module names, see the SCI BPE user exit list PROCLIB member information in the *IMS Version 9: Base Primitive Environment Guide and Reference*.

This exit is invoked amode 31 and should be reentrant.

Contents of Registers on Entry

Register	Contents
1	Address of BPE user exit parameter list (mapped by macro BPEUXPL).
13	Address of the first of 2 prechained 72-byte save areas. These save areas are chained according to standard z/OS save area linkage convention. The first save area can be used by the exit to save registers on entry. The second save area is for use by routines called from the user exit.
14	Return address.
15	Entry point of exit routine.

On entry to the Initialization/Termination exit, register 1 points to a standard BPE user exit parameter list. Field UXPL_EXITPLP in this list contains the address of the

SCI Initialization/Termination user exit parameter list, which is mapped by macro CSLSITX. Field UXPL_COMPTYPEP in this list points to the character string “SCI” indicating an SCI address space.

SCI Init/Term User Exit Parameter List--SCI Initialization: Table 53 lists the user exit parameter list for SCI initialization. Included are the field name, the offset value and length, both in hexadecimal, how the field is used, and a brief description of the field.

Table 53. SCI Init/Term User Exit Parameter List--SCI Initialization

Field Name	Offset	Length	Field Usage	Description
SITX_PVER	X’00’	X’04’	Input	Parameter list version number (00000001).
SITX_FUNC	X’04’	X’04’	Input	Function code **1** SCI initialization.

SCI Init/Term User Exit Parameter List--SCI Termination: Table 54 lists the user exit parameter list for SCI termination. Included are the field name, the offset value and length, both in hexadecimal, how the field is used, and a brief description of the field.

Table 54. SCI Init/Term User Exit Parameter List--SCI Termination

Field Name	Offset	Length	Field Usage	Description
SITX_PVER	X’00’	X’04’	Input	Parameter list version number (00000001).
SITX_FUNC	X’04’	X’04’	Input	Function code **2** SCI normal termination.

SCI Init/Term User Exit Parameter List--IMSplex Initialization: Table 55 lists the user exit parameter list for IMSplex initialization. Included are the field name, the offset value and length, both in hexadecimal, how the field is used, and a brief description of the field.

Table 55. SCI Init/Term User Exit Parameter List--IMSplex Initialization

Field Name	Offset	Length	Field Usage	Description
SITX_PVER	X’00’	X’04’	Input	Parameter list version number (00000001).
SITX_FUNC	X’04’	X’04’	Input	Function code **3** IMSplex normal initialization.
SITX_IPLEXNM	X’08’	X’08’	Input	IMSplex name.

SCI Init/Term User Exit Parameter List--IMSplex Termination: Table 56 lists the user exit parameter list for IMSplex termination. Included are the field name, the offset value and length, both in hexadecimal, how the field is used, and a brief description of the field.

Table 56. SCI Init/Term User Exit Parameter List--IMSplex Termination

Field Name	Offset	Length	Field Usage	Description
SITX_PVER	X’00’	X’04’	Input	Parameter list version number (00000001).
SITX_FUNC	X’04’	X’04’	Input	Function code **4** IMSplex normal termination.

Table 56. SCI Init/Term User Exit Parameter List--IMSplex Termination (continued)

Field Name	Offset	Length	Field Usage	Description
SITX_IPLEXNM	X'08'	X'08'	Input	IMSplex name.

Contents of Registers on Exit

Register	Contents	
15	**Return Code**	**Meaning**
	0	Always zero

All other registers must be restored.

CSL SCI Statistics Available through BPE Statistics User Exit

The BPE Statistics user exit can be used to gather both BPE and SCI statistics. Refer to the BPE user exit information of *IMS Version 9: Base Primitive Environment Guide and Reference* for details on the exit and when it is driven.

The following describes SCI statistics that are available to the BPE Statistics User Exit and are returned on a CSLZQRY FUNC=STATS request directed to SCI. When the user exit is driven, field BPESTXP_COMPSTATS_PTR in the BPE Statistics user exit parameter list, BPESTXP, contains the pointer to the SCI statistics header. When the CSLZQRY FUNC=STATS request is driven, the OUTPUT= buffer points to the output area mapped by CSLZQRYO. The output area field ZQYO_STXOFF contains the offset to the SCI statistics header. The header is mapped by CSLSSTX.

SCI Statistics Header CSLSSTX

Table 57 lists the SCI Statistics Header CSLSSTX. Included are the field name, the offset value and length, both in hexadecimal, how the field is used, and a brief description of the field.

Table 57. SCI Statistics Header CSLSSTX

Field Name	Offset	Length	Field Usage	Description
SSTX_ID	X'00'	X'08'	Input	Eyecatcher "CSLSSTX".
SSTX_LEN	X'08'	X'04'	Input	Length of header.
SSTX_PVER	X'0C'	X'04'	Input	Header version number (0000001).
SSTX_PLEXCNT	X'10'	X'04'	Input	Number of IMSplexes for which statistics are available.
SSTX_STATOFF	X'14'	X'04'	Input	Offset to statistics area for first IMSplex. This is the offset from the beginning of CSLSSTX. The offset points to the CSLSST1 area.
SSTX_SST1OFF	X'18'	X'04'	Input	Offset to the SCI request statistics record for activity performed by SCI requests (mapped by macro CSLSST1). The offset is from the start of the statistics area for this IMSplex. Refer to Table 58 on page 160 for a description of the SCI Request statistics record.

Table 57. SCI Statistics Header CSLSSTX (continued)

Field Name	Offset	Length	Field Usage	Description
SSTX_SST2OFF	X'1C'	X'04'	Input	Offset to SCI IMSplex statistics record for activity performed by SCI for an IMSplex (mapped by macro CSLSST2). The offset is from the start of the statistics area for this IMSplex. Refer to Table 59 on page 161 for a description of the SCI IMSplex statistics record.
SSTX_SST3OFF	X'20'	X'04'	Input	Offset to first SCI member statistics record for SCI activity performed by each member in an IMSplex (mapped by the CSLSST3 macro). The offset is from the start of the statistics area for each IMSplex. Refer to Table 60 on page 161.
	X'24'	X'04'	Input	Reserved.
	X'28'	X'04'	None	Reserved.
	X'2C'	X'04'	None	Reserved.

SCI Statistics Record CSLSST1

CSLSST1 contains statistics that are related to requests that are processed by SCI. Table 58 lists the SCI Statistics Record CSLSST1. Included are the field name, the offset value and length, both in hexadecimal, how the field is used, and a brief description of the field.

Table 58. SCI Statistics Record CSLSST1

Field Name	Offset	Length	Field Usage	Description
SST1_ID	X'00'	X'08'	Input	Eyecatcher "CSLSST1".
SST1_LEN	X'08'	X'04'	Input	Length of CSLSTT1 data.
SST1_PVER	X'0C'	X'04'	Input	Statistics Version Number (00000001).
SST1_SCREG	X'10'	X'04'	Input	Number of local registrations.
SST1_RREG	X'14'	X'04'	Input	Number of remote registrations.
SST1_NREG	X'18'	X'04'	Input	Number of notify remote registrations.
SST1_SCRDY	X'1C'	X'04'	Input	Number of local readys.
SST1_RRDY	X'20'	X'04'	Input	Number of remote readys.
SST1_NRDY	X'24'	X'04'	Input	Number of notify remote readys.
SST1_SCQSC	X'28'	X'04'	Input	Number of local quiesces.
SST1_RQSC	X'2C'	X'04'	Input	Number of remote quiesces.
SST1_SCDRG	X'30'	X'04'	Input	Number of normal local deregistrations.
SST1_SCDRGA	X'34'	X'04'	Input	Number of abnormal local deregistrations.
SST1_RDRG	X'38'	X'04'	Input	Number of normal remote deregistrations.
SST1_RDRA	X'3C'	X'04'	Input	Number of abnormal remote deregistrations.
SST1_NABN	X'14'	X'04'	Input	Number of notify abends.
SST1_SCMI	X'40'	X'04'	Input	Number of member initializations.
	X'44'	X'04'	Input	Reserved.
	X'48'	X'04'	Input	Reserved.
	X'4C'	X'04'	Input	Reserved.
	X'50'	X'04'	Input	Reserved.

Table 58. SCI Statistics Record CSLSST1 (continued)

Field Name	Offset	Length	Field Usage	Description
	X'54'	X'04'	Input	Reserved.
	X'58'	X'04'	Input	Reserved.
	X'5C'	X'04'	Input	Reserved.
	X'60'	X'04'	Input	Reserved.
	X'64'	X'04'	Input	Reserved.

SCI Statistics Record CSLSST2

CSLSST2 contains statistics that are related to an IMSplex, but not to a specific request. Table 59 lists the SCI Statistics Record CSLSST2. Included are the offset value and length, both in hexadecimal, how the field is used, and a brief description of the field.

Table 59. SCI Statistics Record CSLSST2

Field Name	Offset	Length	Field Usage	Description
SST2_ID	X'00'	X'08'	Input	Eyecatcher "CSLSST2".
SST2_LEN	X'08'	X'04'	Input	Length of CSLSST2 data.
SST2_PVER	X'0C'	X'04'	Input	Statistics Version Number (00000001).
SST2_PLEXNAME	X'10'	X'08'	Input	IMSplex name.
SST2_SST3CNT	X'18'	X'04'	Input	Number of CSLSST3 records to follow.
SST2_MSGOGOOD	X'1C'	X'04'	Input	Number of successful IXCMSGO calls.
SST2_MSGOBFSH	X'20'	X'04'	Input	Number of IXCMSGO calls with buffer shortage.
SST2_MSGORSSH	X'24'	X'04'	Input	Number of IXCMSGO calls with other resource shortage.
	X'28'	X'04'	Input	Reserved.
	X'2C'	X'04'	Input	Reserved.
	X'30'	X'04'	Input	Reserved.
	X'34'	X'04'	Input	Reserved.
	X'38'	X'04'	Input	Reserved.
	X'3C'	X'04'	Input	Reserved.
	X'40'	X'04'	Input	Reserved.
	X'44'	X'04'	Input	Reserved.

SCI Member Statistics Record CSLSST3

CSLSST3 contains statistics that are related to specific members of an IMSplex. There is one CSLSST3 entry for each registered IMSplex member when statistics are taken. Table 60 lists the SCI Statistics Record CSLSST3. Included are the offset value and length, both in hexadecimal, how the field is used, and a brief description of the field.

Table 60. SCI Member Statistics Record CSLSST3

Field Name	Offset	Length	Field Usage	Description
SST3_ID	X'00'	X'08'	Input	Eyecatcher "CSLSST3".
SST3_LEN	X'08'	X'04'	Input	Length of CSLSST3 data.
SST3_PVER	X'0C'	X'04'	Input	Statistics Version Number (00000001).

Table 60. SCI Member Statistics Record CSLSST3 (continued)

Field Name	Offset	Length	Field Usage	Description
SST3_PLEXNAME	X'10'	X'08'	Input	IMSplex name.
SST3_MBRNAME	X'18'	X'04'	Input	Member name.
SST3_MBRTYPE	X'20'	X'04'	Input	Member type.
SST3_RQSNTBYL	X'24'	X'04'	Input	Number of requests sent by this member to members on this system (local).
SST3_RQSNTBYR	X'28'	X'04'	Input	Number of requests sent by this member to members on remote systems.
SST3_RQSNTTO	X'2C'	X'04'	Input	Number of requests sent to this member by members on this system (local).
SST3_RQRCVBY	X'30'	X'04'	Input	Number of requests received by this member from all sources.
SST3_MGSNTBYL	X'34'	X'04'	Input	Number of messages sent by this member to members on this system (local).
SST3_MGSNTBYR	X'38'	X'04'	Input	Number of messages sent by this member to members on remote systems.
SST3_MBSNTBYM	X'3C'	X'04'	Input	Number of messages sent by this member to multiple members.
SST3_MGSNTTO	X'40'	X'04'	Input	Number of messages sent to this member by members on this system (local).
SST3_MGRCVBY	X'44'	X'04'	Input	Number of messages received by this member from all sources.
SST3_RQSTMOUT	X'48'	X'04'	Input	Number of requests sent to this member that timed out.
SST3_RQSLOST	X'4C'	X'04'	Input	Number of requests sent to this member that were lost due to an abend or lose system.
	X'50'	X'04'	Input	Reserved.
	X'54'	X'04'	Input	Reserved.
	X'58'	X'04'	Input	Reserved.
	X'5C'	X'04'	Input	Reserved.
	X'60'	X'04'	Input	Reserved.
	X'64'	X'04'	Input	Reserved.
	X'68'	X'04'	Input	Reserved.
	X'6C'	X'04'	Input	Reserved.

CSL SCI IMSplex Member Exit Routines

This topic describes the exits that SCI can drive in the address space of a registered IMSplex member. These exit routines allow an IMSplex member to:

- Monitor what address spaces are active members of the IMSplex.
- Receive messages and requests from other members of the same IMSplex.

SCI member exits are written and supplied by an IMSplex member (such as the IMS control region). Each member must write its own exit routines tailored to the needs of that member product, to be supplied as part of the product. No sample SCI exit routines are provided. The exit routines are given control in the member's address space in one of two ways:

- For authorized members (those running in supervisor state, key 0-7), the exits receive control in SRB mode.
- For non-authorized members (those running in problem state or non-key 0-7), the exits receive control as an IRB under the member TCB associated with the SCI registration.

Because each call to a member exit routine runs under its own SRB, the order in which the exits are driven is not guaranteed. It is possible for member exit routines to be driven out of order (different from the order in which SCI scheduled them). Your exit routines must be able to tolerate events that are received out of order. All member exit routine parameter lists contain an 8-byte time stamp in STCK format, which is the time when SCI scheduled the SRB for the exit routine. This time stamp can be used to help determine the original order of events.

The SCI IMSplex member exit routines are:

- "CSL SCI Input Exit Routine"
- "CSL SCI Notify Client Exit Routine" on page 166

CSL SCI Input Exit Routine

The SCI Input exit routine is called whenever there is a message or a request for the IMSplex member.

The IMSplex member loads the exit routine and passes the exit routine address on the CSLSCREG request. The exit is driven in the member's address space, either as an SRB (for authorized members) or as an IRB (for non-authorized members).

Contents of Registers on Entry

Register	Contents
0	Length in bytes of the parameter list pointed to by R1.
1	Address of SCI Input exit parameter list (mapped by macro CSLSINXP).
13	Address of 2 prechained save areas. The first save area can be used by the exit to save registers on entry. The second save area is for use by routines called from the exit.
14	Return address.
15	Entry point of exit routine.

Restriction: All addresses passed to the SCI Input Exit routine are valid only until the exit routine returns to its caller (with the exception of the member parameter list address). These addresses should never be stored and used after the SCI Input Exit routine has returned. Doing so can cause unpredictable results, because the storage pointed to by the addresses might have changed, or it might have been freed. The member parameter list address is the exception to this restriction. It is available until the storage is released by issuing the CSLSCBFR FUNC=RELEASE request (for messages), or the CSLSCRQR FUNC=RETURN request (for requests).

Contents of Registers on Exit

The SCI Input exit routine must preserve the contents of R13; it does not need to preserve any other register's contents. Therefore, it can use the save areas pointed to by R13 for any calls to other services as needed.

Register	Contents

13 The same value it had on entry to the SCI Input exit routine.

15 Return code

- **0** The message or request was successfully received.
- **8** The message or request was not received. If the input data is for a request, SCI sends a response with return code=SRC_PARM (parameter error) and reason code=SRSN_FUNCTION (invalid function). For both messages and requests, SCI releases the storage containing the input data.

CSL SCI Input Exit Parameter List

Table 61 describes the entry parameters for the parameter list header of the Client SCI Input exit routine. The field name is provided, with its offset and length in hexadecimal, and a brief description of the field.

Table 61. Client SCI Input Exit Routine parameter List - parameter List Header

Field Name	Offset	Length	Description
INXP_PVER	X'00'	X'04'	Parameter list version number (00000001).
INXP_PLEN	X'04'	X'04'	Total length of parameter list.
INXP_SCIVSN	X'08'	X'04'	Version of SCI on the system from which this message or request originated.
INXP_EXITPARM	X'0C'	X'08'	Input exit routine member data that was passed to SCI on the CSLSCREG request with the INPUTPARM parameter. If no data was passed on the CSLSCREG request, this field contains zeros.
INXP_PLEXNAME	X'14'	X'08'	IMSplex name.
INXP_TIMESTMP	X'1C'	X'08'	Time stamp representing the time the exit routine was scheduled (in STCK format).
INXP_DATAOFF	X'24'	X'04'	Offset of Message Data Section from the start of the parameter list header.
INXP_SRCOFF	X'28'	X'04'	Offset of Source Member Data Section from the start of the parameter list header

Table 62 describes the entry parameters for the message data of the Client SCI Input exit routine. The field name is provided, with its offset and length in hexadecimal, and a brief description of the field.

Table 62. Client SCI Input Exit Routine parameter List - Message Data

Field Name	Offset	Length	Description
INXP_FUNC	X'00'	X'04'	Function code.
INXP_SFUNC	X'04'	X'04'	Subfunction code.

Table 62. Client SCI Input Exit Routine parameter List - Message Data (continued)

Field Name	Offset	Length	Description
INXP_DATAFL1	X'08'	X'01'	Data Flag. **X'80'** INXP_RQST This bit indicates that the input data is a request. When the receiver of the request has completed processing the request, it must be returned using the CSLSCRQR request. If the bit is not set, the input data is a message. When the receiver of the message has completed processing the message, it should return the storage to SCI using the CSLSCBFR request. **X'40'** INXP_FTYPSND When this bit is on, the function code in INXP_FUNC is defined by the sender. When this bit is off, the function code is defined by the destination.
	X'09'	X'03'	Reserved.
INXP_MBRPLCNT	X'0C'	X'04'	The number of parameters (pairs of lengths and addresses) passed in the member parameter list.
INXP_MBRPLPTR	X'10'	X'04'	The address of the member parameter list.
	X'14'	X'04'	Reserved.
INXP_RQSTTKN	X'18'	X'08'	Request token. This field is valid only if bit INXP_RQST is set (indicating that this is a request). The request token is used to return the request to the sender when issuing the CSLSCRQR request. If INXP_RQST is not set (indicating this is a message), this field is unused.
	X'20'	X'04'	Reserved.
	X'24'	X'04'	Reserved.

Table 63 describes the entry parameters for the input source data of the Client SCI Input exit routine. The field name is provided, with its offset and length in hexadecimal, and a brief description of the field.

