OS/390 eNetwork Communications Server

CSM Guide

Version 2 Release 5

SC31-8575-00

OS/390 eNetwork Communications Server

CSM Guide

Version 2 Release 5

SC31-8575-00

Note:

Before using this information and the product it supports, be sure to read the general information under Appendix D, “Notices” on page 97.

First Edition (March 1998)

This edition applies to OS/390 V2R5 (program number 5647-A01).

Publications are not stocked at the address given below. If you want more IBM publications, ask your IBM representative or write to the IBM branch office serving your locality.

A form for your comments is provided at the back of this document. If the form has been removed, you may address comments to:

IBM Corporation
Department CGMD
P.O. Box 12195
Research Triangle Park, North Carolina 27709
U.S.A.

If you prefer to send comments electronically, use one of the following methods:

Fax (USA and Canada):	1-800-227-5088
Internet e-mail:	usib2hpd@vnet.ibm.com
World Wide Web:	http://www.s390.ibm.com/os390
IBMLink:	CIBMORCF at RALVM13
IBM Mail Exchange:	USIB2HPD at IBMMAIL

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

Contents

Figures

Tables

About This Book

This manual is intended to help customers write application programs that use the communications storage manager (CSM).

Who Should Use This Book

This book is for system programmers who code authorized application programs for a System/390* host. This audience can include programmers who are modifying existing programs or writing new ones. The manual can also be of use to planners who are estimating the amount of work required to use the application programming interface (API) for CSM.

You should be familiar with programming concepts described in *OS/390 eNetwork Communications Server: SNA Programming* before you use the macroinstructions described in this book. You should also be familiar with programming VTAM* applications as described in *OS/390 eNetwork Communications Server: SNA Programming*.

Knowledge of the following concepts is recommended:

- Systems network architecture
- Data communications
- LU 6.2 architecture

How to Use This Book

This book describes the CSM application program interface (API). It describes the IVTCSM macro and the programming techniques for using this macro. It is intended to be used by application programs using the HPDT interface. See the *OS/390 eNetwork Communications Server: SNA Programmers LU 6.2 Guide*for information about how VTAM application programs use the HPDT interface.

How This Book Is Organized

- Chapter 1, “Introduction to the Communications Storage Manager” on page 1 contains information about CSM definition, operation, and diagnosis. It also describes the responsibilities of applications that use the CSM API and some application design considerations.
- Chapter 2, “Communications Storage Manager Macroinstructions” on page 21 is a reference section for CSM macroinstructions. Each CSM macroinstruction is listed in alphabetical order by request type. Each macroinstruction description includes the purpose and usage, syntax (or format) of the macroinstruction, a description of all input and output parameters, and return and reason codes.

The appendixes listed below describe return codes and DSECTs of the CSM API.

- Appendix A, “Return and Reason Codes for the IVTCSM Macroinstruction” on page 87
- Appendix B, “CSM DSECTs” on page 89

Note: In this manual, examples of macroinstructions have blanks between macroinstruction operands. This convention is used to improve the formatting of the manual. When coding the macroinstructions, you must delete the blanks.

Typographic Conventions Used in This Book

This publication uses the following typographic conventions:

- Commands that you enter verbatim onto the command line are presented in **bold**.
- Variable information and parameters that you enter within commands, such as filenames, are presented in *italic*.
- System responses are presented in `monospace`.

Where to Find More Information

"Bibliography" on page 99, describes the books in the eNetwork Communications Server for OS/390 Version 2 Release 5 library, arranged according to task. The bibliography also lists the titles and order numbers of books related to this book, or cited by name in this book.

Where to Find Related Information on the Internet

You may find the following information helpful.

Note: Any pointers in this publication to websites are provided for convenience only and do not in any manner serve as an endorsement of these websites.

You can read more about VTAM, TCP/IP, OS/390, and IBM on these Web pages:

Home Page	**Uniform Resource Locator (URL)**
VTAM	http://www.networking.ibm.com/vta/vtaprod.html
TCP/IP	http://www.networking.ibm.com/tcm/tcmprod.html
OS/390	http://www.s390.ibm.com/os390/
IBM eNetwork Communications Server	http://www.software.ibm.com/enetwork/commserver.html
IBM	http://www.ibm.com/

For definitions of the terms and abbreviations used in our books, you can view or download the latest *IBM Networking Softcopy Glossary* at the following URL:

http://www.networking.ibm.com/nsg/nsgmain.htm

How to Contact IBM

For telephone assistance in problem diagnosis and resolution (in the United States or Puerto Rico), call the IBM Software Support Center anytime (1-800-237-5511). You will receive a return call within 8 business hours (Monday – Friday, 8:00 a.m – 5:00 p.m., local customer time).

Outside of the United States or Puerto Rico, contact your local IBM representative or your authorized IBM supplier.

Summary of Changes

Summary of Changes
for SC31-8575-00

This is the first edition of this book which contains information previously presented in *VTAM Programming for CSM Version 4 Release 4 for MVS/ESA* (SC31-8420). This book is new for eNetwork Communication Server for OS/390 Release 2 Version 5, which provides OpenEdition function for VTAM in the OS/390 environment.

Chapter 1. Introduction to the Communications Storage Manager

What is CSM?

The communications storage manager (CSM) is a component of VTAM that allows authorized host applications to share data with VTAM and other CSM users without having to physically copy the data. A CSM user can be any system-authorized application program or product that resides on an S/390* host running, at a minimum, Version 4 Release 3 of MVS/ESA.*

Why Use CSM Services?

CSM is provided as part of the HPDT family of services. HPDT optimizes system performance for the transfer of bulk data. By providing a means for authorized applications to share buffers, CSM improves system performance during the transfer of bulk data by reducing the processing required for data movement. As a result, CPU resources (CPU cycles, memory bus, and cache) are conserved.

Application Use of CSM

CSM includes an application programming interface (API) that allows users to obtain and return CSM buffers, change ownership of buffers, copy buffers and perform other functions related to CSM buffer management. Applications must be authorized to use CSM. The storage key for CSM buffers is key 6, fetch protected. Users set up and access data that resides in CSM buffers. These buffers are obtained from buffer pools that are identified by their buffer size and storage type according to Table 1.

Table 1. Buffer Pools in CSM

Storage types	Buffer Sizes				
Data space	**4 KB**	**16 KB**	**32 KB**	**60 KB**	**180KB**
ECSA	**4 KB**	**16 KB**	**32 KB**	**60 KB**	**180KB**

Data space storage is a common area data space and is associated with the master scheduler address space. This association results in a data space that persists for the life of the system.

When an application obtains buffers from CSM, that application is considered the owner of those buffers. Based on application specifications, CSM can associate buffer responsibility with an address space or a task within an address space. Applications using CSM have the responsibility of returning owned buffers so that storage is available for other users.

Ownership can be transferred to another user. In this case, the new owner is responsible for the return of the buffers. CSM manages buffer reclamation during termination at the task or address space level based on ownership. For detailed information about buffer reclamation during termination, see “CSM Recovery for Normal and Abnormal Termination” on page 19.

For more information about buffer ownership, see “Ownership of Buffers” on page 10.

Installing, Defining, and Initializing CSM

CSM is shipped and installed with the VTAM product tape. However, many CSM functions are independent of VTAM. CSM storage limits and tuning parameters are defined in the CSM parmlib member, IVTPRM00. See the *OS/390 eNetwork Communications Server: SNA Planning and Migration Guide* for information about the CSM parmlib member.

CSM is initialized by the first request to create a pool of buffers and remains active for the life of the system, independent of VTAM's status. The CREATE_POOL request could be issued by VTAM or a host application. Upon initialization, CSM reads the CSM parmlib member to determine storage limits and buffer pool related values. Once CSM is initialized, it persists for the life of the system.

Monitoring CSM

The use of CSM storage can be monitored by system operators by issuing the DISPLAY CSM command. CSM messages always start with the message prefix IVT. For a complete list of messages issued by CSM, see *OS/390 eNetwork Communications Server: SNA Messages*.

The following information is provided by the DISPLAY CSM command.

- Amount of storage allocated to each pool
- Amount of storage allocated to each user of the pool
- Cumulative storage allocated to each user across all pools
- Names of CSM data spaces

The DISPLAY CSM command can be used to identify a user of the pool that is consuming inordinate amounts of storage. This could occur in situations where an application fails to free buffers that it obtained from CSM. The report of storage allocated to a user is based on the user's *owner_ID* (OWNERID operand on the DISPLAY CSM command). CSM uses the application's address space identifier (ASID) as the OWNERID.

In some circumstances, the sum of the total of the storage allocated to all users of a pool may be greater than the total amount of storage allocated to a pool. This is due to multiple owners of a buffer resulting from the creation of shared instances using the IVTCSM REQUEST=ASSIGN_BUFFER macroinstruction. The information by OWNERID indicates the amount of storage that must be freed by the user to enable the storage to be returned to the buffer pool. CSM storage limits can be increased or decreased without requiring a re-IPL by issuing the MODIFY CSM command.

Some messages are issued by CSM when CSM storage limits are either at a critical level (97% of defined limits) or exceeded. In this case, the system operator can issue the MODIFY CSM command to increase the amount of fixed or ECSA storage available for CSM.

For more information about these two CSM commands, see *OS/390 eNetwork Communications Server: Operation* .

CSM storage information is also provided to the performance monitor interface (PMI). This information is equivalent to the information provided for the summary

format of the DISPLAY CSM command. See *OS/390 eNetwork Communications Server: SNA Customization* for more information.

Applications using the CSM API can also request information about the status of CSM storage by issuing the IVTCSM REQUEST=RESOURCE_STATS macroinstruction.

CSM Problem Diagnosis

You can obtain trace output of application requests to CSM by using the CSM option on the VTAM internal trace or, when VTAM is not active, the GTF trace facility.

- When VTAM is operational, the CSM trace facility is controlled using VIT. CSM writes records to the VIT using VTAM trace interfaces. The CSM trace option is used to control the generation of CSM trace records for both internal and external tracing.
- When VTAM is not operational, the VIT is not available and only external tracing is provided. The external trace is generated using the VTAM GTF event ID to write trace records directly to GTF in the same format as those recorded using VIT.

CSM tracing records the parameter list information that flows across the CSM interface and key internal events (such as pool expansion and contraction) for functions that manipulate buffer states. This allows you to trace and analyze the usage history of a buffer.

The number of trace records required to represent one IVTCSM request is variable based on the number of buffer operations requested. Since the information required to trace one IVTCSM request may span several CSM trace records, you can use the trace record flag field to determine whether additional trace records exist for a particular IVTCSM request. If the first bit in the trace record flag field is set on, then the trace record is continued. If VIT is not active, then multiple trace records for an IVTCSM request could be interspersed with trace records of IVTCSM requests from other users. A unique trace record number is provided to correlate the continuation trace records for each IVTCSM request.

For more information about tracing events over the CSM API, see *OS/390 eNetwork Communications Server: SNA Diagnosis* .

Formatting CSM Dump Information

IPCS dump formatters provide the following services for displaying CSM information in a dump:

- Find and display CSM data structures
- Find and display CSM data structures for a buffer pool based on the size and source, ECSA or data space
- Search pool extents
- Find and display a buffer based on the input buffer token
- Find all buffers based on an input ownerid

All information is displayed with a header identifying the contents All information is displayed with a header identifying the contents followed by hexadecimal contents only — no field identification is provided. Formatting options that display information in buffers can provide only the requested data when the required storage areas are included in the dump.

Application Responsibilities for Using CSM

An application must be authorized to use the CSM API as described in *OS/390 eNetwork Communications Server: SNA Programming*. As system-authorized applications, all CSM users are expected to be written so that they handle CSM storage in a responsible manner. Therefore, the application design should adhere to the following guidelines for requesting CSM services. Applications using CSM for VTAM's high performance data transfer (HPDT) service should refer to the guidelines described in the *OS/390 eNetwork Communications Server: SNA Programmers LU 6.2 Guide* .

- Data in CSM storage should be modified only by the *original requester* of the buffers. The original requester is considered to be the application that obtained the storage using the IVTCSM REQUEST=GET_BUFFER macroinstruction. All other applications are considered to be *borrowers* of the buffers and must treat the data as read-only. There are possible exceptions to this rule. See “Responsibilities of Buffer Ownership” on page 11 for more details.
- All programs directly referencing CSM storage must do so in the proper storage key. All CSM storage is allocated in key 6.