Table 63. Client SCI Input Exit Routine parameter List - Input Source Data

Field Name	Offset	Length	Description
INXP_SCITKN	X'00'	X'10'	The SCITOKEN of the IMSplex member that is the source of this data.
INXP_MBRNAME	X'10'	X'08'	The name of the IMSplex member that is the source of this data.
INXP_MBRVSN	X'18'	X'04'	The version of the IMSplex member that is the source of this data. If the source IMSplex member did not pass a MBRVSN on the CSLSCREG request, this field is set to zeros.
INXP_TYPE	X'1C'	X'02'	The IMSplex member type of the IMSplex member that is the source of this data.
	X'1E'	X'02'	Reserved.

Table 63. Client SCI Input Exit Routine parameter List - Input Source Data (continued)

Field Name	Offset	Length	Description
INXP_SUBTYPE	X'20'	X'08'	The subtype of the IMSplex member that is the source of this data. If the source IMSplex member did not pass a SUBTYPE on the CSLSCREG request, this field is set to zeros.
INXP_JOBNAME	X'28'	X'08'	The jobname of the IMSplex member that is the source of this data.
INXP_USERID	X'30'	X'08'	The user ID of the IMSplex member that is the source of this data.
INXP_SRCFL1	X'38'	X'01'	Source Flag **X'80'** This bit indicates that the member that sent this data is authorized.
	X'39'	X'03'	Reserved.
	X'3C'	X'04'	Reserved.
	X'40'	X'04'	Reserved.
	X'44'	X'04'	Reserved.

CSL SCI Notify Client Exit Routine

The SCI Notify exit routine is driven whenever there is a change in the SCI status of an IMSplex member. This allows a member to keep track of the status of other members in the IMSplex.

The IMSplex member loads the exit routine and passes the exit routine address on the CSLSCREG request. The exit is driven in the member's address space, either as an SRB (for authorized members) or as an IRB (for non-authorized members).

The exit is driven whenever an IMSplex member:

- Completes a successful CSLSCREG FUNC=REGISTER
- Completes a successful CSLSCRDY FUNC=READY
- Completes a successful CSLSCQSC FUNC=QUIESCE
- Completes a successful CSLSCDRG FUNC=DEREGISTER
- Terminates without issuing a CSLSCDRG FUNC=DEREGISTER request.
- Is not reachable because the local SCI is not active.

Note that some fields are not available in the Notify exit parameter list when the exit is driven for CSLSCDRG-related events (normal and abnormal termination) and when an IMSplex member is not reachable.

If the local SCI is the IMSplex member for which the Notify exit is being driven, the NXFP_LOCALSCI (X'40') bit in the NXFP_FLAG1 is set. When an SCI terminates, processing on the z/OS image for the IMSplex that was managed by the inactive SCI is limited until the SCI restarts:

- No messages or requests can be sent or received by any local IMSplex member.
- The SCI Notify exit cannot be driven for local IMSplex members for the following events:
  - CSLSCREG FUNC=REGISTER
  - CSLSCRDY FUNC=READY
  - CSLSCQSC FUNC=QUIESCE

- CSLSCDRG FUNC=DEREGISTER (non-authorized member)
- Termination without CSLSCDRG FUNC=DEREGISTER (non-authorized member)

The Notify exit continues to be driven for normal and abnormal deregistrations for authorized members.

- No new members can join the IMSplex on the z/OS image.
- No SCI requests can be processed by local IMSplex members (for example, CSLSCQRY and CSLSCDRG requests).

When SCI restarts on the z/OS image, SCI re-registers each IMSplex member that is still active. The SCITOKEN for each IMSplex member is still valid. The Notify exit routine for each local member is driven for the following events:

- Registration for the local SCI
- Registration and Ready (if appropriate) for the local IMSplex members
- Ready for the local SCI
- Registration and Ready (if appropriate) for IMSplex members that are not local

Events for members that are not local can be scheduled before the Ready for the local SCI; however, events for local members are all scheduled before the SCI Ready event is scheduled. Local IMSplex members should not use SCI services until they have received the Ready event for the local SCI.

Contents of Registers on Entry

Register	Contents
0	Length in bytes of the parameter list pointed to by R1.
1	Address of SCI Notify exit parameter list (mapped by macro CSLSNFXP).
13	Address of 2 prechained save areas. The first save area can be used by the exit to save registers on entry. The second save area is for use by routines called from the exit.
14	Return address.
15	Entry point of exit routine.

Restriction: All addresses passed to the SCI Notify exit routine are valid only until the exit routine returns to its caller. These addresses should never be stored and used after the SCI Notify exit routine has returned. Doing so can cause unpredictable results, because the storage pointed to by the addresses might have changed, or it might have been freed.

Contents of Registers on Exit

The SCI Notify exit routine must preserve the contents of R13; it does not need to preserve any other register's contents. Therefore, it is free to use the save areas pointed to by R13 for any calls to other services as needed.

Register	Contents
13	The same value it had on entry to the SCI Notify exit routine.
15	Return code **0** Always set this to zero.

CSL SCI Notify Exit Parameter List

Table 64 describes the parameter list header of the SCI Notify Client exit routine. The field name is provided, with its offset and length in hexadecimal, and a brief description of the field.

Table 64. SCI Notify Client Exit Routine parameter List Header

Field Name	Offset	Length	Description
NFXP_PVER	X'00'	X'04'	Parameter list version number (00000001).
NFXP_PLEN	X'04'	X'04'	Total length of parameter list.
NFXP_EXITPARM	X'08'	X'08'	Notify exit routine member data that was passed to SCI on the CSLSCREG request with the NOTIFYPARM parameter. If no data was passed on the CSLSCREG request, this field contains zeros.
NFXP_PLEXNAME	X'10'	X'08'	IMSplex name.
NFXP_SCIVSN	X'18'	X'04'	SCI Version
NFXP_TIMESTMP	X'1C'	X'08'	Time stamp representing the time the exit routine was scheduled (in STCK format).
NFXP_SUBJOFF	X'24'	X'04'	Offset of Subject Data Section.
	X'28'	X'04'	Reserved.

Table 65 describes the subject data of the SCI Notify Client exit routine. The field name is provided, with its offset and length in hexadecimal, and a brief description of the field.

Table 65. SCI Notify Client Exit Routine Parameter List - Subject Data

Field Name	Offset	Length	Description
NFXP_SCITKN	X'00'	X'10'	The SCITOKEN of the member that is the subject of this event.
NFXP_EVENT	X'10'	X'02'	The event that initiated this notification. **1** CSLSCREG FUNC=REGISTER **2** CSLSCRDY FUNC=READY **3** CSLSCQSC FUNC=QUIESCE **4** CSLSCDRG FUNC=DEREGISTER **5** Termination without CSLSCDRG FUNC=DEREGISTER **6** The member cannot be reached because the local SCI is not active.
NFXP_FLAG1	X'12'	X'01'	The event that initiated this notification. **X'80'** This bit indicates that the subject of this event is authorized. **X'40'** This bit indicates that the subject of this event is the local SCI.
	X'13'	X'01'	Reserved.
NFXP_MBRNAME	X'20'	X'08'	The Name of the IMSplex member that is the subject of this event.

Table 65. SCI Notify Client Exit Routine Parameter List - Subject Data (continued)

Field Name	Offset	Length	Description
NFXP_MBRVSN	X'28'	X'04'	The Version of the IMSplex member that is the subject of this event. If the subject IMSplex member did not pass a MBRVSN on the CSLSCREG request, this field is set to zeros. This data is not filled in for: • NFXP_EVENT= 4 (normal termination) • NXFP_EVENT= 5 (abnormal termination) • NXFP_EVENT=6 (not reachable)
NFXP_TYPE	X'14'	X'02'	The IMSplex member Type of the IMSplex member that is the subject of this event.
	X'16'	X'02'	Reserved.
NFXP_SUBTYPE	X'18'	X'08'	The Subtype of the IMSplex member that is the subject of this event. If the subject IMSplex member did not pass a SUBTYPE on the CSLSCREG request, this field is set to zeros. This data is not filled in for: • NFXP_EVENT= 4 (normal termination) • NXFP_EVENT= 5 (abnormal termination) • NXFP_EVENT=6 (not reachable)
NFXP_JOBNAME	X'18'	X'08'	The Jobname of the IMSplex member that is the subject of this event. This data is not filled in for: • NFXP_EVENT= 4 (normal termination) • NXFP_EVENT= 5 (abnormal termination) • NXFP_EVENT=6 (not reachable)

Writing a CSL SCI Client

If you want to write a program that participates in an IMSplex (such as an automated operator program), you must first establish a connection to SCI. This allows your IMSplex member to communicate with other IMSplex members. Without a connection to SCI, a program cannot participate in an IMSplex and communicate with other IMSplex members.

To establish a connection with SCI, you can use a subset of the SCI requests described in “CSL SCI Requests” on page 171. These requests establish or terminate a connection with SCI and optionally indicate to SCI that the IMSplex member is in a ready state. When a member is in a ready state, it can have requests and messages routed to it by type.

SCI requests are also used by an IMSplex member to communicate with other IMSplex members and to find out information about those members. IMSplex members communicate with other members by using SCI requests to send messages, requests, and responses to requests. A query request, “CSLSCQRY: Query Request” on page 181 can be used to find out information about the other members of the IMSplex.

Sequence of CSL SCI Requests

Like the OM and RM requests, the SCI requests must be issued in a particular sequence.

The first request is CSLSCREG. The member can then issue CSLSCRDY to tell SCI that it is ready to receive messages and requests that are routed by member type. If a member has storage that is allocated by SCI (for example, a message or an SCI allocated output parameter is received), the SCI buffer release request, CSLSCBFR, can be issued to release the storage.

When a member is ready to terminate, the SCI quiesce request, CSLSCQSC, is used to tell SCI that the member does not want to receive messages and requests that are routed by member type. After the SCI deregistration request, CSLSCDRG, is used to terminate the connection with SCI, the member can no longer participate in the IMSplex.

Table 66 lists the sequence of requests issued by an SCI client. The request is listed with its purpose.

Table 66. Sequence of requests for SCI client

Request	Purpose
CSLSCREG	Register to SCI, which establishes the connection with SCI and enables the member to communicate within the IMSplex.
CSLSCRDY	Readies the member to SCI, which allows SCI to route messages and requests that are routed by member type to this member.
CSLSCBFR	Releases storage allocated for the member by SCI (for example, message data or parameters allocated by SCI from a request).
CSLSCQSC	Quiesces the member to SCI, which tells SCI not to route messages and requests that are routed by member type to this member.
CSLSCDRG	Deregisters the member from SCI which ends the member's connection with SCI.

Advanced CSL SCI Requests

After establishing the connection with SCI, an IMSplex member can use advanced SCI requests to:

- Communicate, or request services, from other IMSplex members.

  A message protocol and a request protocol are provided to facilitate communication among IMSplex members. A message is a one-way communication with another IMSplex member. A request requires that a response be returned to the requesting member.
- Find out information about the other members in the IMSplex.

  A query request, CSLSCQRY, allows an IMSplex member to find out who the other members of the IMSplex are and to obtain information about those IMSplex members.

Table 67 lists the advanced SCI requests with their purpose. These requests can be issued without regard to sequence; however, the IMSplex member issuing the request must have registered to SCI.

Table 67. Advanced SCI requests for IMSplex members

Request	Purpose
CSLSCMSG	Sends a one-way message to another IMSplex member.
CSLSCRQS	Sends a request to another IMSplex member. SCI expects a response to the request.
CSLSCRQR	Sends a response to a previously issued request.

Table 67. Advanced SCI requests for IMSplex members (continued)

Request	Purpose
CSLSCQRY	Issues a query to SCI to find out information about members of the IMSplex.

CSL SCI Requests

Most SCI requests can be issued by any IMSplex member; any member can also receive messages from any other IMSplex member.

The SCI requests described in this topic include:

- "CSLSCBFR: Buffer Return Request"
- "CSLSCDRG: Deregistration Request" on page 173
- "CSLSCMSG: Send Message Request" on page 174
- "CSLSCQRY: Query Request" on page 181
- "CSLSCQSC: Quiesce Request" on page 184
- "CSLSCRDY: Ready Request" on page 186
- "CSLSCREG: Registration Request" on page 187
- "CSLSCRQR Request Return Request" on page 193
- "CSLSCRQS: Send Request Request" on page 196

CSLSCBFR: Buffer Return Request

The CSLSCBFR request releases storage that SCI allocated for an IMSplex member. This storage is allocated to receive either an input message that was sent from another IMSplex member with the CSLSCMSG request, or an output parameter generated from a CSLSCRQS request.

Note: Another macro can invoke CSLSCRQS as part of the code generated by the macro which, in turn, can return an SCI data type. The storage allocated for these parameters must be released with the CSLSCBFR macro. An example of an SCI macro that does this is the CSLSCQRY macro. The OUTPUT parameter specifies the address in storage to receive the address of the buffer that contains the output from the CSLSCQRY. This storage should be released using CSLSCBFR.

CSLSCBFR Syntax

The syntax for the CSLSCBFR request follows.

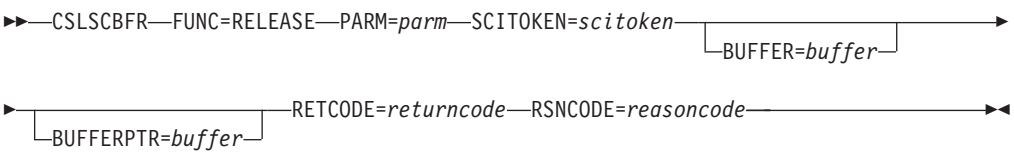
DSECT Syntax: Use the DSECT function of a CSLSCBFR request to include equate (EQU) statements in your program for the CSLSCBFR parameter list length and the CSLSCBFR return and reason codes.

RELEASE Syntax: Use the Release function of CSLSCBFR to release an SCI message buffer or SCI data type buffer. The SCI data type buffer is used for selected output parameters of CSLSCRQS for which SCI allocates storage.

For messages generated from a CSLSCMSG request, the buffer address is the address of the member parameter list that is given to the member input exit in the INXP_MBRPLPTR field in the input exit parm list.

For a response generated from a CSLSCRQS request that uses an SCI data type buffer, the storage is allocated when the request is returned to the IMSplex member that initiated the original request. The buffer address is the address of this storage, which is returned in the field specified by the member on the request.

After the CSLSCBFR request is complete, the storage contained in the message buffer or request response is no longer accessible by the IMSplex member. The CSLSCBFR FUNC=RELEASE request follows.



CSLSCBFR Parameters

The parameters for the CSLSCBFR request follow.

BUFFER=*symbol*
BUFFER=*(r1-r12)*
(Required) - Four-byte parameter that contains the address of a buffer that is to be released.

Either BUFFER or BUFFERPTR is required.

BUFFERPTR=*symbol*
BUFFERPTR=*(r1-r12)*
(Optional) — Four-byte parameter that contains the address of a word in storage that contains the address of the buffer that is to be released.

Either BUFFER or BUFFERPTR is required.

PARM=*symbol*
PARM=*(r1-r12)*
(Required) - Specifies the CSLSCBFR parameter list. The length of the parameter list must be equal to the parameter list length EQU value defined by SBFR_PARMLN.

RETCODE=*symbol*
RETCODE=*(r1-r12)*
(Required) - Specifies a 4-byte field to receive the return code on output. The SCI return codes are defined in CSLSRR. Possible return codes for CSLSCBFR are described in Table 68 on page 173.

RSNCODE=*symbol*
RSNCODE=*(r1-r12)*
(Required) - Specifies a 4-byte field to receive the reason code on output. The SCI reason codes are defined in CSLSRR. Possible reason codes for CSLSCBFR are described in Table 68 on page 173.

SCITOKEN=*symbol*
SCITOKEN=*(r1-r12)*
(Required) - Specifies a 16-byte field containing the SCITOKEN. This token uniquely identifies this IMSplex member's connection to SCI. The SCI token was returned by a successful CSLSCREG FUNC=REGISTER request.

CSLSCBFR Return and Reason Codes

Table 68 lists the return and reason codes that can be returned on a CSLSCBFR macro request. Also included is the meaning of a reason code (that is, what possibly caused it).

Table 68. CSLSCBFR Return and Reason Codes

Return Code	Reason Code	Meaning
X'00000000'	X'00000000'	Request completed successfully.
X'01000008'	X'00002014'	The buffer being released was not an SCI buffer.
	X'00002018'	Invalid SCI token.
	X'00002038'	Parameter list version is invalid.
	X'00002054'	The buffer being released was not an allocated buffer.
X'01000010'	X'00004FFF'	Function is not supported.
X'01000014'	X'00005000'	An SCI internal error occurred.
	X'00005074'	Buffer prefix is damaged on CSLSCBFR call.
	X'00005078'	STORAGE RELEASE failed for SCI buffer on CSLSCBFR call
	X'00005500'	An abend occurred during CSLSCBFR processing.

CSLSCDRG: Deregistration Request

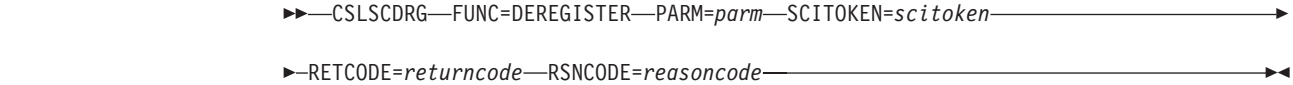
Use the SCI deregistration request to terminate the connection between the IMSplex member and SCI. After successful completion of this request, the SCI token is no longer valid. To make subsequent SCI requests, the IMSplex member must create a new connection with SCI with a CSLSCREG request.

CSLSCDRG Syntax

The syntax for the CSLSCDRG request follows.

CSLSCDRG DSECT Syntax: Use the DSECT function of a CSLSCDRG request to include equate (EQU) statements in your program for the CSLSCDRG parameter list length and the CSLSCDRG return and reason codes.

CSLSCDRG DEREGISTER Syntax: The CSLSCDRG FUNC=DEREGISTER request deregisters the IMSplex member from SCI. After successful completion of the CSLSCDRG FUNC=DEREGISTER request, the SCITOKEN is invalid.



CSLSCDRG Parameters

The parameters for the CSLSCDRG request follow.

PARM=*symbol*
PARM=*(r1-r12)*
(Required) - Specifies the CSLSCDRG parameter list. The length of the parameter list must be equal to the parameter list length EQU value defined by SDRG_LN.

RETCODE=*symbol*
RETCODE=*(r1-r12)*
(Required) - Specifies the address of a 4-byte field to receive the CSLSCDRG return code. The SCI return codes are defined in CSLSRR. Possible return codes for CSLSCDRG are described in Table 69.

RSNCODE=*symbol*
RSNCODE=*(r1-r12)*
(Required) - Specifies the address of a 4-byte field to receive the CSLSCDRG reason code. The SCI reason codes are defined in CSLSRR. Possible reason codes for CSLSCDRG are described in Table 69.

SCITOKEN=*symbol*
SCITOKEN=*(r1-r12)*
(Required) - Specifies a 16-byte field containing the SCITOKEN. This token uniquely identifies this IMSplex member's connection to SCI. The SCI token was returned by a successful CSLSCREG FUNC=REGISTER request.

CSLSCDRG Return and Reason Codes

Table 69 lists the return and reason codes that can be returned on a CSLSCDRG macro request. Also included is the meaning of a reason code (that is, what possibly caused it).

Table 69. CSLSCDRG Return and Reason Codes

Return Code	Reason Code	Meaning
X'00000000'	X'00000000'	The request completed successfully.
X'01000004'	X'00001010'	XCF leave for member failed.
X'01000008'	X'00002018'	Invalid SCI token.
	X'00002038'	Parameter list version is invalid.
X'01000010'	X'00004000'	SCI is not active.
	X'00004014'	CSLSDR00 could not be loaded.
	X'00004018'	There are still outstanding requests during deregistration.
	X'00004FFF'	Function is not supported.
X'01000014'	X'00005000'	An SCI internal error occurred.
	X'00005004'	SCI was unable to add the ESTAE routine.
	X'00005008'	A BPE SVC error occurred.
	X'00005020'	An ENQ resource error occurred.
	X'00005500'	An abend occurred during CSLSCDRG processing.

CSLSCMSG: Send Message Request

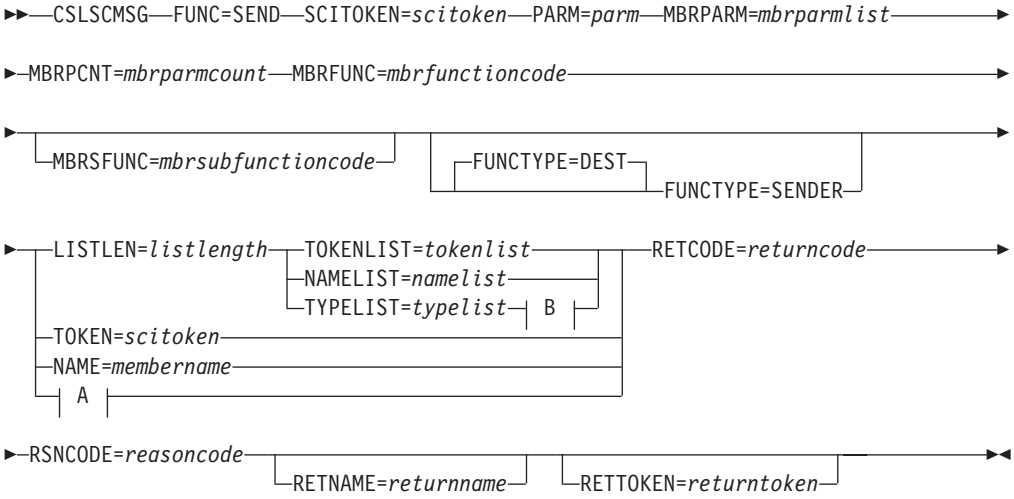
Use the SCI send message request to send a message to one or more other IMSplex members. The target members are specified by SCITOKEN, member name, or member type.

CSLSCMSG Syntax

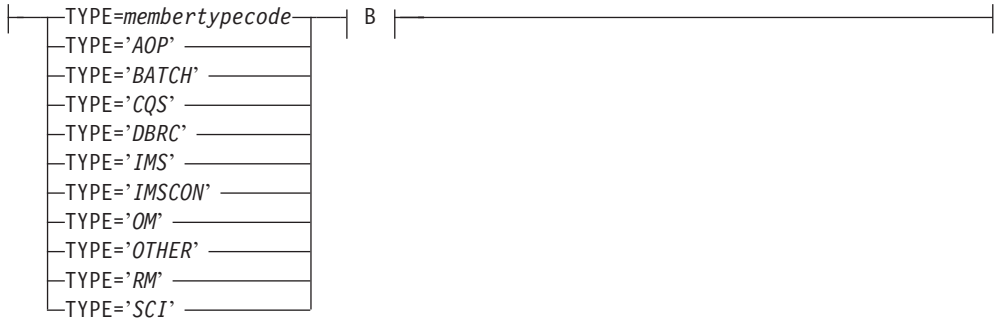
The syntax for the CSLSCMSG request follows.

CSLSCMSG DSECT Syntax: Use the DSECT function of a CSLSCMSG request to include equate (EQU) statements in your program for the CSLSCMSG parameter list length, the IMSplex types and the CSLSCMSG return and reason codes.

CSLSCMSG SEND MESSAGE Syntax: The syntax of the CSLSCMSG FUNC=SEND request is shown below:



A:



B:

CSLSCMSG Parameters

The parameters for the CSLSCMSG request follow.

FUNCTYPE=SENDER
FUNCTYPE=DEST

(Optional) - Specifies that the MBRFUNC and MBRSFUNC are defined by the DEST (destination) of this message or the SENDER of the message. This indicator is passed to the recipient of the message in the SCI Input exit parameter list.

LISTLEN=*<numeric literal>*
LISTLEN=*symbol*

LISTLEN=*(r1-r12)*

(Required if NAMELIST, TOKENLIST or TYPELIST is specified) - Specifies the length of the routing list. The routing list consists of a header and one or more list entries, each entry describing a single message destination (NAMELIST and TOKENLIST) or set of destinations (TYPELIST).

If LISTLEN is a numeric literal, all characters must be numbers. If any character is alphabetic, the parameter will be considered a symbol.

MBRFUNC=*symbol*
MBRFUNC=*(r1-r12)*

(Required) - Specifies a 4-byte member function code that is passed to the destination of the message in the SCI Input exit parameter list. This function code, along with the MBRSFUNC, identifies the message that is being sent.

If MBRFUNC is a symbol, the symbol points to a 4-byte area of storage that contains the function code.

MBRPARM=*symbol*
MBRPARM=*(r1-r12)*

(Required) - Specifies the address of a pre-built parameter list. This parameter list must be built by the messaging module and consists of sets of pairs. Each pair describes a single parameter in the member parameter list and consists of the following:

parameter length
Four-byte parameter that specifies the length of the member parameter.

parameter address
Four-byte parameter that specifies the address of the member parameter.

The two methods for passing parameters in a parameter list are *by address* and *by value*. Both of these methods can be used when passing parameters in a CSLSCMSG request. The pair must be setup so that SCI will handle the parameter properly.

- By address

  To pass a parameter by address, the address of the parameter must be passed in *parameteraddress* and the length of the parameter must be passed in *parameterlength*. SCI will obtain the parameter from *parameteraddress*.

- By value

  To pass a parameter by value, the parameter must be passed in *parameteraddress* and zero must be passed in *parameterlength*. When the length is zero, SCI will copy the value contained in *parameteraddress* to the destination.

Member Parameter List: The user parameters specified here are presented to the IMSplex member that receives the message in the member parameter list, the address of which is contained in the Input exit parameter area field INXP_MBRPLPTR. Each parameter is represented by eight bytes, the first four bytes contain *parameterlength* and the second four bytes contain *parameteraddress* (if *parameteraddress* is an address, the second four bytes point to storage in the local address space, not the requesting address space).

Null Parameters: In some cases the message processing module expects a set number of parameters with a defined order. If a message is to be sent that does not contain all the parameters, null parameters must be sent to ensure the data buffer contains everything that is expected. Null parameters can be sent by

specifying zero for *parameterlength* and *parameteraddress*. The eight bytes that represent the parameter in the data buffer will contain zeros.

MBRPCNT=*symbol*
MBRPCNT=*(r1-r12)*
(Required) - Specifies a 4-byte field that contains the number of member parameters that are included in MBRPARM.

MBRSFUNC=*symbol*
MBRSFUNC=*(r1-r12)*
(Optional) - Specifies a 4-byte member subfunction code that is passed to the destination of the message in the SCI Input exit parameter list. This subfunction code, along with the MBRFUNC, identifies the message that is being sent.

If MBRSFUNC is a symbol, the symbol points to a 4-byte area of storage that contains the subfunction code.

NAME=*symbol*
NAME=*(r1-r12)*
(Optional) - Specifies the address of an 8-byte member name of the destination of this message. This name can be obtained from the Notify exit (when the member joins the IMSplex) or by issuing a CSLSCQRY message.

Note: One of the routing parameters (NAME, TOKEN, TYPE, NAMELIST, TOKENLIST or TYPELIST) must be included.

To route by NAME, the destination member must be authorized. If the member is not authorized, the message is not sent.

NAMELIST=*symbol*
NAMELIST=*(r1-r12)*
(Optional) - Specifies the address of a list of member names to which this message is to be routed. This list consists of a header and one or more list entries, each entry defining a single member name.

Note: One of the routing parameters (NAME, TOKEN, TYPE, NAMELIST, TOKENLIST or TYPELIST) must be included.

The list header DSECT is CSLSMGLH, and the list entry DSECT is CSLSNMLE. These DSECTs are defined in CSLSCMAP.

For a message to be routed to a member using NAMELIST, that member must be an authorized member. If a member name for a non-authorized member is included in NAMELIST, the name will not be found and the message will not be sent to that member.

The NAMELIST is sent to SCI for processing. Then, control is returned to your program. A response of “Request completed successfully” does not mean that the message was sent to all names in the list; it means that the list was successfully sent to SCI. Errors could occur while the list is processed and the message is sent. Possible errors include:

- Name not found
- Name found, but the member terminated before message is sent
- SCI abended

These errors are not returned to your program.

PARM=*symbol*
PARM=*(r1-r12)*
(Required) - Specifies the address of a parameter list used by the message to

pass the parameters to SCI. The length of the storage must be at least equal to the value of SMSG_LN. The storage must begin on a word boundary.

RETCODE=*symbol*
RETCODE=*(r1-r12)*
(Required) - Specifies the address of a 4-byte field to receive the CSLSCMSG return code. The SCI return codes are defined in CSLSRR. Possible return codes for CSLSCMSG are described in "CSLSCMSG Return and Reason Codes" on page 180.

RETNAME=*symbol*
RETNAME=*(r1-r12)*
(Optional) - Specifies the address of an 8-byte field to receive the name of the IMSplex member to which the message was sent. If the message is sent to more than one destination, nothing is returned in this field.

RETTOKEN=*symbol*
RETTOKEN=*(r1-r12)*
(Optional) - Specifies the address of an 8-byte field to receive the token of the IMSplex member to which the message was sent. If the message is sent to more than one destination, nothing is returned in this field.

ROUTE=*ANY*
ROUTE=*ALL*
ROUTE=*LOCAL*
(Optional) - Specifies how the message should be routed to the type specified in the TYPE parameter or the types specified in the TYPELIST parameter. This parameter is valid only if TYPE or TYPELIST is specified.

ANY
Routes the message to a single member of the types specified. SCI selects the member that will receive the message. TYPE=ANY is not valid with TYPELIST.

ALL
Routes the message to all members of the specified types.

LOCAL
Routes the message to all members of the specified types that are active on the local z/OS image.

RSNCODE=*symbol*
RSNCODE=*(r1-r12)*
(Required) - Specifies the address of a 4-byte field to receive the CSLSCMSG reason code. The SCI reason codes are defined in CSLSRR. Reason codes for CSLSCMSG are described in "CSLSCMSG Return and Reason Codes" on page 180.

SCITOKEN=*symbol*
SCITOKEN=*(r1-r12)*
(Required) - Specifies the address of a 16-byte field that contains the SCI token of the member making the request. The token was returned on the CSLSCREG request.

TOKEN=*symbol*
TOKEN=*(r1-r12)*
(Optional) - Specifies the address of the 16-byte SCI token of the destination of this message. This token can be obtained from the Notify exit (when the member joins the IMSplex) or by issuing a CSLSCQRY message.

Note: One of the routing parameters (NAME, TOKEN, TYPE, NAMELIST, TOKENLIST or TYPELIST) must be included.

TOKENLIST=*symbol*
TOKENLIST=*(r1-r12)*

(Optional) - Specifies the address of a list of SCI tokens that represent members to which this message is to be routed. This list consists of a header and one or more list entries, each entry defining a single SCI token.

Note: One of the routing parameters (NAME, TOKEN, TYPE, NAMELIST, TOKENLIST or TYPELIST) must be included.

The list header DSECT is CSLSMGLH, and the list entry DSECT is CSLSTKLE. These DSECTs are defined in CSLSCMAP.

The TOKENLIST is sent to SCI for processing. Then, control is returned to your program. A response of "Request completed successfully" does not mean that the message was sent to all SCI tokens in the list; it means that the list was successfully sent to SCI. Errors could occur while the list is processed and the message is sent. Possible errors include:

- Token not found
- Token found but member terminated before message is sent
- SCI abended

These errors are not returned to your program.

TYPE=*symbol*
TYPE=*'AOP'*
TYPE=*'BATCH'*
TYPE=*'CQS'*
TYPE=*'DBRC'*
TYPE=*'IMS'*
TYPE=*'IMSCON'*
TYPE=*'OM'*
TYPE=*'OTHER'*
TYPE=*'RM'*
TYPE=*'SCI'*

(Optional) - TYPE specifies the SCI type of the destination of this message. SCI routes the message to one or more members of the specified type (depending on the value of the route parameters). If there are no members of the specified type, an error is returned.

Note: One of the routing parameters (NAME, TOKEN, TYPE, NAMELIST, TOKENLIST or TYPELIST) must be included.

If this parameter is passed as a literal, the literal must be enclosed in single quotes. If this parameter is passed as a symbol or register, the symbol or register must contain the member type code. The member type code can be obtained by using the CSLSTPIX macro.

For a description of the IMSplex member types, see "CSLSCREG: Registration Request" on page 187.

TYPELIST=*symbol*
TYPELIST=*(r1-r12)*

(Optional) - Specifies the address of a list of member types to which this message is to be routed. This list consists of a header and one or more list entries, each entry defining a single SCI token.

Note: One of the routing parameters (NAME, TOKEN, TYPE, NAMELIST, TOKENLIST or TYPELIST) must be included.

The list header DSECT is CSLSMGLH, and the list entry DSECT is CSLSTPLE. These DSECTs are defined in CSLSCMAP.

The TYPELIST is sent to SCI for processing. Then, control is returned to your program. A response of "Request completed successfully" does not mean that the message was sent to all types in the list; it means that the list was successfully sent to SCI. Errors could occur while the list is processed and the message is sent. Possible errors include:

- No members of the specified type are active
- A member of the specified type was found but terminated before the message is sent
- SCI abended

These errors are not returned to your program.

CSLSCMSG Return and Reason Codes

Table 70 lists the return and reason codes that can be returned on a CSLSCMSG macro request. Also included is the meaning of a reason code (that is, what possibly caused it).

Table 70. CSLSCMSG Return and Reason Codes

Return Code	Reason Code	Meaning
X'00000000'	X'00000000'	The request completed successfully.
X'01000008'	X'00002004'	An invalid function was passed to the SCI interface PC routine.
	X'00002008'	The number of parameters passed was either less than or equal to zero, or greater than the maximum allowed.
	X'00002010'	An invalid type was passed.
	X'00002018'	The SCI token was invalid.
	X'00002024'	The PHDR length was invalid.
	X'00002028'	The routing data length was invalid.
	X'00002034'	The length of the parameters is too large for a non-authorized caller.
	X'00002038'	The parameter list version is invalid.
X'01000010'	X'00004000'	SCI is not active.
	X'0000400C'	The destination IMSplex member is not active. The requested member might have been specified by name, token, or type.
	X'0000401C'	The calling member is in the process of deregistering from SCI.
	X'00004FFF'	The function is not supported.

Table 70. CSLSCMSG Return and Reason Codes (continued)

Return Code	Reason Code	Meaning
X'01000014'	X'00005000'	An SCI internal error occurred.
	X'00005004'	An ESTAE add error occurred.
	X'00005024'	An error in the SRB routine occurred.
	X'00005028'	The routing type was invalid.
	X'0000502C'	The member could not be found due to an internal BPE hash table services error.
	X'00005030'	An SCI buffer could not be obtained.
	X'00005034'	A key 7 buffer in the SCI address space could not be obtained for a copy of PHDR and parameters.
	X'00005038'	An IEAMSCHD error occurred; the SRB could not be scheduled to the target address space.
	X'0000504C'	The message SRB key 7 parameter area could not be obtained.
	X'00005500'	An abend occurred during CSLSCMSG processing.
	X'00005504'	An abend occurred when the member parameters were copied to the target address space.

CSLSCQRY: Query Request

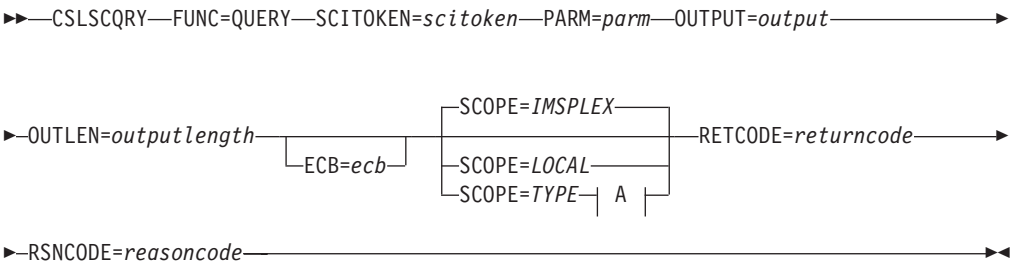
The SCI Query request allows an IMSplex member to obtain information about the members of the IMSplex.

CSLSCQRY Syntax

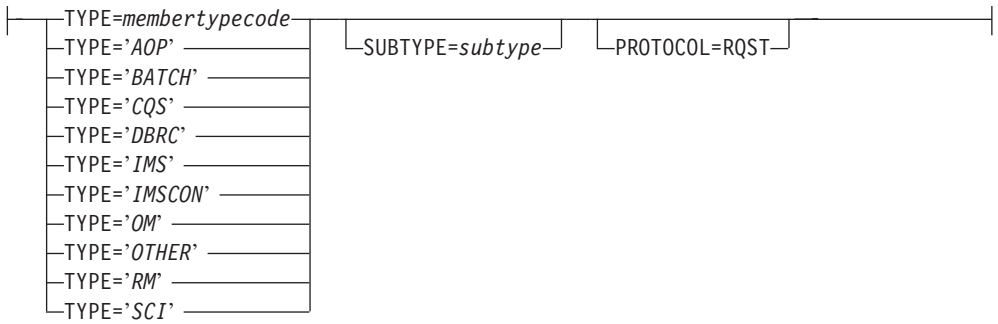
The syntax for the CSLSCQRY request follows.

CSLSCQRY DSECT Syntax: Use the DSECT function of a CSLSCQRY request to include equate (EQU) statements in your program for the CSLSCQRY parameter list length, the IMSplex types and the CSLSCQRY return and reason codes.

CSLSCQRY QUERY Syntax: Use the following syntax to issue the CSLSCQRY service request. The output is returned to the caller when the request is complete.