  The IVTCSM REQUEST=COPY_DATA macroinstruction allows data to be copied into or out of CSM storage. The authorized invoker can be in any key. Use of this service may reduce the impact to the application due to storage key mismatches when CSM storage must be accessed.
- An application must not reference or use CSM storage after passing ownership of that storage to another user.
- An application that has accepted ownership of CSM storage is obligated to return the storage to CSM unless that application is passing the storage to another user. See “Ownership of Buffers” on page 10 for more information. Storage is returned to CSM on the IVTCSM REQUEST=FREE_BUFFER macroinstruction.
- Applications should use the IVTCSM REQUEST=RESOURCE_STATS macroinstruction to monitor the status of CSM storage. The application must be capable of reacting to storage constraint conditions that might jeopardize the application's or system's operation. See “Obtaining CSM Resource Statistics” on page 16 for more information. The RESOURCE_STATS request is described on page 83.
- In general, applications should request buffers from data space instead of ECSA to ensure that more virtual storage is available to all users of CSM. ECSA should be used only if there are special application requirements.
- The application's use of CSM should be documented so that the installation can adjust ECSA and fixed storage limits in the CSM parmlib member as necessary. This information should be available in application installation documentation so that necessary changes to the limits can be made prior to application installation.

Functions Provided by the CSM API

This section describes the functions that a user of CSM needs to be able to create and use storage pools. The IVTCSM macroinstruction provides the interface for applications issuing requests to CSM. The following shows the types of requests that can be issued using the IVTCSM macroinstruction:

- IVTCSM REQUEST=ASSIGN_BUFFER
- IVTCSM REQUEST=CHANGE_OWNER
- IVTCSM REQUEST=COPY_DATA
- IVTCSM REQUEST=CREATE_POOL
- IVTCSM REQUEST=DELETE_POOL
- IVTCSM REQUEST=DUMP_INFO
- IVTCSM REQUEST=FIX_BUFFER
- IVTCSM REQUEST=FREE_BUFFER
- IVTCSM REQUEST=GET_BUFFER
- IVTCSM REQUEST=PAGE_BUFFER
- IVTCSM REQUEST=RESOURCE_STATS

See Chapter 2, “Communications Storage Manager Macroinstructions” on page 21 for the complete description and syntax of each request. Table 2 on page 8 contains a cross reference of IVTCSM requests and their valid input and output parameters.

Table 2. Valid Operands for IVTCSM Macroinstruction

REQUEST	CREATE POOL	DELETE POOL	GET BUFFER	FREE BUFFER	ASSIGN BUFFER	CHANGE OWNER	FIX BUFFER	PAGE BUFFER	COPY DATA	DUMP INFO	RESOURCE STATS
PLISTVER	Oi	Oi	Oi	Oi	Oi	Oi	Oi	Oi	Oi	Oi	Oi
BUFLIST			Ri	Ri	Ri	Ri	Ri	Ri			
BUFNUM			Ri	Ri	Ri	Ri	Ri	Ri			
BUFSIZE	Ri										
BUFSOURC	Ri										
BUFTYPE			Ri		Oi			Ri			
CLEAR			Oi	Oi							
DS_INFO	Oo									Oo	
ERRBFLST			Oo	Oo	Oo	Oo	Oo	Oo			
EXPBUF	Ri										
FREERTN			Oi								
FREETO				Oi							
GAP			Oi	Oi	Oi	Oi	Oi	Oi			
INITBUF	Ri										
MF	Ri	Ri	Ri	Ri	Ri	Ri	Ri	Ri	Ri	Ri	Ri
MINFREE	Ri										
OWNERID			Oi		Oi	Oi					
PAD									Oi		
POOLTOKN		Ri	Ri								
PADCHAR									Oi		
RETCODE	Oo	Oo	Oo	Oo	Oo	Oo	Oo	Oo	Oo	Oo	Oo
RETPTOKN	Oo										
RSNCODE	Oo	Oo	Oo	Oo	Oo	Oo	Oo	Oo	Oo	Oo	Oo
SKIPBUF				Oi		Oi					
SRCERRL									Oo		
SRCGAP									Oi		
SRCLIST									Ri		
SRCNUM									Ri		
STATAREA	Oo										Oo
TARGERRL									Oo		
TARGGAP									Oi		
TARGLIST									Ri		
TARGNUM									Ri		
TASKID			Oi		Oi	Oi					
THREAD			Oi	Oi	Oi	Oi	Oi	Oi	Oi		
UTILRTN			Oi	Oi	Oi	Oi	Oi	Oi	Oi		
WAIT			Oi				Oi				

Key: **R**=Required **O**=Optional **i**=input **o**=output

Buffer Pool Creation and Registration

Upon application request, CSM can create buffer pools a based on the size and types listed in Table 1 on page 3. Altogether, a total of 10 CSM buffer pools can be created, one for each storage type and buffer size.

The structures to maintain the storage pools are created as a result of the first IVTCSM REQUEST=CREATE_POOL macroinstruction issued by a user of CSM (this could be VTAM or a host application). The pool may or may not already exist. In either case, the application that issues the CREATE_POOL request is a registered user and can obtain buffers from that pool. CSM returns a token in the RETPTOKN parameter that the application uses on subsequent requests for buffers from that pool.

An application can specify buffer pool tuning parameters on the CREATE_POOL request. See “CSM Buffer Pool Expansion and Contraction” on page 17 for more information.

Requesting Storage from CSM

Before an application can retrieve storage from CSM, it must be registered as a user of a CSM buffer pool as described in “Buffer Pool Creation and Registration” on page 8. The CSM buffer pools available are summarized in Table 1 on page 3. CSM returns a pool token in the RETPTOKN parameter that the application uses on subsequent requests for buffers from that pool.

Applications obtain buffers from CSM by specifying the token of the pool with which they are registered and the number of buffers on the IVTCSM REQUEST=GET_BUFFER macroinstruction. CSM allocates the requested number of buffers from that pool to the application. CSM returns a buffer list. Each entry in the buffer list contains a token and other information that the application and CSM use to reference the buffers. See “CSM Buffer Lists” for more information about buffer lists in CSM.

An application can obtain buffers from CSM as they are needed or manage its own pool of buffers to be retained for multiple uses. By default, buffers are returned to CSM when the current owner issues the IVTCSM REQUEST=FREE_BUFFER macroinstruction. An application designed to manage its own pool of buffers uses a buffer return exit routine that assumes control of the buffers once they are freed by a user.

See “Buffer Return Exit Routine” on page 18 for more information about the buffer return exit routine that can receive control after buffers are released by a FREE_BUFFER request.

GET_BUFFER requests that exceed the storage available in CSM are rejected. Users of CSM should monitor CSM storage use and take action to prevent critical shortages that might jeopardize the application's or system's operation. See “Obtaining CSM Resource Statistics” on page 16 for more information.