A:



CSLSCQRY Parameters

The parameters for the CSLSCQRY request follow.

ECB=*symbol*
ECB=*(r1-r12)*

(Optional) - Specifies the address of a z/OS ECB used for asynchronous requests. When the request is complete, the ECB specified is posted. If an ECB is not specified, the task is suspended until the request is complete. If an ECB is specified, the invoker of the macro must issue a WAIT (or equivalent) after receiving control from CSLSCQRY, before using or examining any data returned by this macro (including the RETCODE and RSNCODE fields).

OUTLEN=*symbol*
OUTLEN=*(r1-r12)*

(Required) - Specifies a 4-byte field to receive the length of the output returned by the CSLSCQRY request. OUTLEN receives the length of the output pointed to by the OUTPUT= parameter.

The output length is zero if no output is built, for example, if an error is detected before any output can be built.

OUTPUT=*symbol*
OUTPUT=*(r1-r12)*

(Required) - Specifies a field to receive a pointer to the variable length output returned by the CSLSCQRY request. The output length is returned in the OUTLEN= field.

The output address is zero if no output was built, for example, if an error was detected before any output could be built.

The CSLSQRYO macro maps the output that is returned. The output contains a header and one or more list entries.

The output buffer is not preallocated by the caller. After being returned by the request, this word contains the address of a buffer containing the query output. It is the caller's responsibility to release this storage by issuing the CSLSCBFR FUNC=RELEASE request when it is through with the storage.

PARM=*symbol*
PARM=*(r1-r12)*
(Required) - Specifies the CSLSCQRY parameter list. The length of the parameter list must be equal to the parameter list length EQU value defined by SQRY_PARMLN.

PROTOCOL=RQST
(Optional) - SCI protocol for sending the request to SCI. RQST indicates that the SCI request interface protocol is to be used for the request.

RETCODE=*symbol*
RETCODE=*(r1-r12)*
(Required) - Specifies the address of a 4-byte field to receive the CSLSCQRY return code. SCI return codes are defined in CSLSRR. Possible return codes for CSLSCQRY are described in "CSLSCQRY Return and Reason Codes" on page 184.

RSNCODE=*symbol*
RSNCODE=*(r1-r12)*
(Required) - Specifies a 4-byte field to receive the reason code on output. SCI reason codes are defined in CSLSRR. Possible reason codes for CSLSCQRY are described in "CSLSCQRY Return and Reason Codes" on page 184.

SCITOKEN=*symbol*
SCITOKEN=*(r1-r12)*
(Required) - Specifies a 16-byte field containing the SCI token. This token uniquely identifies this connection to SCI. The SCI token was returned by a successful `CSLSCREG FUNC=REGISTER` request.

SCOPE=*IMSPLEX*
SCOPE=*LOCAL*
SCOPE=*TYPE*
(Optional) - Specifies the scope of information that is being requested.

IMSPLEX
This option returns data for all of the members in the IMSplex.

LOCAL
This option returns information for all of the members on the local z/OS image.

TYPE
This option returns information for all of the members that are of the specified IMSplex member type (and optionally subtype).

SUBTYPE=*symbol*
SUBTYPE=*(r1-r12)*
(Optional) - Four-byte input parameter that specifies the address of an 8-byte subtype that further qualifies the IMSplex member type about which information is being requested. This subtype is defined by the IMSplex member and was specified on the CSLSCREG request.

This parameter is valid only when SCOPE=TYPE.

TYPE=*symbol*
TYPE=*'AOP'*
TYPE=*'BATCH'*

TYPE=*'CQS'*
TYPE=*'DBRC'*
TYPE=*'IMS'*
TYPE=*'IMSCON'*
TYPE=*'OM'*
TYPE=*'OTHER'*
TYPE=*'RM'*
TYPE=*'SCI'*

(Optional) - Specifies the IMSplex member type for which the query is being issued. SCI will return information for all of the members that are of the specified IMSplex member type (and, optionally, subtype). This parameter is required when SCOPE=TYPE.

If this parameter is passed as a literal, the literal must be enclosed in single quotes. If it is passed as a symbol, the symbol points to a word in storage that contains the code for the member type. If it is passed as a register, the register contains the member type code in the low-order half word of the register.

The code for the member type can be obtained by using the CSLSTPIX macro. For information on member types, refer to "CSLSCREG Parameters" on page 188.

CSLSCQRY Return and Reason Codes

Table 71 lists the return and reason codes that can be returned on a CSLSCQRY macro request. Also included is the meaning of a reason code (that is, what possibly caused it). In addition, CSLSCQRY can return any of the return codes listed in Table 76 on page 201.

Table 71. CSLSCQRY Return and Reason Codes

Return Code	Reason Code	Meaning
X'00000000'	X'00000000'	Request completed successfully.
X'01000008'	X'00002050'	The caller of the service attempted to pass an invalid parameter list. The request is rejected.
X'0100000C'	X'00003004'	No member data was returned for the request.
X'01000014'	X'00005048'	SCI was unable to obtain storage for the output area of the request.

CSLSCQSC: Quiesce Request

The SCI Quiesce request tells SCI to stop routing messages and requests that have been routed by TYPE to the issuing IMSplex member. After this request has successfully completed, the only messages and requests that are routed to the member are those that are routed directly by SCITOKEN or by NAME.

Note: Because of the asynchronous nature of the processes within the IMSplex and z/OS, messages and requests routed by TYPE might still be received by the IMSplex member after successful completion of the CSLSCQSC FUNC=QUIESCE request. The potential for this occurring is small, but it can happen. The IMSplex member must be able to handle a message or request coming in after the CSLSCQSC FUNC=QUIESCE has successfully completed.

CSLSCQSC Syntax

The syntax for the CSLSCQSC request follows.

CSLSCQSC DSECT Syntax: Use the DSECT function of a CSLSCQSC request to include equate (EQU) statements in your program for the CSLSCQSC parameter list length and the CSLSCQSC return and reason codes.

```
►►—CSLSCQSC—FUNC=DSECT———————————————————————————————————►◄
```

CSLSCQSC QUIESCE Syntax: The CSLSCQSC FUNC=QUIESCE request quiesces the connection between SCI and the IMSplex member. After the successful completion of the request, only messages and requests that are routed directly by SCITOKEN or by NAME are sent to this IMSplex member.

```
►►—CSLSCQSC—FUNC=QUIESCE—PARM=parm—SCITOKEN=scitoken—RETCODE=returncode———►

►—RSNCODE=reasoncode———————————————————————————————————————►◄
```

CSLSCQSC Parameters

The parameters for the CSLSCQSC request follow.

PARM=*symbol*
PARM=*(r1-r12)*
(Required) - Specifies the CSLSCQSC parameter list. The length of the parameter list must be equal to the parameter list length EQU value defined by SQSC_PARMLN.

RETCODE=*symbol*
RETCODE=*(r1-r12)*
(Required) - Specifies a 4-byte field to receive the return code on output. SCI return codes are defined in CSLSRR. Possible return codes for CSLSCQSC are described in Table 72.

RSNCODE=*symbol*
RSNCODE=*(r1-r12)*
(Required) - Specifies a 4-byte field to receive the reason code on output. SCI reason codes are defined in CSLSRR. Possible reason codes for CSLSCQSC are described in Table 72.

SCITOKEN=*symbol*
SCITOKEN=*(r1-r12)*
(Required) - Specifies a 16-byte field containing the SCI token. This token uniquely identifies this connection to SCI. The SCI token was returned by a successful CSLSCREG FUNC=REGISTER request.

CSLSCQSC Return and Reason Codes

Table 72 lists the return and reason codes that can be returned on a CSLSCQSC macro request. Also included is the meaning of a reason code (that is, what possibly caused it). In addition, CSLSCQSC can return any of the return codes in Table 70 on page 180.

Table 72. CSLSCQSC Return and Reason Codes

Return Code	Reason Code	Meaning
X'00000000'	X'00000000'	The request completed successfully.
X'01000008'	X'01002038'	The parameter list version is invalid.
X'01000010'	X'00004000'	SCI is not active.
	X'00004FFF'	The function is not supported.

Table 72. CSLSCQSC Return and Reason Codes (continued)

Return Code	Reason Code	Meaning
X'01000014'	X'00005000'	An SCI internal error occurred.

CSLSCRDY: Ready Request

The SCI ready request enables the IMSplex member to receive messages and requests that are routed by TYPE. After the CSLSCREG request is issued and until CSLSCRDY is issued, the IMSplex member can only receive requests that are routed directly to a single target address space. The IMSplex member can send messages and requests that are routed by any method.

Note: The IMSplex member must be ready to process messages and requests that have been routed by TYPE when CSLSCRDY is issued. Because of the asynchronous nature of an IMSplex, the member might receive a message or request that has been routed by TYPE before control is returned after issuing CSLSCRDY.

CSLSCRDY Syntax

The syntax examples for the CSLSCRDY request follow.

DSECT Syntax: Use the DSECT function of a CSLSCRDY request to include equate (EQU) statements in your program for the CSLSCRDY parameter list length and the CSLSCRDY return and reason codes.

```
►►—CSLSCRDY—FUNC=DSECT———————————————————————————————————————►◄
```

READY Syntax: The CSLSCRDY FUNC=READY request tells SCI that the IMSplex member is now ready to receive messages and requests that are routed by an IMSplex member type.

```
►►—CSLSCRDY—FUNC=READY—SCITOKEN=scitoken—PARM=parm—RETCODE=returncode————►

►—RSNCODE=reasoncode———————————————————————————————————————————►◄
```

CSLSCRDY Parameters

The parameters for the CSLSCRDY request follow.

PARM=*symbol*
PARM=*(r1-r12)*
(Required) - Specifies the CSLSCRDY parameter list. The length of the parameter list must be equal to the parameter list length EQU value defined by SRDY_PARMLN.

RETCODE=*symbol*
RETCODE=*(r1-r12)*
(Required) - Specifies a 4-byte field to receive the return code on output. SCI return codes are defined in CSLSRR. Possible reason codes for CSLSCRDY are described in “CSLSCRDY Return and Reason Codes” on page 187.

RSNCODE=*symbol*
RSNCODE=*(r1-r12)*
(Required) - Specifies the address of a 4-byte field to receive the CSLSCRDY

reason code. SCI reason codes are defined in CSLSRR. Possible reason codes for CSLSCRDY are described in "CSLSCRDY Return and Reason Codes."

SCITOKEN=*symbol*
SCITOKEN=*(r1-r12)*
(Required) - Specifies a 16-byte field containing the SCI token. This token uniquely identifies this connection to SCI. The SCI token was returned by a successful CSLSCREG FUNC=REGISTER request.

CSLSCRDY Return and Reason Codes

Table 73 lists the return and reason codes that can be returned on a CSLSCRDY macro request. Also included is the meaning of a reason code (that is, what possibly caused it). In addition, CSLSCRDY can return any of the return codes in Table 70 on page 180.

Table 73. CSLSCRDY Return and Reason Codes

Return Code	Reason Code	Meaning
X'00000000'	X'00000000'	The request completed successfully.
	X'00002038'	The parameter list version is invalid.
X'01000010'	X'00004000'	SCI is not active.
	X'00004FFF'	The function is not supported.
X'01000014'	X'00005000'	An SCI internal error occurred.

CSLSCREG: Registration Request

The SCI registration request is used to create a connection between an IMSplex member and SCI. Before SCI can be used for communication within the IMSplex, an IMSplex member must issue the CSLSCREG request and receive an SCI token when the request has successfully completed. This token is used with all subsequent SCI requests. If SCI terminates while the IMSplex member is active, the member is still registered when SCI becomes active again. The SCI token that the member received on the initial CSLSCREG request is still valid.

Restriction: CSLSCREG is not supported when the caller's address space has been marked non-swappable by a `SYSEVENT DONTSWAP` call. Issuing a CSLSCREG in this environment produces unpredictable results. A caller that issued a `SYSEVENT DONTSWAP` must issue a `SYSEVENT OKSWAP` before registering with SCI.

CSLSCREG Syntax

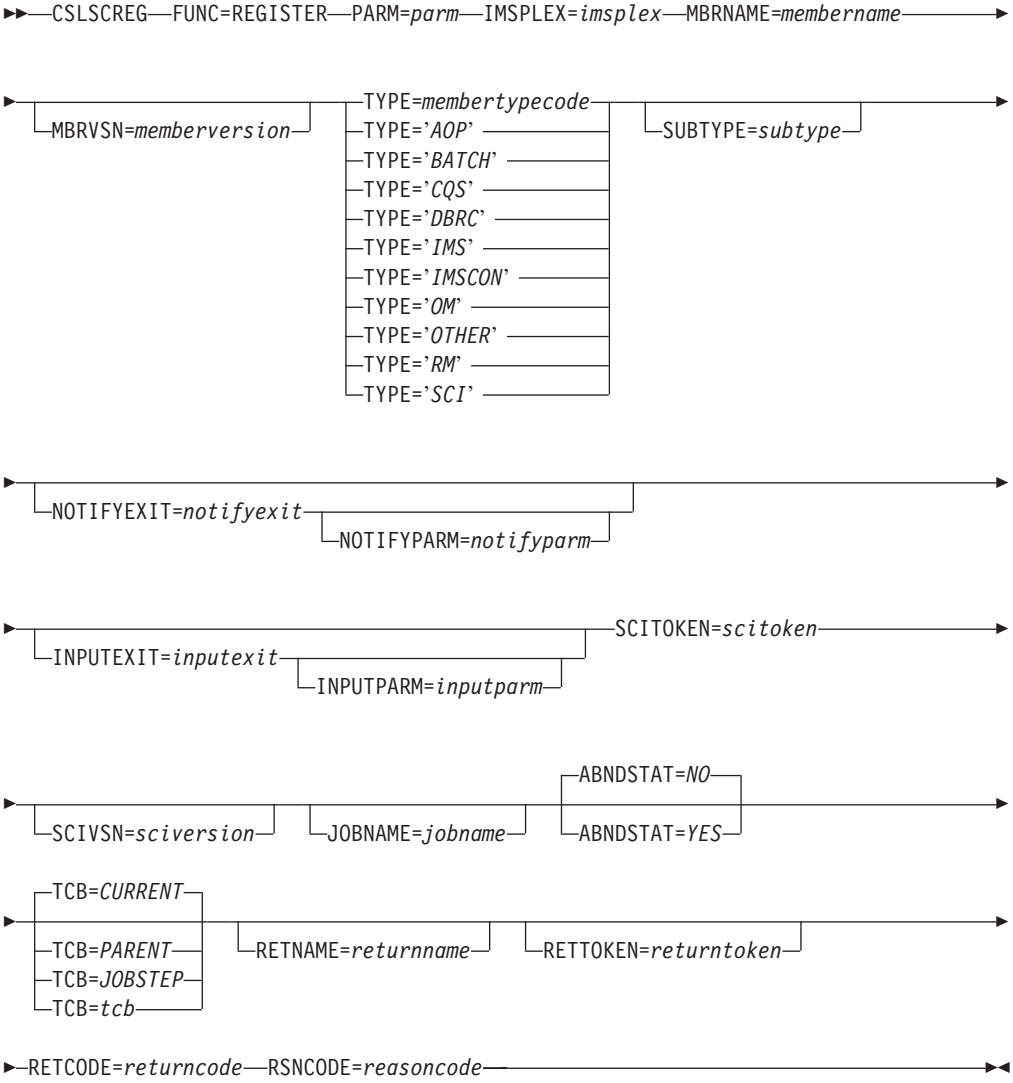
The syntax for the CSLSCREG request follows.

CSLSCREG DSECT Syntax: Use the DSECT function of a CSLSCREG request to include equate (EQU) statements in your program for the CSLSCREG parameter list length, the IMSplex types and the CSLSCREG return and reason codes.

```
►►—CSLSCREG—FUNC=DSECT————————————————————————————————►◄
```

CSLSCREG REGISTER Syntax: The CSLSCREG FUNC=REGISTER request establishes a connection between an IMSplex member and SCI. An SCI token is returned on successful completion of this request. This token must be used on all subsequent SCI requests. Until the CSLSCRDY FUNC=READY request is issued, the IMSplex member receives only messages and requests that are routed directly to it (by SCITOKEN or by NAME). Messages and requests that are routed by TYPE

are not routed to this member. Messages and requests routed by any method can be sent by this member when the CSLSCREG FUNC=READY request has been successfully completed. The syntax for the REGISTER function of the CSLSCREG request follows.



CSLSCREG Parameters

The parameters for the CSLSCREG request follow.

ABNDSTAT=*NO*
ABNDSTAT=*YES*

(Optional) - Indicates if SCI is to keep track of the member if the member abnormally terminates. If ABNDSTAT=YES is specified, SCI will retain an entry for the member with a status of ABTERM. If the member normally terminates or if the member abnormally terminates after a successful CSLSCDRG, SCI does not keep a record of the member.

This parameter is ignored for non-authorized IMSplex members.

IMSPLEX=*symbol*

IMSPLEX=*(r2-r12)*
(Required) - Specifies the address of a 1- to 5-character IMSplex name. The IMSPlex name identifies the SCI to which this request is directed. If specified as a symbol, the symbol references storage that contains the IMSplex name.

INPUTEXIT=*symbol*
INPUTEXIT=*(r2-r12)*
(Optional) - Specifies the address of the SCI input exit routine. The input exit is called each time there is a message or request for the member.

INPUTPARM=*symbol*
INPUTPARM=*(r2-r12)*
(Optional) - Specifies the address of an 8-byte area that contains member data. This data is passed to the input exit routine each time it is called. If specified as a symbol, the symbol references storage that contains the member data.

JOBNAME=*symbol*
JOBNAME=*(r2-r12)*
(Optional) - Specifies the address of an 8-byte area to receive the SCI jobname.

MBRNAME=*symbol*
MBRNAME=*(r2-r12)*
(Required) - Specifies the address of an 8-byte name of the IMSplex member registering with SCI. For an authorized member, this name must be unique within the IMSplex. For a non-authorized member, this name does not need to be unique. If it is specified as a symbol, the symbol refers to storage that contains the IMSplex member name. Valid characters for the name are A-Z, 0-9, and special characters @, #, and $.

MBRVSN=*symbol*
MBRVSN=*(r2-r12)*
(Optional) - Specifies the address of a 4-byte version of the IMSplex member registering with SCI. This version number is passed in the parameter list of the SCI Notify exit when this IMSplex member is the subject of the event. It is also passed in the parameter list of the SCI Input exit for messages and requests that originate from this member. If MBRVSN is not specified, the version number in the exit parameter list is set to zeros. If it is specified as a symbol, the symbol references storage that contains the IMSplex member version.

SCI does not validate this field; however, the field can be output on the QRY IMSPLEX command. It is assumed to have the following format: X'00vvrrmm'.
- 00 - this byte is ignored
- vv - version number
- rr - release number
- mm - modification level or subrelease number

For example, X'00080100' would be output as 8.1.0.