CSM Buffer Lists

The following requests require the application to provide the address of a buffer list:

- IVTCSM REQUEST=ASSIGN_BUFFER
- IVTCSM REQUEST=CHANGE_OWNER
- IVTCSM REQUEST=FIX_BUFFER
- IVTCSM REQUEST=FREE_BUFFER
- IVTCSM REQUEST=GET_BUFFER
- IVTCSM REQUEST=PAGE_BUFFER

CSM builds a buffer list in the area provided on the BUFLIST parameter of the GET_BUFFER request. The buffer list contains, among other information, a token used by CSM to manage each buffer. The format of the CSM buffer list is described in “CSM Buffer List Entry (IVTBUFL)” on page 89. The IVTBUFL DSECT maps an entry in the CSM buffer list, which can be indicated by the BUFLIST, SRCLIST or TARGLIST parameters on the IVTCSM macroinstruction.

Each buffer list entry may be contiguous to the previous entry with the number of entries in the list defined by the BUFNUM operand on the request. The GAP parameter may be used to separate entries.

Fixed Buffers Versus Pageable Buffers

An application can specify buffers that are in one of the following formats:

- Guaranteed to be fixed (BUFTYPE=FIXED)
- Guaranteed to be pageable (BUFTYPE=PAGEABLE)
- Eligible to be made pageable (BUFTYPE=PAGEELIG)

Guaranteeing That a Buffer Is Fixed

Some processes require buffers to be in a fixed format. An application can specify fixed buffers (BUFTYPE=FIXED parameter) on the GET_BUFFER and ASSIGN_BUFFER requests. Availability of fixed buffers is limited by the system definitions in the installation's CSM parmlib member. The application can obtain pageable buffers when fixed buffers are unavailable and make the buffers fixed at a later time using the FIX_BUFFER request.

Making a Buffer Eligible to Be Paged

An application can classify buffers as *eligible to be paged* by specifying BUFTYPE=PAGEELIG on the GET_BUFFER, ASSIGN_BUFFER, and PAGE_BUFFER requests. Eligible to be paged is a status maintained by CSM. The actual system state of a buffer with this status can be either fixed or pageable. CSM internally monitors the level of fixed storage usage. The buffers may remain fixed unless CSM determines that the storage should be made pageable. This avoids the overhead of unnecessary fixing and freeing of storage from a system perspective.

This function may be used to avoid consuming fixed storage for data that is being held in a buffer for possible use at a later time. If an eligible to be paged buffer must later be fixed, the application must issue the FIX_BUFFER request.

Making a Buffer Guaranteed to Be Pageable

An application can classify buffers as *guaranteed to be made pageable* by specifying BUFTYPE=PAGEABLE on the GET_BUFFER and PAGE_BUFFER requests. This function can be issued only for a buffer consisting of one image. For information about how multiple images of a buffer can be created, see “IVTCSM REQUEST=ASSIGN_BUFFER” on page 26.

Ownership of Buffers

CSM uses the address space identifier (ASID) as the basis for the OWNERID value. OWNERID can be specified on the following requests:

- IVTCSM REQUEST=ASSIGN_BUFFER
- IVTCSM REQUEST=GET_BUFFER
- IVTCSM REQUEST=CHANGE_OWNER

The owner is the application responsible for returning buffers to CSM using the IVTCSM REQUEST=FREE_BUFFER macroinstruction. CSM assigns initial buffer ownership to the original requester of the buffers. This can be overridden by specifying another user's ASID as the OWNERID on the IVTCSM

REQUEST=GET_BUFFER macroinstruction. Ownership of CSM buffers can be passed to another user by issuing an IVTCSM REQUEST=CHANGE_OWNER macroinstruction.

Ownership of a buffer can be associated to a task by specifying a TCB address on the TASKID parameter on the following requests:

- IVTCSM REQUEST=ASSIGN_BUFFER
- IVTCSM REQUEST=GET_BUFFER
- IVTCSM REQUEST=CHANGE_OWNER

If TASKID is not specified, buffer ownership is associated with the ASID.

Ownership of a buffer that has an associated buffer return exit is not actually changed due to an IVTCSM REQUEST=CHANGE_OWNER macroinstruction; the buffer is actually borrowed. The original owner of the buffer will be maintained so that ownership is restored when the buffer is freed. However, if the original owner's address space terminates before the buffer return exit is invoked, the current borrower, if one exists, becomes the new owner. If the buffer is not being borrowed, it is returned to CSM.

Responsibilities of Buffer Ownership

When an application obtains buffers from CSM on a IVTCSM REQUEST=GET_BUFFER macroinstruction, that application is considered to be the *original requester* of that storage. Ownership responsibility entails either ultimately freeing the storage (on an IVTCSM REQUEST=FREE_BUFFER macroinstruction) or changing ownership to another user. Failure to return the storage ultimately creates CSM storage constraint conditions.

The original requester of the storage can specify a buffer return exit routine at storage allocation time and is entitled to the return of storage without modification. Therefore, the receiving application should consider this as *read-only* storage and should not modify the contents.

It is possible that applications, if written as a cooperative set of processes, could determine that it is acceptable to modify the data if the original application does not require the original data returned unmodified. The caution here is that applications written in this manner must be able to guarantee that the original requester of the storage is one of the cooperative applications. If the storage allocation source is unknown to the receiver, the read-only requirement applies.

Changing Ownership of a Buffer

The CHANGE_OWNER request can be issued by any user that can address the buffers by using the buffer tokens provided by CSM. This includes the following scenarios:

- Application A passes buffer tokens to application B. Application B issues the IVTCSM REQUEST=CHANGE_OWNER macroinstruction.
- Application A uses the IVTCSM REQUEST=CHANGE_OWNER macroinstruction to pass ownership of the buffers to application B.

After an ownership change, the former owner of the buffers must not address the buffers except when the former owner has specified a buffer return exit. In this case, the application must not address the buffers until its exit routine is scheduled

by CSM. The exit routine is scheduled when the current owner of the buffers issues the IVTCSM REQUEST=FREE_BUFFER macroinstruction.

High Performance Data Transfer — An Example of How Ownership is Changed: The HPDT interface demonstrates how two CSM users perform ownership change. On an HPDT send, the application passes the buffer tokens to VTAM in an extended buffer list (XBUFLST) on the send request. VTAM performs the CHANGE_OWNER request. If an error occurs, VTAM notifies the application so that the application can recover buffers that were not accepted. On the receive side, VTAM passes the buffer list to the application and issues the CHANGE_OWNER request.

Storage Return

An application uses the IVTCSM REQUEST=FREE_BUFFER macroinstruction to release buffers back to CSM. CSM returns the buffers back to the original requester if all of the following are true:

- The original requester specified a buffer return exit address on the FREERTN parameter of the IVTCSM REQUEST=GET_BUFFER macroinstruction.
- The original requester is active at the time the IVTCSM REQUEST=FREE_BUFFER macroinstruction is issued.
- The application issuing IVTCSM REQUEST=FREE_BUFFER does not specify FREETO=CSM. This parameter is intended for situations where the original requester needs to return buffers without invoking its own buffer return exit.

The requester may optionally specify that the buffer is to be cleared when issuing the FREE_BUFFER request. See “Clearing Data from Buffers” for more information.

All storage manager GET_BUFFER and ASSIGN_BUFFER requests must have a corresponding FREE_BUFFER request before the buffer is considered available for reallocation by CSM or before a buffer return exit is invoked for a buffer obtained specifying a user free routine. This is necessary to ensure that all users have finished using the buffer.

Clearing Data from Buffers

An application can instruct CSM to clear data from buffers that are returned to the buffer pool by specifying CLEAR=YES on the following macroinstructions:

- IVTCSM REQUEST=FREE_BUFFER
- IVTCSM REQUEST=GET_BUFFER

This provides for secure data to be passed to another user such that any residual data is eliminated when that buffer has returned to the pool.

Notes:

1. The CLEAR=YES specification on a IVTCSM REQUEST=GET_BUFFER macroinstruction overrides a CLEAR=NO specification on a IVTCSM REQUEST=FREE_BUFFER macroinstruction.
2. Specifying CLEAR=YES will not cause a buffer to be cleared that is returned to an application's buffer return exit routine. However, if CLEAR=YES is specified, the buffer is cleared in the event that it is returned to the storage pool.

Removing Registration from a Pool

Since each pool may have multiple users, a storage pool is not deleted until all buffers have been returned by all users and DELETE_POOL requests have been received for each corresponding CREATE_POOL request. To deregister as the user of the pool, the application issues the IVTCSM REQUEST=DELETE_POOL macroinstruction.

Copying Data to or from a CSM Buffer

Applications can use the IVTCSM REQUEST=COPY_DATA macroinstruction to copy data to or from a CSM buffer or a user data area. The authorized invoker can be in any key. Use of this request may reduce the impact to the application due to storage key mismatches when CSM storage must be accessed. It also assist users of CSM data space buffers by isolating the application from the addressing method used to access a data space.

The IVTCSM REQUEST=COPY_DATA macroinstruction allows multiple source buffers to be copied to or from one or multiple target buffers. The source buffers are copied to the target buffers using the source and target buffer lengths to pack data or span data across the target buffers as required.

If the cumulative length of the source buffers is greater than the cumulative length of the target buffers, truncation of the source data occurs. The application can specify a character on the PADCHAR input parameter to pad the target buffers when the cumulative length of the source buffers is less than the cumulative length of the target buffers.

The application must supply a source buffer list and a target buffer list on the COPY_DATA request. (See Figure 1 on page 14.) The number of entries in each list are not required to be equal. Within each list, entries may or may not represent a CSM buffer. The value of the BUFL_SOURCE field dictates whether the entry represents a CSM buffer. For entries representing CSM buffers, the address that is the source or target of the copy is provided by the requester and is not required to be the actual start address of the CSM buffer. CSM validates that the specified address and length corresponds to a storage area that is within the bounds of the CSM buffer. This validation is based on the size of the buffer as determined at the time the buffer pool was created.

A user data area that is involved in the copy data operation may be optionally ALET-qualified to allow this area to reside in a data space.