NOTIFYEXIT=*symbol*
NOTIFYEXIT=*(r2-r12)*
(Optional) - Specifies the address of the SCI Notify exit routine. The Notify exit is driven whenever there is a change of status of an IMSplex member. See “CSL SCI Notify Client Exit Routine” on page 166 for a list of events that result in this exit being driven.

NOTIFYPARM=*symbol*
NOTIFYPARM=*(r2-r12)*
(Optional) - Specifies the address of an 8-byte area that contains member data.

This data is passed to the Notify exit routine each time it is called. If it is specified as a symbol, the symbol references storage that contains the member data.

PARM=*symbol*
PARM=*(r2-r12)*
(Required) - Specifies the address of a parameter list used by the request to pass the parameters to SCI. The length of the storage must be at least equal to the value of SREG_LN.

RETCODE=*symbol*
RETCODE=*(r2-r12)*
(Required) - Specifies the address of a 4-byte field to receive the CSLSCREG return code. SCI return codes are defined in CSLSRR. Possible return codes for CSLSCREG are described in “CSLSCREG Return and Reason Codes” on page 192.

RETNAME=*symbol*
RETNAME=*(r2-r12)*
(Optional) - Specifies the address of an 8-byte field to receive the name of the SCI that processes the registration request.

RETTOKEN=*symbol*
RETTOKEN=*(r2-r12)*
(Optional) - Specifies the address of a 16-byte field to receive the SCI token of the SCI that processes the registration request.

RSNCODE=*symbol*
RSNCODE=*(r2-r12)*
(Required) - Specifies the address of a 4-byte field to receive the CSLSCREG reason code. SCI reason codes are defined in CSLSRR. Possible reason codes for CSLSCREG are described in “CSLSCREG Return and Reason Codes” on page 192.

SCITOKEN=*symbol*
SCITOKEN=*(r2-r12)*
(Required) - Specifies a 16-byte field containing the SCI token. This token uniquely identifies this connection to SCI. The SCI token was returned by a successful CSLSCREG FUNC=REGISTER request.

SCIVSN=*symbol*
SCIVSN=*(r2-r12)*
(Optional) - Specifies the address of a 4-byte field to receive the SCI version number. The version number has the following format: `00vvrrmm`.

`00` This byte is reserved for future use. Currently, it is always 00.

`vv` Version number.

`rr` Release number.

`mm` Modification level or subrelease number.

Example: SCI version 1.1.0 is shown as X'00010100'.

SUBTYPE=*symbol*
SUBTYPE=*(r2-r12)*
(Optional) - Specifies the address of the 8-byte subtype of the member registering with SCI. The subtype is defined by the user and can be any eight characters. If it is specified as a symbol, the symbol references storage that contains the subtype. If not specified, this parameter is set to blanks. If SUBTYPE is not specified, it will be set to blanks.

The subtype can contain alphanumeric characters (A-Z, 0-9), or printable characters (not case sensitive), with the exception of the characters &, <, and >. OM converts any invalid data placed in this field to periods (.) before sending the XML output to the client.

TCB=*CURRENT*
TCB=*JOBSTEP*
TCB=*PARENT*
TCB=*symbol*
TCB=*(r2-r12)*

(Optional) - Specifies the TCB with which the new SCI connection is associated. The SCI connection persists until one of the following occurs:

- The member deregisters by using CSLSCDRG.
- The TCB associated with the connection terminates.

All callers of CSLSCREG can specify the following values for the TCB parameter:

CURRENT
Associates the SCI connection with the currently executing TCB. This is the default.

JOBSTEP
Associates the SCI connection with the address space jobstep TCB (the TCB pointed to by ASCBXTCB).

PARENT
Associates the SCI connection with the TCB that attached the currently-executing TCB.

For non-authorized callers, the indicated TCB must have the same storage key associated with it as the caller's current PSW key (that is, TCBPKF must match the current PSW key).

Authorized callers can, in addition, identify an explicit TCB by specifying a symbol or register. If specified as a symbol, the symbol must be the label on a word of storage containing the address of the TCB. If specified as a register, the register must contain the TCB address.

TYPE=*membertypecode*
TYPE=*'AOP'*
TYPE=*'BATCH'*
TYPE=*'CQS'*
TYPE=*'DBRC'*
TYPE=*'IMS'*
TYPE=*'IMSCON'*
TYPE=*'OM'*
TYPE=*'OTHER'*
TYPE=*'RM'*
TYPE=*'SCI'*

(Required) - Specifies the SCI member type of the address space that is registering with SCI.

If this parameter is passed as a literal, the literal must be enclosed in single quotes. If this parameter is passed as a symbol or register, the symbol or register must contain the member type code.

The code for the member type can be obtained by using the CSLSTPIX macro. Member types include:

AOP This SCI type is an automated operator program. It interacts with OM by sending commands and receiving responses to the commands.

Batch This SCI type is an IMS batch region. It interacts as an IMS DL/I batch or utility region.

CQS This SCI type is an IMS Common Queue Server. It provides access to a set of common queues within the IMSplex.

DBRC
This SCI type is an IMS Database Recovery Control Region.

IMS This SCI type is an IMS region. It can include the database manager, transaction manager, and FDBR (an IMS control region that recovers database resources when an IMS database manager fails). SUBTYPE is used to further qualify a particular control region (for example, DBDC, DBCTL, DCCTL, or FDBR).

IMSCON
This SCI type is a connector to IMS. It acts as an interface between IMS and protocols that are not supported by IMS directly (such as TCP/IP).

OM This SCI type is an IMS Operations Manager, which is part of the CSL. It receives commands from AOPs, routes the commands to other members of the IMSplex that have registered for the command, consolidates the responses to the command, and sends the output back to the originating AOP.

Other This SCI type is any other address space that does not fall into one of the defined SCI types.

RM This SCI type is an IMS Resource Manager, which is part of the CSL. It manages resources within the IMSplex and coordinates IMSplex-wide processes. SUBTYPE is used to further qualify whether there is a single RM in the IMSplex (SNGLRM) or there are multiple RMs in the IMSplex (MULTRM).

SCI This SCI type is an IMS SCI, which is part of the CSL. It manages communications within the IMSplex.

CSLSCREG Return and Reason Codes

Table 74 lists the return and reason codes that can be returned on a CSLSCREG macro request. Also included is the meaning of a reason code (that is, what possibly caused it).

Table 74. CSLSCREG Return and Reason Codes

Return Code	Reason Code	Meaning
X'00000000'	X'00000000'	The request completed successfully.
X'01000004'	X'00001000'	The member is already registered with SCI. The member's current SCITOKEN is returned.
X'01000008'	X'00002010'	An invalid type was passed
	X'00002038'	The parameter list version is invalid.

Table 74. CSLSCREG Return and Reason Codes (continued)

Return Code	Reason Code	Meaning
X'01000010'	X'01004000'	SCI is not active.
	X'01004004'	CSLSRG00 could not be loaded.
	X'00004008'	The user ID of the member address space is not authorized to register with this SCI.
	X'00004010'	The member name, *membername*, is not unique for the authorized client. The registration is rejected.
	X'00004028'	A non-authorized member tried to register as an authorized system SCI type.
	X'00004FFF'	The function is not supported.
X'01000014'	X'00005000'	An SCI internal error occurred.
	X'00005004'	An ESTAE add error occurred.
	X'00005008'	A BPE SVC error occurred.
	X'0000500C'	A z/OS Name/Token retrieve error occurred.
	X'00005010'	An error occurred while establishing ResMgr.
	X'00005014'	An error occurred while obtaining storage.
	X'00005018'	An error occurred while obtaining a TTOKEN.
	X'0000501C'	An ALESERV error occurred.
	X'00005020'	An ENQ resource error occurred.
	X'00005050'	A BPECGBET error occurred in CSLSRGS0.
	X'00005054'	An ALESERV error occurred in CSLSRGS0.
	X'00005058'	A BPEHTADD error occurred in CSLSRGS0.
	X'00005064'	A BPEHTFND token error occurred in CSLSRGS0.
	X'00005070'	The SCI buffer manager could not be initialized.
	X'00005080'	The XCF join for the member failed.
	X'00005084'	A non-authorized member specified an explicit connection TCB.
	X'00005088'	The connection TCB key does not match the CSLSCREG caller's key.
	X'0000508C'	The TCB type code passed on the CSLSCREG request is invalid.
	X'00005090'	Error enqueueing registration AWE. This is an internal error.
	X'00005094'	Error scheduling SRB to SCI. This is an internal error.
	X'00005500'	An abend occurred during CSLSCREG processing.

CSLSCRQR Request Return Request

CSLSCRQR returns a request to the IMSplex member from which the request originated. It should be issued when the server has completed the request and is ready to return the output from the request. It copies the output back to the requestor's address space.

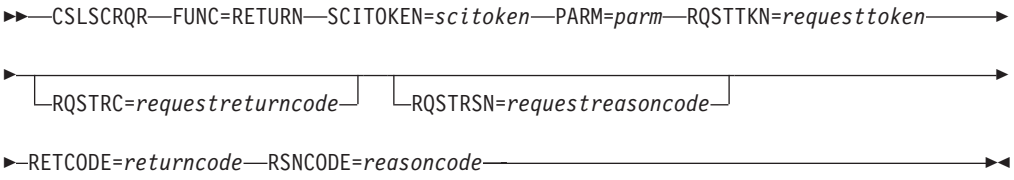
Only request servers can issue CSLSCRQR because an IMSplex member cannot issue the macro without first receiving a request. A request server must be authorized and running key 7.

CSLSCRQR Syntax

The syntax for the CSLSCRQR request follows.

CSLSCRQR DSECT Syntax: Use the DSECT function of a CSLSCRQR request to include equate (EQU) statements in your program for the CSLSCRQR parameter list length and the CSLSCRQR return and reason codes.

CSLSCRQR RETURN Syntax: The syntax for the CSLSCRQR FUNC=RETURN request follows.



CSLSCRQR Parameters

The parameters for the CSLSCRQR request follow.

PARM=*symbol*
PARM=*(r1-r12)*
(Required) - Specifies the CSLSCRQR parameter list. The length of the parameter list must be equal to the parameter list length EQU value defined by SRQR_PARMLN.

RQSTRC=*symbol*
RQSTRC=*(r1-r12)*
(Optional) - Specifies the return code that is associated with the request being returned. This return code will be given to the requesting member in the storage pointed to by the RETCODE parameter of the CSLSCRQS that originated this request. If this parameter is not specified, a return code of zero will be given to the requesting member.

If specified as a symbol, the symbol references storage that contains the return code.

RQSTRSN=*symbol*
RQSTRSN=*(r1-r12)*
(Optional) - Specifies the reason code that is associated with the request being returned. This reason code will be given to the requesting member in the storage pointed to by the RSNCODE parameter of the CSLSCRQS that originated this request. If this parameter is not specified, a reason code of zero will be given to the requesting member.

If specified as a symbol, the symbol references storage that contains the return code.

RQSTTKN=*symbol*
RQSTTKN=*(r1-r12)*
(Required) - Specifies the request token that is associated with the request being returned. This request token can be obtained from the input exit parameter list (INXP_RQSTTKN) when the request was presented to the request processing member.

If specified as a symbol, the symbol references storage that contains the return code.

RETCODE=*symbol*
RETCODE=*(r1-r12)*
(Required) - Specifies the address of a 4-byte field to receive the CSLSCRQR return code. SCI return codes are defined in CSLSRR. Possible return codes for CSLSCRQR are described in "CSLSCRQR Return and Reason Codes."

RSNCODE=*symbol*
RSNCODE=*(r1-r12)*
(Required) - Specifies the address of a 4-byte field to receive the CSLSCRQR reason code. SCI reason codes are defined in CSLSRR. Possible reason codes for CSLSCRQR are described in "CSLSCRQR Return and Reason Codes."

SCITOKEN=*symbol*
SCITOKEN=*(r1-r12)*
(Required) - Specifies a 16-byte field containing the SCI token. This token uniquely identifies this connection to SCI. The SCI token was returned by a successful CSLSCREG FUNC=REGISTER request.

CSLSCRQR Return and Reason Codes

Table 75 lists the return and reason codes that can be returned on a CSLSCRQR macro request. Also included is the meaning of a reason code (that is, what possibly caused it).

Table 75. CSLSCRQR Return and Reason Codes

Return Code	Reason Code	Meaning
X'00000000'	X'00000000'	The request completed successfully.
X'01000008'	X'00002004'	The function passed to the SCI interface PC routine was invalid.
	X'00002018'	The SCI token is invalid.
	X'00002038'	The parameter list version is invalid.
X'01000010'	X'00004000'	SCI is not active.
	X'0000400C'	The target member is not active.
	X'00004FFF'	The function is not supported.
X'01000014'	X'00005000'	An SCI internal error occurred.
	X'0000502C'	The member could not be found due to an internal BPE hash table services error.
	X'00005030'	An SCI buffer could not be obtained.
	X'00005034'	A key 7 buffer in the SCI address space could not be obtained for a copy of PHDR and parameters.
	X'00005038'	An IEAMSCHD error occurred; SRB could not be scheduled to the target address space.
	X'00005040'	The request is not outstanding and cannot be returned.
	X'00005044'	An SCI-allocated output buffer could not be obtained.
	X'00005500'	An abend occurred during the processing of an SCI request.
	X'00005504'	An abend occurred when the member parameters were copied to the target address space.

CSLSCRQS: Send Request Request

CSLSCRQS allows an IMSplex member to send a request to another member in the IMSplex. The target member can be specified by SCITOKEN, member name, or member type.

A request in an IMSplex can contain both input and output data (from the target member's perspective). This contrasts to a message that can only contain input data (again, from the target member's perspective). The data of a request is copied to the target member's address space. The function is processed, and the output is returned to the requestor's address space. If the request included an ECB, control is returned to the requesting module after the request has been processed by SCI. The requestor must then wait on the ECB.

The ECB is posted when the request processing has completed. The requestor then looks at the RETCODE and RSNCODE fields to determine the outcome of the request. If no ECB is included in the request, the RETCODE and RSNCODE fields can be used to determine the outcome of the request when the requesting module gets control back from SCI.

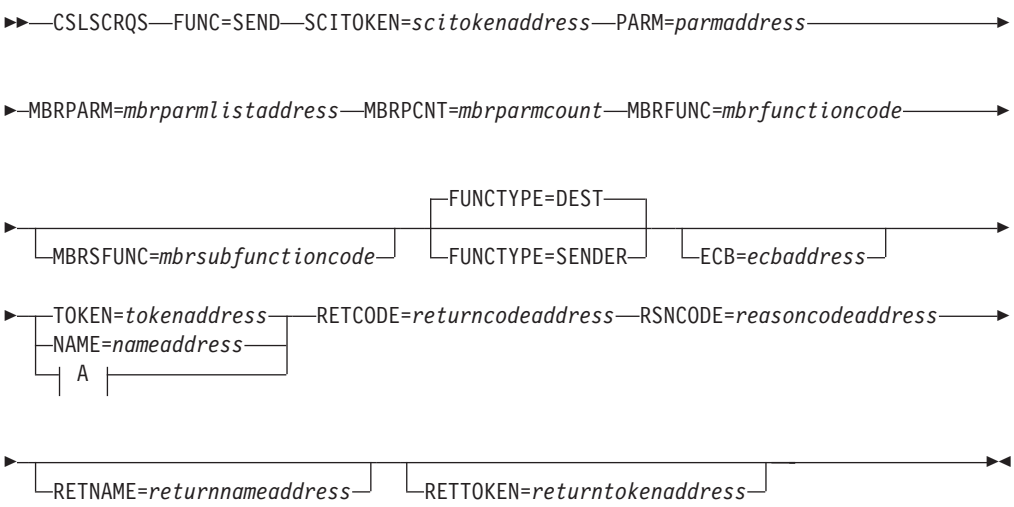
Note: Before issuing CSLSCRQS, the requester should clear the fields that will receive the address and length of the SCI Allocated Output parameters. If the request is not sent to the destination because of an error, or if there is no data to output, SCI will not update the length and address fields.

CSLSCRQS Syntax

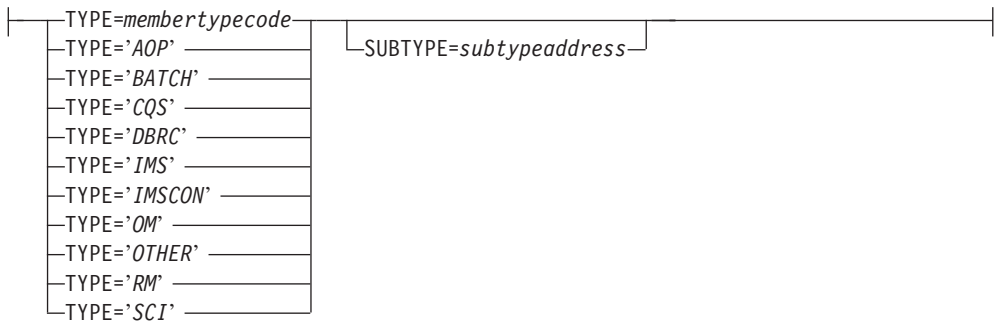
The syntax for the CSLSCRQS request follows.

DSECT Syntax: Use the DSECT function of a CSLSCRQS request to include equate (EQU) statements in your program for the CSLSCRQS parameter list length, the IMSplex types and the CSLSCRQS return and reason codes.

SEND REQUEST Syntax: The syntax for the `CSLSCRQS FUNC=SEND` request follows.



A:



CSLSCRQS Parameters

The parameters for the CSLSCRQS request follow.

ECB=*symbol*
ECB=*(r1-r12)*

(Optional) - Specifies the address of a z/OS ECB used for asynchronous requests. When the request is complete, the ECB specified is posted. If an ECB is not specified, the task is suspended until the request is complete. If an ECB is specified, the invoker of the macro must issue a WAIT (or equivalent) after receiving control from CSLSCRQS, before using or examining any data returned by this macro (including the RETCODE and RSNCODE fields).

FUNCTYPE=DEST
FUNCTYPE=SENDER

(Optional) - Specifies that the MBRFUNC and MBRSFUNC are defined by the DEST (destination) of this request or the SENDER of the request. This indicator is passed to the recipient of the request in the SCI Input exit parameter list.

MBRFUNC= *symbol*
MBRFUNC= *(r1-r12)*

(Required) - Specifies a 4-byte member function code that is passed to the destination of the request in the SCI Input exit parameter list. This function code, along with the MBRSFUNC, identifies the request that is being sent.

If MBRFUNC is a symbol, the symbol points to a four-byte area of storage that contains the function code.