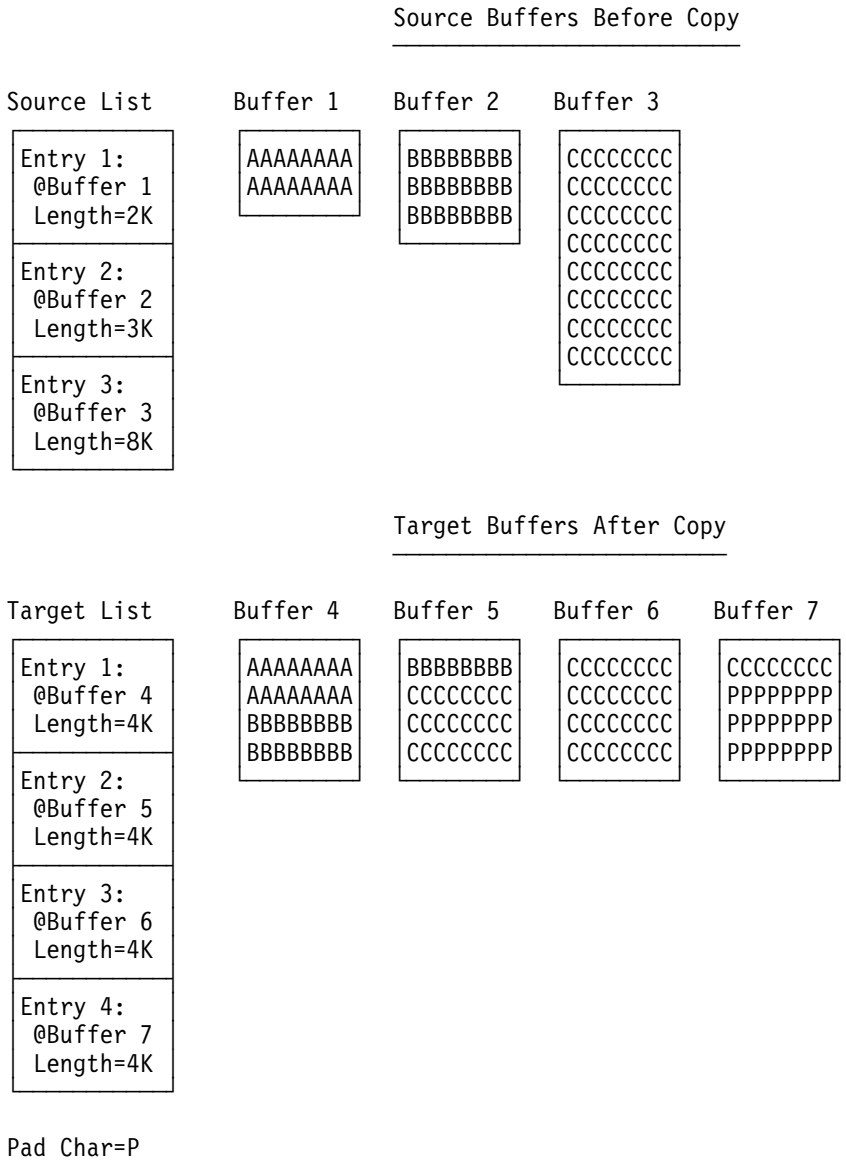


Figure 1. Example of Copy Data Buffer List and Copy Results

Sharing Buffers Among Multiple Users

The IVTCSM REQUEST=ASSIGN_BUFFER macroinstruction provides the capability for a buffer to be concurrently shared between multiple users. A logical instance of the buffer is created for each user. A new physical copy of the buffer is not created. This function is provided to allow specific areas of the buffer to be allocated to different owners. This function could be used to allow multiple users to have read access to the same data. No serialization is provided to prevent concurrent updates by users.

A new buffer token is returned representing the new instance of the buffer. The buffer token is the means by which this new instance of the buffer is known to

CSM. This token must be used with all other requests to CSM for the associated buffer instance.

On the ASSIGN_BUFFER request, multiple shared instances of a single buffer can be created by passing a multiple entry buffer list with the same buffer token in each entry.

A request to create a new image of a buffer that is in a guaranteed to be pageable state is not permitted. The reason for this restriction is to guarantee that a user of a buffer that has multiple images can successfully issue a FIX_BUFFER request if necessary. Fixing a buffer requires that the entire buffer be fixed regardless of the fact that the user may only be interested in a piece of the buffer. The application can specify the BUFTYPE parameter as described in “Fixed Buffers Versus Pageable Buffers” on page 10.

Obtaining CSM Dumping Information

Including CSM storage in a dump may be necessary to debug problems associated with the application's use of CSM buffers. An application can use the following information to include CSM storage in a dump.

- Dumping area containing CSM data structures, pool structures, and buffer headers:
  - Location extended common service area (ECSA)
  - Subpool 241
  - Key 6

  These areas can be included in a user dump by specifying the subpool and key on the invocation of the SDUMPX macroinstruction.
- Dumping buffers in an ECSA buffer pool:
  - Location Extended CSA (ECSA)
  - Subpools 231
  - Key 6

  The ECSA buffer pool can be included in a user dump by specifying the subpool and key on the invocation of the SDUMPX macroinstruction.
- Dumping buffers in a data space buffer pool
  - Location common area data space (CADS) owned by the master address space
  - Key 6
  - Data space address range 4KB-2Gigabyte
  - Data space STOKENs and ALETs provided by IVTCSM REQUEST=DUMP_INFO.

It is recommended that applications include the areas containing CSM data structures by specifying the subpools and key to reduce the amount of data included in the dump. Selective dumping is most important when dumping data in a CSM data space rather than dumping the entire 2 gigabyte contents.

For ease of dumping CSM buffers during testing, an application can use ECSA buffers since this area can be included in a dump using the subpool and key. Once the application is debugged, data space buffers could be used for the production application.

An application can request the location of information required to obtain CSM data space information in a dump on the following macroinstructions:

- IVTCSM REQUEST=CREATE_POOL
- IVTCSM REQUEST=DUMP_INFO

The application can specify the address of an area on the DS_INFO parameter where CSM will place the address of the area containing the CSM data space information. This information is mapped by the IVTDATSP DSECT. (See “CSM Data Space Information (IVTDATSP)” on page 90.) The application can request the information during initialization processing and use the information throughout normal processing.

Dumping the entire data space requires a long processing time and a large amount of external recording media. You may wish to limit the amount of area dumped based on address ranges within the data space believed to be pertinent to the user. For example, using the ALETs returned by the IVTCSM REQUEST=DUMP_INFO macroinstruction, you can determine whether, at the time of abend, any access registers (AR) contain an ALET associated with a CSM data space. If an AR contains an ALET of a CSM data space and the program was executing in AR mode at the time of the failure, you may want to dump a limited number of bytes of the data surrounding the address represented by the general register (GR) and AR pair.

Obtaining CSM Resource Statistics

An application can request the address of information required to monitor usage of CSM resources such as ECSA, data space and fixed storage. The address of this information is returned on the STATAREA parameter when the following macroinstructions are issued:

- IVTCSM REQUEST=CREATE_POOL
- IVTCSM REQUEST=RESOURCE_STATS

The application can request the address of the resource statistics area during initialization processing and can reference this area to obtain resource statistics throughout normal processing.

For each resource type, two bits are defined. One to indicate the usage of the resource is constrained and one to indicate the usage is critical. If neither bit is set, the usage of the resource is considered to be normal.

Critical indicates that CSM storage usage is at 97% of defined limits or higher.

Constrained
indicates that CSM storage usage is approaching the critical level. If this bit is set, the application should determine if the critical bit is also set.

GET_BUFFER requests that exceed the storage available in CSM are rejected. Users of CSM should monitor CSM storage use and take action to prevent critical shortages that might jeopardize the application's or system's operation. Possible user actions might include freeing buffers that are no longer needed, selecting a different storage source for buffer pools, or limiting usage of fixed storage.

The CSM resource statistics information is mapped by the IVTSTATA DSECT. (See “CSM Resource Status Area (IVTSTATA)” on page 91.)

CSM Buffer Pool Expansion and Contraction

CSM buffer pools can be expanded or contracted based on the MINFREE, INITBUF, and EXPBUF specifications in the CSM parmlib member, IVTPRM00. If these values are not specified in the CSM parmlib member, or if CSM cannot read that parmlib member during initialization, CSM uses the values specified by the application on the IVTCSM REQUEST=CREATE_POOL macroinstruction. If buffer pool tuning specifications are not available from either the CSM parmlib member or application request, then system defaults are used. This section describes how expansion and contraction of CSM buffer pools is performed based on application settings. For more information about the CSM parmlib member, see the *OS/390 eNetwork Communications Server: SNA Planning and Migration Guide* .

Number of Buffers When Buffer Pool Is Created Using Application Settings

CSM creates a pool of buffers when the first CREATE_POOL request is received. The initial number of buffers created in that pool is specified by the INITBUF parameter. INITBUF is required on the CREATE_POOL request.