MBRPARM= *symbol*
MBRPARM= *(r1-r12)*
(Required) - Specifies the address of a pre-built parameter list. This parameter list must be built by the requesting module and consists of sets of triplets. Each triplet describes a single parameter in the member parameter list and consists of:

parameterlength
Four-byte parameter that specifies the length of the member parameter.

parameteraddress
Four-byte parameter that specifies the address of the member parameter.

datatype
Four-byte parameter that specifies how this parameter is to be handled by SCI. Equates are provided for each type (included with CSLSCODE). These equates can be used to set the value of data type. Possible values are:

IN The parameter is an input parameter. It is copied to the destination address space with the request.

OUT The parameter is an output parameter. It is copied back to the requesting address space when the request is completed by the server. The storage for the parameter must be allocated before the request is issued.

IO The parameter is both an input and an output parameter. It is copied to the target address space with the request and it is copied back to the requesting address space when the request is complete.

SCI The parameter is an SCI allocated output parameter. The storage for the parameter is allocated in the requestor's address space when the request is complete. The address of the storage will be returned in the parameter address field and the length will be returned in the parameter length field. The storage must be released by the requestor using the CSLSCBFR request. The eight bytes immediately in front of the address returned for an SCI-allocated output parameter are available for use by the requestor. These eight bytes are not cleared, and might contain residual data from a prior use of the buffer.

The two methods for passing parameters in a parameter list are *by address* and *by value*. Both of these methods can be used when passing parameters in a CSLSCRQS request. The triplet must be setup so that SCI will handle the parameter properly.

- By address

  To pass a parameter by address, the address of the parameter must be passed in *parameteraddress* and the length of the parameter must be passed in *parameterlength*. SCI will get the parameter from *parameteraddress* for data type IN and IO and will store the parameter at *parameteraddress* for data type OUT and IO. The address at which the parameter is stored and its length is returned for data type SCI.
- By value

To pass a parameter by value, the parameter must be passed in *parameteraddress* and zero must be passed in *parameterlength*. When the length is zero, SCI will copy the value contained in *parameteraddress* to the destination for data type IN. All other data types must be passed by address since SCI requires an address to store any output parameters.

Member Parameter List: The user parameters specified here are presented to the program that receives the request in the member parameter list, the address of which is contained in the Input Exit Parm area field INXP_MBRPLPTR. Each parameter is represented by eight bytes, the first four bytes contain *parameterlength* and the second four bytes contain *parameteraddress* (if *parameteraddress* is an address, the second four bytes point to storage in the local address space, not the requesting address space). If the parameter's data type is SCI, the first four bytes will contain a length of four and the second word's value is unpredictable.

Null Parameters: In some cases a request processing module expects a set number of parameters with a defined order. If a request is to be sent that does not contain all the parameters, null parameters must be sent to ensure the data buffer contains everything that is expected. Null parameters can be sent by specifying zero for *parameterlength* and *parameteraddress*. The eights bytes that represent the parameter in the data buffer will contain zeros. This is true for any data type (IN, OUT, IO or SCI) or method of passing parameters (by address or by value).

MBRPCNT=*symbol*
MBRPCNT=*(r1-r12)*
(Required) - Specifies a 4-byte field that contains the number of member parameters that are included in MBRPARM.

MBRSFUNC=*symbol*
MBRSFUNC=*(r1-r12)***=**
(Optional) - Specifies a 4-byte member subfunction code that is passed to the destination of the request in the SCI input exit parameter list. This subfunction code, along with the MBRFUNC, identifies the request that is being sent.

If MBRSFUNC is a symbol, the symbol points to a 4-byte area of storage that contains the sub-function code.

NAME=*symbol*
NAME=*(r1-r12)*
(Optional) - Specifies the address of an 8-byte member name of the destination of this request. This name can be obtained from the Notify exit (when the member joins the IMSplex) or by issuing a CSLSCQRY request.

Note: One of the routing parameters (NAME, TOKEN or TYPE) must be included.

PARM=*symbol*
PARM=*(r1-r12)*
(Required) - Specifies the address of a parameter list used by the request to pass the parameters to SCI. The length of the storage must be at least equal to the value of SRQS_LN.

RETCODE=*symbol*
RETCODE=*(r1-r12)*
(Required) - Specifies the address of a 4-byte field to receive the CSLSCRQS

return code. SCI return codes are defined in CSLSRR. Possible return codes for CSLSCRQS are described in “CSLSCRQS Return and Reason Codes” on page 201.

RETNAME=*symbol*
RETNAME=*(r1-r12)*
(Optional) - Specifies the address of an 8-byte field to receive the name of the SCI that processes the request.

RETTOKEN=*symbol*
RETTOKEN=*(r1-r12)*
(Optional) - Specifies the address of a 16-byte field to receive the SCI token of the SCI that processes the request.

RSNCODE=*symbol*
RSNCODE=*(r1-r12)*
(Required) - Specifies the address of a 4-byte field to receive the CSLSCRQS reason code. SCI reason codes are defined in CSLSRR. Possible reason codes for CSLSCRQS are described in “CSLSCRQS Return and Reason Codes” on page 201.

SCITOKEN=*symbol*
SCITOKEN=*(r1-r12)*
(Required) - Specifies a 16-byte field containing the SCI token. This token uniquely identifies this connection to SCI. The SCI token was returned by a successful CSLSCREG FUNC=REGISTER request.

TOKEN=*symbol*
TOKEN=*(r1-r12)*
(Optional) - Specifies the address of the 16-byte SCI token of the destination of this request. This token can be obtained either from the Notify exit (when the member joins the IMSplex) or by issuing a SLSCQRY message.

Note: One of the routing parameters (NAME, TOKEN, TYPE) must be included.

TYPE=*symbol*
TYPE=*(r1-r12)*
TYPE=*'AOP'*
TYPE=*'BATCH'*
TYPE=*'CQS'*
TYPE=*'DBRC'*
TYPE=*'IMS'*
TYPE=*'IMSCON'*
TYPE=*'OM'*
TYPE=*'OTHER'*
TYPE=*'RM'*
TYPE=*'SCI'*
Input parameter that specifies the IMSplex member type of the IMSplex member to which this request should be routed. The IMSplex member type routing can be further qualified by using the SUBTYPE parameter. If TYPE is specified, SCI chooses the IMSplex member of the requested type to which the request is sent.

If member type is specified as a literal, the literal must be enclosed in single quotes. If this parameter is passed as a symbol or register, the symbol or register must contain the member type code. The member type code can be obtained by using the CSLSTPIX macro.

For a description of the IMSplex member types, see “CSLSCREG: Registration Request” on page 187.

Note: One of the routing parameters (NAME, TOKEN, TYPE) must be included.

CSLSCRQS Return and Reason Codes

Table 76 lists the return and reason codes that can be returned on a CSLSCRQS macro request. Also included is the meaning of a reason code (that is, what possibly caused it).

Table 76. CSLSCRQS Return and Reason Codes

Return Code	Reason Code	Meaning
X'00000000'	X'00000000'	The request completed successfully.
X'01000008'	X'00002004'	The function passed to the SCI interface PC routine is invalid.
	X'00002008'	The number of parameters passed was either less than or equal to zero, or greater than the maximum allowed.
	X'00002010'	An invalid type was passed.
	X'00002018'	This SCI token is invalid.
	X'00002024'	The PHDR length is invalid.
	X'00002028'	The routing data length is invalid.
	X'0000202C'	The request target member is not key 7.
	X'00002030'	The request target member is not authorized.
	X'00002034'	The length of the parameters is too large for a non-authorized caller.
	X'00002038'	The parameter list version is invalid.
X'01000010'	X'00004000'	SCI is not active.
	X'0000400C'	The destination member is not active. The destination member might have been designated by name, token, or type.
	X'0000401C'	The calling member is in the process of deregistering from SCI.
	X'00004020'	The request timed out.

Table 76. CSLSCRQS Return and Reason Codes (continued)

Return Code	Reason Code	Meaning
X'01000014'	X'00005000'	An SCI internal error occurred.
	X'00005004'	An ESTAE add error occurred.
	X'00005024'	An SRB routine error occurred.
	X'00005028'	The routing type is invalid.
	X'0000502C'	The member could not be found due to an internal BPE hash table services error.
	X'00005030'	A buffer in the destination member's address space could not be obtained.
	X'00005034'	A key 7 buffer in the SCI address space could not be obtained for a copy of PHDR and parameters.
	X'00005038'	An IEAMSCHD error occurred; an SRB could not be scheduled to the target address space.
	X'0000503C'	MRT could not be expanded.
	X'00005044'	An SCI-allocated output buffer could not be obtained.
	X'0000504C'	A message SRB key 7 parameter area could not be obtained.
	X'0000507C'	An IXCMSGO error occurred.
	X'00005500'	An abend occurred during CSLSCRQS processing.
	X'00005504'	An abend occurred while the member parameters were copied to the target address space.

Appendix A. CSL Operations Manager XML Output

Command responses that are returned through the OM API are embedded in XML tags. XML output is generated for responses to the CSLOMI, CSLOMCMD, and CSLOMQRY requests. The tags and their descriptions are described in this topic:

- "CSLOMI Output"
- "CSLOMCMD Output" on page 207
- "CSLOMQRY Output" on page 208
- "Descriptions of XML Tags Returned as CSL OM Response" on page 210

Note: The OM response is intended as a programming interface, not as an interface that produces prebuilt messages to be displayed on a screen. For OM requests, the output is passed back in the OUTPUT= buffer. For messages, the output is returned to the SCI input exit. The OM response is returned encapsulated in XML tags.

CSLOMI Output

One or more of the sets of tags in Figure 25 on page 204 is returned on each CSLOMI request:

```
<imsout>
   <ctl>
       <omname> </omname>
       <omvsn> </omvsn>
       <xmlvsn> </xmlvsn>
       <statime> </statime>
       <stotime> </stotime>
       <staseq> </staseq>
       <stoseq> </stoseq>
       <rqsttkn1> </rqsttkn1>
       <rqsttkn2> </rqsttkn2>
       <rc> </rc>
       <rsn> </rsn>
  </ctl>
  <cmdclients>
    <mbr name="membername">
      <typ> </typ>
      <styp> </styp>
      <vsn> </vsn>
      <jobname> </jobname>
   </mbr>
  </cmdclients>
  <cmdsyntax> </cmdsyntax>
  <cmddtd> </cmddtd>
  <cmdtext> </cmdtext>
  <cmderr>
    <mbr name="membername">
      <typ> </typ>
      <styp> </styp>
      <rc> </rc>
      <rsn> </rsn>
   </mbr>
  </cmderr>
  <cmdsecerr>
    <exit>
       <rc> </rc>
       <userdata> </userdata>
    </exit>
    <saf>
       <rc> </rc>
       <racfrc> </racfrc>
       <racfrsn> </racfrsn>
    </saf>
 </cmdsecerr>
 <cmd>
    <master> </master>
    <userid> </userid>
    <verb> </verb>
    <kwd> </kwd>
    <input> </input>
 </cmd>
 <cmdrsphdr>
    <hdr ... />
 </cmdrsphdr>
 <cmdrspdata>
    <rsp> </rsp>
 </cmdrspdata>
 <msgdata>
   <mbr name="membername">
     <msg> </msg>
   </mbr>
 </msgdata>
</imsout>
```

Figure 25. CSLOMI XML Output

The descriptions for each of these tags is in “Descriptions of XML Tags Returned as CSL OM Response” on page 210. Following are some examples of CSLOMI XML output. In Figure 26, the QUERY TRAN command was routed to IMSA with a timeout value of 10 seconds. See *IMS Version 9: Command Reference* for more examples of the cmdrsphdr and cmdrspdata fields.

```
OM API Input:
CMD(QUERY TRAN) NAME(SKS*)) ROUTE(IMSA) TIMEOUT(10) RQSTTKN2(QTRANCMD)

OM API Output:
<imsout>
  <ctl>
    <omname>OM1</omname>
    <omvsn>1.1.0</omvsn>
    <xmlvsn>1</xmlvsn>
    <statime>1999.341 12:52:44.46</statime>
    <stotime>1999.341 12:52:44.46</stotime>
    <staseq>B342BCC72A34D206</staseq>
    <stoseq>B342BCC75CD52208</stoseq>
    <rqsttkn2>QTRANCMD</rqsttkn2>
    <rc>0</rc> <rsn>0</rsn>
  </ctl>
  <cmd>
    <master>IMS1</master>
    <verb>QRY</verb>
    <kwd>TRAN</kwd>
    <input>QUERY TRAN</input>
  </cmd>
  <cmdrsphdr>
    <hdr slbl="TRAN" llbl="TranCode" scope="LCL" sort="a" key="1" scroll="no"
     len="8" dtype="CHAR" align="left" />
    <hdr slbl="MBR" llbl="MbrName" scope="LCL" sort="a" key="4" scroll="no"
     len="8" dtype="CHAR" align="left" />
    <hdr slbl="CC" llbl="CC" scope="LCL" key="0" scroll="YES" len="4" dtype="INT"
     align="right" />
  </cmdrsphdr>
  <cmdrspdata>
    <rsp> TRAN(SKS1) MBR(IMSA) CC(0) </rsp>
    <rsp> TRAN(SKS2) MBR(IMSA) CC(0) </rsp>
    <rsp> TRAN(SKS3) MBR(IMSA) CC(0) </rsp>
    <rsp> TRAN(SKS4) MBR(IMSA) CC(0) </rsp>
    <rsp> TRAN(SKS5) MBR(IMSA) CC(0) </rsp>
  </cmdrspdata>
</imsout>
```

Figure 26. Issue IMS Command example

In Figure 27 on page 206, the OM returns a list of client names that are currently registered for command processing.

```
OM API Input:
QUERY(CMDCLIENTS) RQSTTKN2(CLIENTLIST)
OM API Output:
<imsout>
  <ctl>
    <omname>OM1</omname>
    <omvsn>1.1.0</omvsn>
    <xmlvsn>1</xmlvsn>
    <statime>1999.341 12:52:44.46</statime>
    <stotime>1999.341 12:52:44.46</stotime>
    <staseq>B342BCC72A34D206</staseq>
    <stoseq>B342BCC75CD52208</stoseq>
    <rqsttkn2>CLIENTLIST</rqsttkn2>
    <rc>0</rc> <rsn>0</rsn>
  </ctl>
  <cmdclients>
    <mbr name=IMSA>
      <typ>DBDC</typ>
      <vsn>0800</vsn>
      <jobname>IMSJOB01</jobname>
    </mbr>
    <mbr name=IMSB>
      <typ>DBDC</typ>
      <vsn>0800</vsn>
      <jobname>IMSJOB02</jobname>
    </mbr>
  </cmdclients>
</imsout>
```

Figure 27. Query Client List example

Figure 28 on page 207 returns the command syntax for currently registered commands. In this example, the `QUERY TRAN` command is the only command registered to OM, and the keyword NAME is associated with it.

```
OM API Input:
QUERY(CMDSYNTAX) RQSTTKN2(CMDLIST)
OM API Output:
<imsout>
  <ctl>
    <omname>OM1</omname>
    <omvsn>1.1.0</omvsn>
    <xmlvsn>1</xmlvsn>
    <statime>1999.341 12:52:44.46</statime>
    <stotime>1999.341 12:52:44.46</stotime>
    <staseq>B342BCC72A34D206</staseq>
    <stoseq>B342BCC75CD52208</stoseq>
    <rqsttkn2>CMDLIST</rqsttkn2>
    <rc>0</rc> <rsn>0</rsn>
  </ctl>

  <cmdsyntax>
    <root>
      <resource name="TRAN">
        <verb name="QUERY">
          <keyword name="NAME">
            <var name="tranname*"/>
          </keyword>
        </verb>
      </resource>
    </root>
  </cmdsyntax>

  <cmdtext>
    NEXT "Next"
    BACK "Back"
    FINISH "Finish"
    CANCEL "Cancel"
    SUMMARY "Summary"
    TRAN_NAME "Transaction"
    TRAN_QUERY_NAME "Query"
    TRAN_QUERY_NAME_NAME "Name"
    TRAN_QUERY_NAME_TEXT "Name of transaction."
    TRAN_QUERY_NAME_VAR "tranname*"
  </cmdtext>
</imsout>
```

Figure 28. Query Command Syntax example

CSLOMCMD Output

The tags in Figure 29 on page 208 can be returned as a result of a CSLOMCMD request.

```
<?xml version="1.0"?>
<!DOCTYPE imsout SYSTEM "imsout.dtd">
<imsout>
   <ctl>
       <omname> </omname>
       <omvsn> </omvsn>
       <xmlvsn> </xmlvsn>
       <statime> </statime>
       <stotime> </stotime>
       <staseq> </staseq>
       <stoseq> </stoseq>
       <rqsttkn1> </rqsttkn1>
       <rc> </rc>
       <rsn> </rsn>
   </ctl>
   <cmderr>
     <mbr name="membername">
       <typ> </typ>
       <styp> </styp>
       <rc> </rc>
       <rsn> </rsn>
     </mbr>
   </cmderr>
   <cmdsecerr>
     <exit>
        <rc> </rc>
        <userdata> </userdata>
     </exit>
     <saf>
        <rc> </rc>
        <racfrc> </racfrc>
        <racfrsn> </racfrsn>
     </saf>
   </cmdsecerr>
   <cmd>
     <master> </master>
     <userid> </userid>
     <verb> </verb>
     <kwd> </kwd>
     <input> </input>
   </cmd>
   <cmdrsphdr>
     <hdr ... /hdr>
   </cmdrsphdr>
   <cmdrspdata>
      <rsp> </rsp>
   </cmdrspdata>
   <msgdata>
     <mbr name="membername">
       <msg> </msg>
     </mbr>
   </msgdata>
</imsout>
```

Figure 29. CSLOMCMD Output

CSLOMQRY Output

The tags in Figure 30 on page 209 can be returned as a result of a CSLOMQRY request.