The range for INITBUF is 0 - 9999. If zero is specified, only the base pool structure is created. In this case, the pool will be expanded on the first GET_BUFFER request based on the EXPBUF parameter value. If a value is specified that is outside of the range for INITBUF, one of the following values is used, depending on the size of the buffers in the pool.

Pool size	Initial number of buffers (INITBUF)
4K	64
16K	32
32K	16
60K	16
180K	2

CSM Buffer Pool Expansion Using Application Settings

CSM buffer pools are expanded as needed to maintain the specified number of free buffers in the pool. The number of free buffers maintained is determined by the highest MINFREE (minimum free buffer) specifications by all users of a pool.

CSM buffer pools are expanded by the maximum of the EXPBUF (expand buffers) specifications by all users of a pool. The storage pool will be expanded if the number of free buffers falls below MINFREE.

The pool is managed in extents. Expansion consists of creating a new extent with the number of buffers as determined by the EXPBUF parameters. The expansion of the pool is scheduled as a side process in order to avoid excessive pathlength on a IVTCSM REQUEST=GET_BUFFER macroinstruction due to in-line pool expansion. In the event that pool expansion must complete to satisfy a request for buffers, the application has the option of waiting for pool expansion to complete by specifying WAIT=EXPAND or WAIT=YES on the IVTCSM REQUEST=GET_BUFFER macroinstruction.

MINFREE Parameter: MINFREE is required on the CREATE_POOL request. The range for MINFREE is 0–9999. If a value is specified that is outside of the range for MINFREE, one of the following values is used, depending on the size of the buffers in the pool.

Pool size	Minimum buffers that are free (MINFREE)
4K	8
16K	4
32K	2
60K	2
180K	1

EXPBUF Parameter: EXPBUF is required on the CREATE_POOL request. The valid range for EXPBUF depends on the size of the buffers in the pool. If a value is specified that is outside of the range for EXPBUF, one of the following values is used.

Pool size	Valid range	Number of buffers to expand pool (EXPBUF)
4K	1-256	16
16K	1-256	8
32K	1-128	4
60K	1-68	4
180K	1-22	2

CSM Buffer Pool Contraction Using Application Settings

CSM buffer pools contract as necessary to prevent the pools from consuming system resources permanently as the result of usage peaks. Contraction occurs when the number of buffers that are not in use exceeds one of the following values, whichever is higher:

- INITBUF
- MINFREE + 2(EXPBUF)

The pool will not contract below the level specified on the INITBUF parameter.

Buffer Return Exit Routine

An application can manage its own pool of buffers to be retained for multiple uses by coding a buffer return exit routine that assumes control of the buffers once they are freed by a user. By default, buffers are returned to CSM when the current owner issues the IVTCSM REQUEST=FREE_BUFFER macroinstruction. An application that provides the address of its buffer return exit on the FREERTN parameter of the IVTCSM REQUEST=GET_BUFFER macroinstruction will receive ownership when the buffers are freed by another user.

The buffer return exit routine is scheduled to execute in the address space which owns the buffer to ensure that the owning environment still exists.

The buffer return exit routine is called by a CSM routine that receives control from the SRB scheduler. The CSM routine passes the address of the parameter list to the buffer return exit routine in register 1. To map the passed parameter list, the

buffer return exit routine should include a DSECT that issues the LIST form of the IVTFREE macro as coded in the following example:

```
name    DSECT
        IVTFREE MF=(L,listaddr)
```

The parameter list contains the address of the buffer list and the number of buffer entries in the list. The following is a sample output of the LIST form of the IVTFREE macroinstruction with a *listaddr* value of FREPL:

```
         FREPL    DS    0D                ++ IVTFREE PARM LIST
         FREPL_XVERSION DS XL1            ++ INPUT XVERSION
         FREPL_XRSVFREE1 DS CL03          ++ RESERVED  XRSVFREE1
         FREPL_XBUFLIST DS A              ++   XBUFLIST
         FREPL_XBUFNUM DS F               ++   XBUFNUM
0000C    FREPLL   EQU   *-FREPL           ++ LENGTH OF PLIST
```

Each buffer list entry is mapped by the IVTBUFL DSECT. The application can examine the tokens in each entry to correlate them with the buffers referenced by the original GET_BUFFER request. When a buffer return exit is driven, the pageability of the buffer may have changed. The original buffer address, token, and length remains the same.

After regaining ownership of the buffers, the application can determine whether to release the buffers back to CSM using the FREE_BUFFER request. To prevent looping, the application must not specify FREETO=USER on the the FREE_BUFFER request, which would reschedule the application's buffer return exit. To return the buffers back to CSM, the buffer return exit should specify FREETO=CSM on the FREE_BUFFER request.

CSM Recovery for Normal and Abnormal Termination

For normal or abnormal termination, CSM users free all owned buffers prior to completing termination processing. However, this may not always be possible for abnormal termination.

In order to ensure that buffers are not lost due to normal or abnormal termination, CSM uses the following resource termination managers to reclaim buffers that are owned by the terminating environment.

- Job Step Task Resource Termination Manager

  Job Step Task clean-up is performed if task is not an MVS started task (for example, batch).
- Memory Resource Termination Manager

  Memory clean-up is performed whenever address space memory termination occurs.

The resource managers will receive control from recovery termination management (RTM) during job step task and memory termination processing, determine if any of the allocated buffers are owned by the terminating address space and as appropriate free the buffers back to the buffer pool.

Additionally, CSM allows the user to associate buffers to a task. Buffers that are task associated will be reclaimed by CSM at the end of the specified task only if the task abnormally terminates. The user of the buffer is responsible for ensuring the buffers are freed during normal termination.

If a user of CSM desires to provide recovery mechanisms to free buffers at events other than those provided by CSM, the user can create ESTAE/FRR recovery routines to recover at a program level or RESMGR to create a resource manager to recover at a task termination event not provided by CSM. When performing this type of processing, the user must ensure that their application processing is properly synchronized with any applications to which they are passing buffers.

Chapter 2. Communications Storage Manager Macroinstructions

About This Chapter

This chapter describes all varieties of the IVTCSM macroinstruction. Separate descriptions are included for each value of the REQUEST parameter. Macroinstruction descriptions are arranged alphabetically.