The command syntax and translatable text that is returned as a result of the CSLOMQRY QUERY TYPE(CMDSYNTAX) request includes information for type-2 commands. For more information on these commands, see *IMS Version 9: Command Reference*.

```
<imsout>
   <ctl>
       <omname> </omname>
       <omvsn> </omvsn>
       <xmlvsn> </xmlvsn>
       <statime> </statime>
       <stotime> </stotime>
       <staseq> </staseq>
       <stoseq> </stoseq>
       <rqsttkn1> </rqsttkn1>
       <rc> </rc>
       <rsn> </rsn>
   </ctl>
   <cmdclients>
     <mbr name="membername">
       <typ> </typ>
       <styp> </styp>
       <vsn> </vsn>
       <jobname> </jobname>
     </mbr>
   </cmdclients>
   <cmdsyntax> </cmdsyntax><cmddtd>
       <!ELEMENT imsout ( ctl, cmdclients?,   cmdsyntax?, cmddtd?,
       cmdtext?, cmderr?, cmd, cmdrsphdr, cmdrspdata?, msgdata? )>
       <!ELEMENT ctl (omname?, omvsn?, xmlvsn?,   stattime, stotime,
       statseq, stoseq,   rqsttkn1?, rqsttkn 2?, rc, rsn )>
       <!ELEMENT omname (#PCDATA) >
       <!ELEMENT omvsn (#PCDATA) >
       <!ELEMENT xmlvsn (#PCDATA) >
       <!ELEMENT statime (#PCDATA) >
       <!ELEMENT stotime (#PCDATA) >
       <!ELEMENT staseq (#PCDATA) >
       <!ELEMENT stoseq (#PCDATA) >
       <!ELEMENT rqsttkn1 (#PCDATA) >
       <!ELEMENT rqsttkn2 (#PCDATA) >
       <!ELEMENT rc (#PCDATA) >
       <!ELEMENT rsn (#PCDATA) >
       <!ELEMENT cmdclients ( mbr+ )>
       <!ELEMENT cmdsyntax (#PCDATA) >
       <!ELEMENT cmddtd (#PCDATA ) >
       <!ELEMENT cmdtext (#PCDATA) >
       <!ELEMENT cmderr ( mbr* )>
       <!ELEMENT MBR ( (TYP, STYP, ((VSN, JOBNAME) | (rc, rsn))) | msg)>
```

Figure 30. CSLOMQRY Output (Part 1 of 2)

```
        <!ELEMENT typ (#PCDATA) >
        <!ELEMENT styp (#PCDATA) >
        <!ELEMENT vsn (#PCDATA) >
        <!ELEMENT jobname (#PCDATA) >
        <!ELEMENT msg (#PCDATA) >
        <!ELEMENT cmdsecerr ( exit, saf )>
        <!ELEMENT exit ( rc, userdata ) >
        <!ELEMENT saf ( rc, racfc, racfrsn )>
        <!ELEMENT userdata (#PCDATA) >
        <!ELEMENT racfc (#PCDATA) >
        <!ELEMENT racfrsn (#PCDATA) >
        <!ELEMENT cmd ( master?, userid?, verb, kwd, input )>
        <!ELEMENT master (#PCDATA) >
        <!ELEMENT userid (#PCDATA) >
        <!ELEMENT verb (#PCDATA) >
        <!ELEMENT kwd (#PCDATA) >
        <!ELEMENT input (#PCDATA) >
        <!ELEMENT cmdrsphdr ( hdr* ) >
        <!ELEMENT hdr (#PCDATA) >
        <!ELEMENT cmdrspdata ( rsp* ) >
        <!ELEMENT rsp (#PCDATA) >
        <!ELEMENT msgdata ( mbr ) >
        <!ATTLIST hdr slbl   CDATA #REQUIRED >
        <!ATTLIST hdr llbl   CDATA #REQUIRED >
        <!ATTLIST hdr scope  CDATA #REQUIRED >
        <!ATTLIST hdr sort   CDATA #REQUIRED >
        <!ATTLIST hdr key    CDATA #REQUIRED >
        <!ATTLIST hdr scroll CDATA #REQUIRED >
        <!ATTLIST hdr len    CDATA #REQUIRED >
        <!ATTLIST hdr dtype  CDATA #REQUIRED >
        <!ATTLIST hdr align  CDATA #REQUIRED >
</cmddtd>
<cmdtext> </cmdtext>
</imsout>
```

Figure 30. CSLOMQRY Output (Part 2 of 2)

Descriptions of XML Tags Returned as CSL OM Response

Following are the descriptions of the possible XML tags returned as an OM response. The tag names are delimited by the characters “<” and “>”. Data or other sets of tags are contained between these start and end tags, respectively. In the list of tags, indentation indicates that the tags are nested within the parent tags.

<?xml version ″1.0″?>
The version of XML used in this output.

<!DOCTYPE imsout SYSTEM ″imsout.dtd″>
The DOCTYPE tag identifies the file that contains the document type definition (DTD). The DTD describes the structure that is supported for this type of XML document. The imsout.dtd file can be accessed using the DB2 Universal Database Control Center. Users of z/OS can find the DTD information in IMS.SDFSRESL(CSLOMDTD).

<imsout> </imsout>
The <imsout> </imsout> tags encapsulate the output from OM. These tags are returned on every request.

<ctl> </ctl>

The <ctl> </ctl> tags encapsulate the control information returned by OM. These tags are returned on every request. The control information consists of the following:

<omname>*om name***</omname>**

Indicates the name of the OM that processed this request. The name is specified on the OMNAME= execution parameter of the CSLOIxxx PROCLIB member.

<omvsn>*om version number***</omvsn>**

Indicates the OM version number.

<xmlvsn>*xml version number***</xmlvsn>**

Indicates the XML version number.

<statime>*starttime***</statime>**

Indicates the time that OM started processing the request. The field is in the following format: yyyy.ddd hh:mm:ss.th

<stotime>*stoptime***</stotime>**

Indicates the time that OM completed request processing. The field is in the following format: yyyy.ddd hh:mm:ss.th

<staseq>*startsequence***</staseq>**

Indicates the sequence value when OM started processing the request. This value can be used for sorting. It is in printable EBCDIC hexadecimal format.

<stoseq>*stopsequence***</stoseq>**

Indicates the sequence value when OM stopped processing the request. This value can be used for sorting. It is in printable EBCDIC hexadecimal format.

<rqsttkn1>*requesttoken1***</rqsttkn1>**

Indicates the user specified RQSTTKN1 value associated with the response. OM converts unprintable characters to periods (.) in the output.

<rqsttkn2>*requesttoken2***</rqsttkn2>**

Indicates the user specified RQSTTKN2 value associated with the response. OM converts unprintable characters to periods (.) in the output.

<rc>*returncode***</rc>**

The return code for the request in printable EBCDIC hexadecimal format.

<rsn>*reasoncode***</rsn>**

The reason code for the request in printable EBCDIC hexadecimal format.

<cmdclients> </cmdclients>

The <cmdclients> </cmdclients> tags encapsulate information about OM clients. These tags can be returned on a QUERY(CMDCLIENTS) request.

<mbr name=*"membername"***></mbr>**

Indicates the name of the IMSplex member that is registered for commands.

<typ>*membertype***</typ>**

Indicates the type of IMSplex member.

<styp>*membersubtype***</styp>**
Indicates the IMSplex member subtype. OM converts unprintable characters to periods (.) in the output.

<vsn>*memberversion***</vsn>**
Indicates the member version number.

<jobname>*memberjobname***</jobname>**
Indicates the member jobname.

<cmddtd> </cmddtd>
The <cmddtd> </cmddtd> tags encapsulate the Document Type Definition (DTD) defined by OM for command syntax and OM output XML. These tags can be returned on a QUERY(CMDSYNTAX) request.

<cmdsyntax> </cmdsyntax>
The <cmdsyntax> </cmdsyntax> tags encapsulate the XML definitions for the commands that are registered to OM from all of its clients. These tags can be returned on a QUERY(CMDSYNTAX) request.

<cmdtext> </cmdtext>
The <cmdtext> </cmdtext> tags encapsulate the translatable text strings associated with the XML command syntax tags. These tags can be returned on a QUERY(CMDSYNTAX) request.

<cmd> </cmd>
The <cmd> </cmd> tags encapsulate the command information that was passed to OM. These tags can be returned on a command request. The output returned in these tags is what was provided on the CMD= parameter on the CSLOMBLD macro. The following tags are included within the <cmd> tags:

<master> </master>
The <master> </master> tags encapsulate the name of the command processing client that was tagged as the master when sending the command. This information will not be present unless the command was successfully sent to at least one command processing client.

<userid> </userid>
The <userid> </userid> tags encapsulate the user ID of the originator of the command.

<verb> </verb>
The <verb> </verb> tags encapsulate the short form of the command verb that was processed by OM. The verb might have been passed to OM in a long form.

<kwd> </kwd>
The <kwd> </kwd> tags encapsulate the command keyword that was processed by OM.

<input> </input>
The <input> </input> tags encapsulate the actual input command string that was passed to OM.

<cmdrsphdr> </cmdrsphdr>
The <cmdrsphdr> </cmdrsphdr> tags encapsulate the command header information which describes the data fields returned in the command response. These tags can be returned on a command request.

<hdr ... />

Defines the attributes of columns of data fields.

The command header information is in the format shown in Figure 31:

```
<hdr slbl="ss" llbl="llll" scope="c" sort="d" key="e" scroll="f" len="g" dtype="h" align="i" skipb="no"/>
```

Figure 31. Command Response Header Format

slbl Short label used to match data description with data values returned by <cmdrspdata>.

The short label values vary by command. Refer to the documentation for each command to determine what values can be returned for a specific command.

llbl Long label which can be used as the table column header.

The long label values vary by command. Refer to the documentation for each command to determine what values can be returned for a specific command.

scope Indicates if the data is global or local.

GBL Indicates that the data is global. For query output, global data applies to all resources of the same name, but is only returned once in the command response for a specific resource name. Global information applies to other rows of the same resource name for different IMSplex member names. The resource name is the data field identified by a KEY=″1″ attribute. If an application chooses to transform the command response data into a table to be displayed for a user, the global data value can be propagated to other rows for the same resource name.

LCL Indicates that the data is local. For query output, local data applies only to a specific resource name in a specific IMS. Different IMS systems can return different values for local data fields. Each IMS returns its local value when it is available. If an application chooses to transform the command response data into a table to be displayed for a user, the local data value should not be propagated to other rows for the same resource name.

sort Indicates whether or not this field should be sorted and the sort direction.

A Sort in ascending order.

D Sort in descending order.

N Do not sort field.

key Indicates the sort priority for this field.

0 Field is not sorted.

1 The highest priority sort field.

2 The second highest priority sort field.

n The *n*th priority sort field.

The priority value indicated on KEY= in the <cmdrsphdr> tag has been predetermined. Some command responses can specify multiple sort fields. In Figure 33 on page 215, several fields are listed within the <cmdrsphdr> tags with their sort priorities:

- Trancode - 1
- MbrName - 4
- CC - 0
- PSBname - 0
- QCnt - 2
- LCls - 0
- LQCnt - 3

Figure 33 on page 215 shows that only Trancode, MbrName, QCnt, and LQcnt will be used to sort the command results. The sort priority, therefore, will be:

1. Trancode
2. Qcnt
3. LQcnt
4. MbrName

If two records have the same Trancode, they will be sorted by Qcnt. If they also have the same Qcnt, they are sorted by LQcnt. If they have the same LQcnt value, they are sorted by MbrName, and so on, until the *n*th sort field is used.

The results of the XML in Figure 33 on page 215 are displayed in Figure 32.

Depending on which fields were selected using the SHOW

```
Response for: QRY TRAN NAME(ADD*) SHOW(PSB,QCNT,CLASS)

Trancode MbrName    CC PSBname      QCnt LCls    LQCnt
ADDINV   IMS2        0                 0
ADDINV   SYS3        0 DFSSAM04             4        3
ADDINV   IMS2        0 DFSSAM04             4        0
ADDPART  IMS2        0                 0
ADDPART  IMS2        0 DFSSAM04             4        0
ADDPART  SYS3        0 DFSSAM04             4        0
```

Figure 32. Sorted Results

parameter of the QUERY command, not all intermediate priority value fields will be displayed. That is, the results could display fields whose priority values were set at 1 and 4, but not display fields whose priority values were set at 2 and 3. A program might leave the records in the original order, sort them using the predetermined priority values, or sort by other fields using criteria set locally by the user.

```
<?xml version="1.0"?>
<!DOCTYPE imsout SYSTEM "imsout.dtd">
<imsout>
<ctl>
<omname>OM1OM   </omname>
<omvsn>1.1.0</omvsn>
<xmlvsn>1   </xmlvsn>
<statime>2002.261 18:33:56.425140</statime>
<stotime>2002.261 18:33:56.487941</stotime>
<staseq>B8400987409B4A0E</staseq>
<stoseq>B84009874FF05409</stoseq>
<rqsttkn1>USRT002 10113356</rqsttkn1>
<rc>00000000</rc>
<rsn>00000000</rsn>
</ctl>
<cmd>
<master>IMS2    </master>
<userid>USRT002 </userid>
<verb>QRY </verb>
<kwd>TRAN            </kwd>
<input>QRY TRAN NAME(ADD*) SHOW(PSB,QCNT,CLASS) </input>
</cmd>
<cmdrsphdr>
<hdr slbl="TRAN" llbl="Trancode" scope="LCL" sort="a" key="1"
scroll="no" len="8" dtype="CHAR" align="left" />
<hdr slbl="MBR" llbl="MbrName" scope="LCL" sort="a" key="4"
scroll="no" len="8" dtype="CHAR" align="left" />
<hdr slbl="CC" llbl="CC" scope="LCL" sort="n" key="0"
scroll="yes" len="4" dtype="INT" align="right" />
<hdr slbl="PSB" llbl="PSBname" scope="LCL" sort="n" key="0"
scroll="yes" len="8" dtype="CHAR" align="left" />
<hdr slbl="Q" llbl="QCnt" scope="GBL" sort="d" key="2"
scroll="yes" len="8" dtype="INT" align="right" />
<hdr slbl="LCLS" llbl="LCls" scope="LCL" sort="n" key="0"
scroll="yes" len="3" dtype="INT" align="right" />
<hdr slbl="LQ" llbl="LQCnt" scope="LCL" sort="d" key="3"
scroll="yes" len="8" dtype="INT" align="right" /></cmdrsphdr>
<cmdrspdata>
<rsp>TRAN(ADDPART ) MBR(IMS2    ) CC(   0) PSB(DFSSAM04) LCLS(  4)
LQ(   0) </rsp>
<rsp>TRAN(ADDINV  ) MBR(IMS2    ) CC(   0) PSB(DFSSAM04) LCLS(  4)
LQ(   0) </rsp>
<rsp>TRAN(ADDPART ) MBR(IMS2    ) CC(   0) Q(    0) </rsp>
<rsp>TRAN(ADDINV  ) MBR(IMS2    ) CC(   0) Q(    0) </rsp>
<rsp>TRAN(ADDPART ) MBR(SYS3    ) CC(   0) PSB(DFSSAM04) LCLS(  4)
LQ(   0) </rsp>
<rsp>TRAN(ADDINV  ) MBR(SYS3    ) CC(   0) PSB(DFSSAM04) LCLS(  4)
LQ(   3) </rsp>
</cmdrspdata>
</imsout>
```

Figure 33. Sample XML to Illustrate KEY=

scroll Indicates whether or not this field should be scrolled off of the screen when TSO SPOC shifts the screen to the right.

NO Do not scroll field.

YES Allow field to scroll off the screen.

len Maximum length of data (data returned could contain fewer characters). If a table of data is being created from the output response, this value can be used to determine the width of the column that is displayed for this attribute. If the value for this field is '*', this is a variable length field.

dtype Describes the original data type. All data is returned in character format. However, some fields represent numeric data. Data that originated as integer might need to be converted from character to integer in order to perform mathematical calculations.

CHAR The output field represents character data.

INT The output field is the character representation of integer data.

align Indicates recommended column alignment if data is to be formatted into columns.

RIGHT
Data is right aligned, for example, numeric data

CENTER
Data is centered

LEFT Data is left aligned, for example, character data

skipb

no The column is displayed on the TSO SPOC output, even if no client returned any information for this column. This is the default.

yes The column is not displayed on the TSO SPOC output if no client returned any information for this column.

<cmdrspdata> </cmdrspdata>
The <cmdrspdata> </cmdrspdata> tags encapsulate the command response detail information. These tags can be returned on a command request. The <cmdrspdata> </cmdrspdata> tags contain the actual data that is described by the <cmdrsphdr> </cmdrsphdr> tags.

Refer to the documentation for each command to determine what values can be returned for a specific command.

<rsp>*response data***</rsp>**

Contains a logical line of command response output for a particular resource. The response data contains various tags in the form name(value). The name maps to short label (slbl=) values in the <hdr> tag. This is shown in Figure 34, with the values TRAN and PSB.

```
<cmdrsphdr>
<hdr slbl="TRAN"    llbl="Trancode"... />
<hdr slbl="PSB"     llbl="PSBname" ... />
</cmdrsphdr>
<cmdrspdata>
<rsp>TRAN(A        ) PSB(A11      ) </rsp>
<rsp>TRAN(B        ) PSB(B22      ) </rsp>
<rsp>TRAN(C        ) PSB(C33      ) </rsp>
</cmdrspdata>
```

Figure 34. <cmdrsphdr> Sample Tags

The <hdr> tag includes a long label value (llbl=), which can be used as column headings. This is shown in Figure 35, specifically Trancode and PSBname.

```
Trancode  PSBname
A         A11
B         B22
C         C33
```

Figure 35. SPOC Output from <cmdrsphdr>

The values included in the response data propagate the data columns of the SPOC output. Other tags in the <hdr> tag describe formatting attributes for values in that column.

<msgdata> </msgdata>
The <msgdata> </msgdata> tags encapsulate prebuilt IMS messages. The messages can be of any type including informational, warning, or error messages. These tags can be returned on a command request.

<mbr name=*"membername"***></mbr>**
Indicates the name of the IMSplex member that returned the message.

<msg>*message data***</msg>**
Contains a logical command response output for a resource in a message format. The message starts with a message number (for example,DFSnnnnI). There is no LL field or X'15' new line character.

<cmderr> </cmderr>
The <cmderr> </cmderr> tags encapsulate the return and reason code information returned by OM or a command processing client. These tags are returned on command requests when an error specific to a command processing client must be returned. For each IMSplex member with an error the following information is returned.

<mbr name=*"membername"***></mbr>**
Indicates the name of the IMSplex member for which an error was detected.

<typ>*membertype***</typ>**
Indicates the type of IMSplex member.

<styp>*membersubtype***</styp>**
Indicates the IMSplex member subtype. OM converts unprintable characters to periods (.) in the output.

<rc>*returncode***</rc>**
Indicates the return code for the IMSplex member in printable EBCDIC hexadecimal format.

<rsn>*reasoncode***</rsn>**
Indicates the reason code for the IMSplex member in printable EBCDIC hexadecimal format.

<cmdsecerr> </cmdsecerr>
The <cmdsecerr> </cmdsecerr> tags encapsulate the return and reason code information returned by the OM SECURITY exit, SAF and RACF, or

equivalent. If the OM SECURITY exit rejected the command for any reason, the user data from the SECURITY exit is also encapsulated here.

<exit> </exit>
Encapsulates the return code and user data from the OM SECURITY exit.

<rc>*returncode***</rc>**
Indicates the return code from the OM SECURITY exit in printable EBCDIC hexadecimal format.

<userdata>*userdata***</userdata>**
Indicates the user data returned from the OM SECURITY exit in the OSCX_USERDATA field of the OM Command Security User Exit parameter list (CSLOSCX). OM converts unprintable characters to periods (.) in the output.

<saf> </saf>
Encapsulates the return and reason codes from the SAF and RACF or equivalent.

<rc>*returncode***</rc>**
Indicates the return code from the SAF in printable EBCDIC hexadecimal format.

<racfrc>*racfreturncode***</racfrc>**
Indicates the return code from RACF or equivalent in printable EBCDIC hexadecimal format.

<racfrsn>*racfreasoncode***</racfrsn>**
Indicates the reason code from RACF or equivalent in printable EBCDIC hexadecimal format.

Appendix B. REXX SPOC API and the CSL

This topic describes the REXX SPOC API. It includes:

- "Using the REXX SPOC API Environment with the CSL OM"
- "REXX SPOC Return and Reason Codes" on page 221
- "REXX Samples and Examples" on page 222

Using the REXX SPOC API Environment with the CSL OM

The REXX SPOC API allows REXX programs to submit commands to OM and to retrieve the command responses. There are 3 phases related to the REXX SPOC API:

1. Set up the REXX environment
2. Set up the IMSplex environment and issue commands
3. Retrieve the command responses

Each of these phases is described in this topic.

Setting Up the REXX Environment in a CSL

To set up the REXX environment, call program CSLULXSB using the `ADDRESS` command. This program establishes the REXX subcommand environment for the REXX SPOC API.

Note: Other forms of the `ADDRESS` command might not work in the Tivoli NetView for z/OS environment.

Setting Up the IMSplex Environment

After setting up the REXX environment using CSLULXSB, you can set up the IMSplex environment. To set up the IMSplex environment and begin to issue commands, switch the default host command to IMSSPOC using the address command.

After you set the default host command to IMSSPOC, IMSSPOC executes subsequent host commands issued by the REXX program that is running. You can switch to other host commands by using the ADDRESS command with other hosts. For example:

```
ADDRESS TSO
ADDRESS MVS
ADDRESS ISPEXEC
```

You can then issue commands specific to those environments.

Note: If you issue commands other than the subcommands described here in the REXX environment, they are sent to OM for processing.

IMS Subcommand

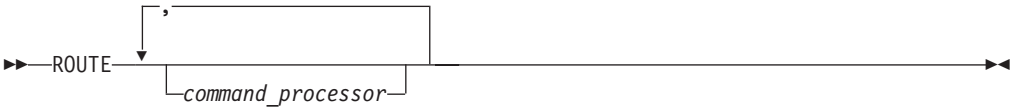
The IMS subcommand establishes the name of the IMSplex. You must issue the IMS subcommand to establish the IMSplex name before any other commands can be issued. A prefix of “CSL” is automatically added to the name that you specify.

ROUTE Subcommand

Use the ROUTE subcommand to set the name of the command processors. The command processors are the specific systems that will execute subsequent IMS commands. If you do not specify a command processor:

- the previous routing value is removed
- commands will be routed to all members of the IMSplex (this is the default).

The ROUTE subcommand is optional.



CART Subcommand

Use the Command and Response Token (CART) subcommand to set the name of the command and response token. This 16-character text string token is used to retrieve the command response.

You must issue the CART subcommand before you can issue any IMS commands.

WAIT Subcommand

Use the WAIT subcommand to provide a timeout value to OM. The time value must be in the form MMM:SS or ssss. The maximum value you can specify is 999:59. The WAIT subcommand is optional

Issuing Type-2 IMS Commands

You issue IMS commands, including type-2 commands, by including them in the REXX program stream as quoted strings or as REXX variables that resolve to quoted strings. Examples of commands are shown in Figure 36.

```
"QUERY IMPLEX SHOW(ALL)"
"DIS ACT"
tranlist = "PETER1,MATT1"
"QUERY TRAN NAME("tranlist")"
```

Figure 36. Examples of type-2 commands

Retrieving Command Responses

Use CSLULGTS to retrieve command responses. CSLULGTS puts the command responses to a stem variable so that REXX can access them.

```
►►—CSLULGTS—(—stem_name,token_name,"wait_time"—)————————————————————►◄
```

stem_name
After the CSLULGTS completes successfully, the stem variable contains XML statements. Each row of the stem variable contains one XML statement. If the beginning and ending XML tags are adjacent (that is, no other XML tags exist between them), they are placed in the same row of the stem variable. A single row of a stem variable might look like this:

```
<rsp>TRAN(VIDB    ) MBR (IMS2    ) CC(    0) </rsp>
```

token_name
token_name is the name of the command and response token (CART). It should match the name specified on the CART subcommand.

wait_time
wait_time is a time out value for CSLULGTS. CSLULGTS waits until the command completes, but the wait lasts only as long as the time specified. The wait time is in the format MMM:SS, or ssss. The maximum time out value is 999:59. Enclose this value in quotes.

Note: This time out value is not the same as the time out value for the WAIT subcommand; however, this *wait_time* should be at least as long as the value specified on the WAIT subcommand. Otherwise, no command response will be received for long running commands.

If no response was received the first time, CSLULGTS can be issued again.

Ending the IMSSPOC Environment

You can end the IMSSPOC environment when you no longer want to execute IMS commands. Use the END subcommand to signify that the SPOC environment is no longer needed. After the END subcommand is issued, the control blocks associated with the SPOC environment are freed.

Note: END is a valid IMS command. If END is specified with no operands, it is treated as an IMS SPOC subcommand. If END is specified with parameters, it is sent to the IMSplex for processing as an IMS command.

```
►►—END————————————————————————————————————————————————————————————►◄
```

REXX SPOC Return and Reason Codes

Return and reason codes issued from the REXX SPOC are described in Table 77 on page 222. Note that the return and reason codes are character values, not hexadecimal values. The X at the end of the code is for easier reading. Also included is the meaning of a reason code (that is, what possibly caused it).

IMSSPOC commands, IMS commands, and the CSLULGTS command can set special variables. These variables are IMSRC and IMSREASON. Refer to those variables if the standard REXX return code is non-zero.

Table 77. REXX SPOC Return and Reason Codes

Return Code	Reason Code	Meaning
0000000X		The request completed successfully.
08000004X	00001000X	The command is still executing.
08000008X	00002000X	The wait value was missing or invalid.
	00002008X	The IMSplex value was missing or invalid.
	00002012X	The STEM name was missing or invalid.
	00002016X	The token name was missing or invalid.
	00002020X	Too many parameters were specified.
	00002024X	The request token could not be found.
	00002028X	The CART value was missing or invalid.
08000010X		An environmental error occurred.
08000014X	00004000X	A GETMAIN failure occurred.

REXX Samples and Examples

This topic provides both sample programs and examples for REXX SPOC environments.

Sample REXX Program

A sample REXX program is provided in Figure 37 on page 223.

```
Address LINK 'CSLULXSB'
    Address IMSSPOC
    /*-------------------------------------------------------------
    | 'ims' defines the IMSplex that receives the commands        |
    |                                                             |
    | 'route' defines which IMSplex members in the IMSplex        |
    | receives the commands.  If ROUTE is not specified or if     |
    | ROUTE * is specified, commands are routed to all IMSplex    |
    | members.                                                    |
    |                                                             |
    | 'wait' provides a timeout value to OM.  The time is in      |
    |  mmm:ss format (or ssss if no colon is specified).          |
    |                                                             |
    | 'cart' establishes the command response token for subsequent|
    | commands.                                                   |
    |                                                             |
    | 'end' frees control blocks                                  |
    |                                                             |
    -------------------------------------------------------------*/
    "IMS  IPLX4"
    "ROUTE IMS1,IMSB"

    "WAIT  5:00"

    "CART DISTRAN"
    "/DIS TRAN PART"

    /*-------------------------------------------------------------
     | The cslulgts function retrieves data associated with a     |
     | a specific token and fills in a REXX stem variable.  In    |
     | this example, it waits 59 seconds.                         |
     |                                                            |
     | The XML statements returned are put in the stem variable   |
     | specified by the user.                                     |
     |                                                            |
     -------------------------------------------------------------*/
     spoc_rc = cslulgts('DISINFO.','DISTRAN',"59")
     do z1 = 1 to  DISINFO.0
       /* display each line of XML information */
       Say disinfo.z1
     end
     "END"
```

Figure 37. Sample REXX Program

REXX Batch Job Example

This topic provides a sample batch job, REXX program, and job output.

The batch job shown in Figure 38 on page 224 calls the batch TSO command processor.

```
//REXXSPOC JOB ,
//   MSGCLASS=H,NOTIFY=USRT002,USER=USRT002,TIME=(,30)
//*
//SPOC     EXEC PGM=IKJEFT01,DYNAMNBR=45
//STEPLIB  DD DISP=SHR,DSN=IMS810.SDFSRESL
//SYSPROC  DD DISP=SHR,DSN=LOCAL.IMS.CLIST
//SYSTSPRT DD SYSOUT=A
//SYSTSIN  DD *
  %REXXSPOC QRY TRAN NAME(V*)
/*EOF
```

Figure 38. Sample batch job

The DD names in this batch job include:

STEPLIB
Contains the load modules.

SYSPROC
Contains the REXX programs.

SYSTSPRT
Contains the output produced by the REXX program.

SYSTSIN
Contains the command to execute, including its parameters.

The QRY TRAN command in the JCL is passed as an argument to the sample REXX program. The command is issued, and the response is sent to the SYSTSPRT file.

Figure 39 shows the sample REXX program, REXXSPOC.

```
/* rexx */
parse upper arg theIMScmd
Address LINK 'CSLULXSB'
if rc = 0 then
 do
   Address IMSSPOC
   "IMS plex1" ; if rc > 0 then say 'rc='imsrc 'reason='imsreason
   "route ims2"; if rc > 0 then say 'rc='imsrc 'reason='imsreason
   cartid = "TEST13"
   "cart" cartid ; if rc > 0 then say 'rc='imsrc 'reason='imsreason
   "WAIT 1:00" ; if rc > 0 then say 'rc='imsrc 'reason='imsreason
   theIMScmd
   if rc > 0 then say 'rc='rc 'imsrc='imsrc 'reason='imsreason
      do
        results = cslulgts('TEMP.', cartid,"1:30")
        say 'results='results ' imsrc='imsrc ' reason='imsreason
        if temp.0 /= '' then
          do
            say 'temp.'0'=('temp.0')'
            do idx = 1 to temp.0
              say 'temp.'idx'= 'temp.idx
            end
          end
      end
"END"
End
exit
```

Figure 39. REXXSPOC sample program

The output from the REXXSPOC sample program is shown in Figure 40.

```
READY
%REXXSPOC QRY TRAN NAME(V*)
results=00000000X imsrc=00000000X reason=00000000X
temp.0=(30)
temp.1= <imsout>
temp.2= <ctl>
temp.3= <omname>OM1OM </omname>
temp.4= <omvsn>1.1.0</omvsn>
temp.5= <xmlvsn>1 </xmlvsn>
temp.6= <statime>2001.198 16:08:39.944953</statime>
temp.7= <stotime>2001.198 16:08:40.271944</stotime>
temp.8= <staseq>B625CACD49AF914A</staseq>
temp.9= <stoseq>B625CACD99848CC6</stoseq>
temp.10= <rqsttkn1>TEST13 </rqsttkn1>
temp.11= <rc>00000000</rc>
temp.12= <rsn>00000000</rsn>
temp.13= </ctl>
temp.14= <cmd>
temp.15= <master>IMS2 </master>
temp.16= <userid>USRT002 </userid>
temp.17= <verb>QRY </verb>
temp.18= <kwd>TRAN </kwd>
temp.19= <input>QRY TRAN NAME(V*)</input>
temp.20= </cmd>
temp.21= <cmdrsphdr>
temp.22= <hdr slbl="TRAN" llbl="Trancode" scope="LCL" sort="a"
key="1" scroll="no" len="8" dtype=" CHAR" align="left" />
temp.23= <hdr slbl="MBR" llbl="MbrName" scope="LCL" sort="a"
key="4" scroll="no" len="8" dtype="CHAR" align="left" />
temp.24= <hdr slbl="CC" llbl="CC" scope="LCL" sort="n"
key="0" scroll="yes" len="4" dtype="INT" align="right" />
temp.26= </cmdrsphdr>
temp.26= <cmdrspdata>
temp.27= <rsp>TRAN(VIDB ) MBR(IMS2 ) CC( 0) </rsp>
temp.28= <rsp>TRAN(VIDA ) MBR(IMS2 ) CC( 0) </rsp>
temp.29= </cmdrspdata>
temp.30= </imsout>
READY
END
```

Figure 40. Sample Output

Autonomic Computing Examples

In this topic, two examples are provided that illustrate autonomic computing capabilities associated with the REXX SPOC API. *Autonomic* indicates that the code is responsive and can take certain actions to correct what it determines to be incorrect.

Autonomic Example 1

In Figure 41 on page 226, a transaction is queried. If the transaction is stopped, the REXX SPOC API attempts to start it. The REXX SPOC API examines the information returned by CSLULGTS, looking specifically for the line that refers to the transaction of interest.

```
/* autonomic computing example 1 */
"CART  qrytran12"
"qry tran name(CDEBTRN3) show(status)"
results = cslulgts("resp.","qrytran12","3:15")

Do idx = 1 to resp.0
  /* find which IMS and the status of tran */
  parse var resp.idx . "TRAN(CDEBTRN3" . ,
                        "MBR(" imsname ")" . ,
                       "LSTT(" status ")" .

  /* if tran is stoppped, try to start it */
  If pos('STOSCHD', status)  > 0 Then
    Do
      /* send command to IMS that needs to restart tran */
      "ROUTE"  imsname
      "UPD TRAN NAME(CDEBTRN3) START(SCHD)"
    End
End
```

Figure 41. Autonomic Example 1

Autonomic Example 2

In Figure 42, the QUERY command is used to determine the queue count (qcnt) of a transaction. A qcnt with a value greater than 5 is considered a problem. The REXX SPOC API attempts to resolve the problem by starting another region and changing the transaction to a different class.

```
/* autonomic computing example 2  */
"CART  qrytran13"
"qry tran name(sks1) show(qcnt)"
results = cslulgts("resp.", "qrytran13", "3:15")
Do idx = 1 to resp.0
   parse var resp.idx . "TRAN(SKS1" . "Q(" count ")" .
   If count ≠ ''  &,
      count  > 5 Then
    Do
       "CART  strtrgn05"
       "START REGION IMSRG5"
       start? = cslulgts("strt.", "strtrgn05", "10:00")
       if imsrc = '00000000X' then
         Do
           "CART  updtran14"
           "update tran name(SKS1) set(class(5))"
         End
     End
End
"END"
```

Figure 42. Autonomic Example 2

Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs

and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
J46A/G4
555 Bailey Avenue
San Jose, CA 95141-1003
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

BookManager	MVS
CICS	NetView
DataPropagator	OS/390
DB2	RACF
DB2 Universal Database	Tivoli
IBM	WebSphere
IMS	z/OS

Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc., in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, and service names may be trademarks or service marks of others.

Bibliography

This bibliography lists all of the information in the IMS Version 9 library.

- *z/OS MVS: Initialization and Tuning Guide*, SA22-7592
- *z/OS MVS: Setting Up a Sysplex*, SA22-7625

IMS Version 9 Library

Title	Acronym	Order number
IMS Version 9: Administration Guide: Database Manager	ADB	SC18-7806
IMS Version 9: Administration Guide: System	AS	SC18-7807
IMS Version 9: Administration Guide: Transaction Manager	ATM	SC18-7808
IMS Version 9: Application Programming: Database Manager	APDB	SC18-7809
IMS Version 9: Application Programming: Design Guide	APDG	SC18-7810
IMS Version 9: Application Programming: EXEC DLI Commands for CICS and IMS	APCICS	SC18-7811
IMS Version 9: Application Programming: Transaction Manager	APTM	SC18-7812
IMS Version 9: Base Primitive Environment Guide and Reference	BPE	SC18-7813
IMS Version 9: Command Reference	CR	SC18-7814
IMS Version 9: Common Queue Server Guide and Reference	CQS	SC18-7815
IMS Version 9: Common Service Layer Guide and Reference	CSL	SC18-7816
IMS Version 9: Customization Guide	CG	SC18-7817
IMS Version 9: Database Recovery Control (DBRC) Guide and Reference	DBRC	SC18-7818
IMS Version 9: Diagnosis Guide and Reference	DGR	LY37-3203
IMS Version 9: Failure Analysis Structure Tables (FAST) for Dump Analysis	FAST	LY37-3204
IMS Version 9: IMS Connect Guide and Reference	CT	SC18-9287
IMS Version 9: IMS Java Guide and Reference	JGR	SC18-7821
IMS Version 9: Installation Volume 1: Installation Verification	IIV	GC18-7822
IMS Version 9: Installation Volume 2: System Definition and Tailoring	ISDT	GC18-7823
IMS Version 9: Master Index and Glossary	MIG	SC18-7826
IMS Version 9: Messages and Codes, Volume 1	MC1	GC18-7827
IMS Version 9: Messages and Codes, Volume 2	MC2	GC18-7828
IMS Version 9: Open Transaction Manager Access Guide and Reference	OTMA	SC18-7829
IMS Version 9: Operations Guide	OG	SC18-7830
IMS Version 9: Release Planning Guide	RPG	GC17-7831
IMS Version 9: Summary of Operator Commands	SOC	SC18-7832
IMS Version 9: Utilities Reference: Database and Transaction Manager	URDBTM	SC18-7833
IMS Version 9: Utilities Reference: System	URS	SC18-7834

Supplementary Publications

Title	Order number
IMS Connector for Java 2.2.2 and 9.1.0.1 Online Documentation for WebSphere Studio Application Developer Integration Edition 5.1.1	SC09-7869
IMS Version 9 Fact Sheet	GC18-7697
IMS Version 9: Licensed Program Specifications	GC18-7825

Publication Collections

Title	Format	Order number
IMS Version 9 Softcopy Library	CD	LK3T-7213
IMS Favorites	CD	LK3T-7144
Licensed Bill of Forms (LBOF): IMS Version 9 Hardcopy and Softcopy Library	Hardcopy and CD	LBOF-7789
Unlicensed Bill of Forms (SBOF): IMS Version 9 Unlicensed Hardcopy Library	Hardcopy	SBOF-7790

Title	Format	Order number
OS/390 Collection	CD	SK2T-6700
z/OS Software Products Collection	CD	SK3T-4270
z/OS and Software Products DVD Collection	DVD	SK3T-4271

Accessibility Titles Cited in This Library

Title	Order number
z/OS V1R1.0 TSO Primer	SA22-7787
z/OS V1R5.0 TSO/E User's Guide	SA22-7794
z/OS V1R5.0 ISPF User's Guide, Volume 1	SC34-4822

Index

Special characters

A

B

C

D

E

F

G

H

I

L

M

N

O

P

Q

R

S

T

U

W

X

Z

Program Number: 5655-J38

Printed in USA

SC18-7816-00

Spine information:

IMS Common Service Layer Guide and Reference Version 9