How the Macroinstructions Are Described

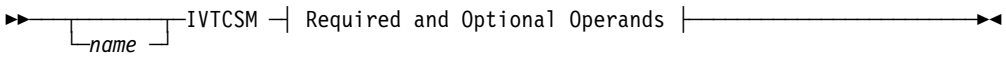
Each macroinstruction description begins with:

- Purpose of the macroinstruction
- General comments about the use of the macroinstruction
- The format or syntax of the macroinstruction and all parameters
- A description of each parameter
- Return and reason codes that can be returned for the macroinstruction.

Syntax Descriptions

The syntax for each macroinstruction is described using the following format:

main syntax diagram



If you are not familiar with this type of syntax diagram, see Appendix C, “How to Read the Syntax Diagrams” on page 93.

The macroinstructions are coded in the same format as assembler instructions, using name, operation, and operand fields. Use the *IBM High Level Assembler Language Reference for MVS and VM* for complete information on coding guidelines.

Operand Descriptions

The name and description of each operand follows the syntax diagram. Each operand description begins with an explanation of the operand's function.

Parameter values that are shown in upper case bold type must be coded as they appear in the syntax. For parameter values that are shown in lower case italic type, specify a location that is to be the source of input data or the target of output data. The location must be defined in a manner that is consistent with the indicated data type. If you wish to pass the storage address in general purpose register 2-12, code a valid expression for the register within parentheses. Otherwise, code an expression that is valid as a storage operand on RS-type instructions.

Exception: If a register is specified for RETCODE or RSNCODE, the output is loaded straight into the register.

To reference the parameter list within your program, use the list form of the macro, `MF=(L,...)`, to define the parameter list structure. The *listaddr* value you specify becomes the name by which you can reference the structure in your program. For example, the assembler instruction LA 1,*listaddr* puts the parameter list address in register 1. The name of the field associated with a parmlist operand *opername* is *listaddr*_X*opername*. If macro-defined values are associated with *opername*, the constant for a particular value, *valuenam*, is *listaddr*_X*opername*_*valuenam*.

Note: The PLISTVER operand is an exception to the rule. To reference its field, substitute *opername* with the keyword **VERSION**. Also, the macro does not define constants for any of the allowable PLISTVER values.

Completion Information

All of the executable macroinstructions pass return codes in registers, and most indicate status information in various control block fields when they are posted complete. For all macroinstructions that invoke CSM, the application can examine return codes in register 15 and reason codes in register 0. Descriptions of this status information can be found at the end of the macroinstruction description.

Computing Environment for the CSM Application Programming Interface

This section describes the environment in which the IVTCSM macro is issued.

Environment

The requirements for the caller are as follows:

Minimum authorization:	Supervisor state. Any PSW key
Dispatchable unit mode:	Task or SRB Exception: CREATE_POOL and DELETE_POOL requests must be issued in Task Mode.
Cross memory mode:	Any PASN, any HASN, any SASN CREATE_POOL and DELETE_POOL requests are PASN=HASN=SASN only.
AMODE:	31-bit
ASC mode:	Primary
Interrupt status:	Enabled for I/O and external interrupts
Locks:	No locks may be held.
Control parameters:	Control parameters must be in the primary address space.

Programming Requirements

The user must provide a recovery environment if one is necessary during the invocation of the IVTCSM Service, as the service does not provide a recovery environment during all its functions.

The service does provide for buffer reclamation at end-of-memory, end-of-Job-Step-Task, and, optionally, at abnormal end-of-task. See “CSM Recovery for Normal and Abnormal Termination” on page 19 for more information.

Restrictions

This section describes the restrictions on the caller.

- Do not use MVS page-fix services directly for buffers provided by this service. Establish the BUFTYPE attribute of these buffers using CSM service requests.
- Do not issue ALESERV delete for an ALET returned from CSM.

Input Register Information

Before issuing this macro, the caller must ensure that register 13 contains the address of a 72-byte standard save area in the primary address space.

Output Register Information

When control returns to the caller, the general purpose registers contain:

Register	Contents
0	Error reason code from the requested function
1	Used as work register by the system
2-13	Unchanged
14	Used as work register by the system
15	Return code from requested function

IVTCSM REQUEST=ASSIGN_BUFFER

Purpose

This macroinstruction allows an application to request that a buffer be logically assigned to another owner (shared) in order to make multiple owners of a buffer.

Usage

This macroinstruction allows a buffer to be concurrently shared between multiple users. A logical instance of the buffer is created for each user. A new physical copy of the buffer is not created. This macroinstruction can be used to allow specific areas of the buffer to be allocated to different owners and to allow multiple users to have read access to the same data. No serialization is provided to prevent concurrent updates by users.

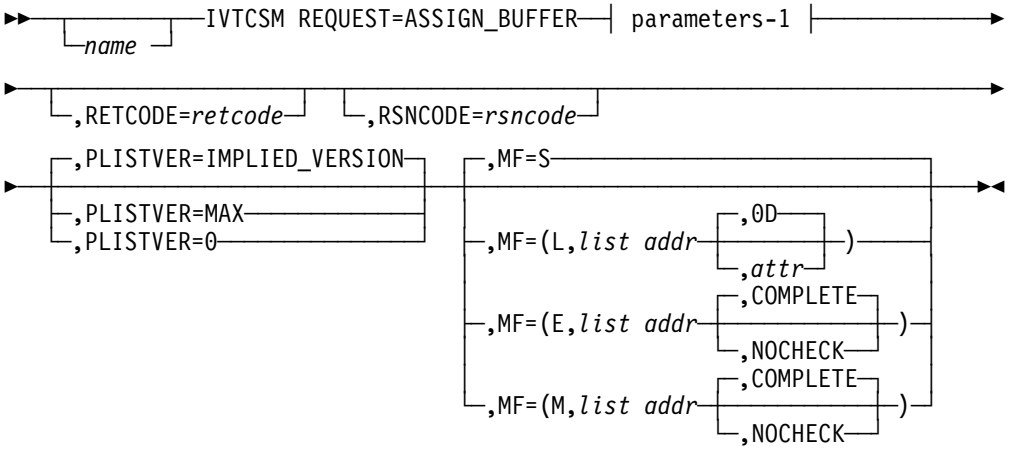
The ownership of the new instance of the buffer is assigned to the requesting application's ASID by default. Ownership of a new instance of the buffer may be optionally qualified by specifying a TASKID on the macroinstruction. The TASKID is a TCB address with the default being no task association.

On completion of this macroinstruction, a new buffer token is returned representing the new instance of the buffer. The buffer token is the means by which this new instance of the buffer is known to CSM. This token must be used with all other requests to CSM for the associated buffer instance.

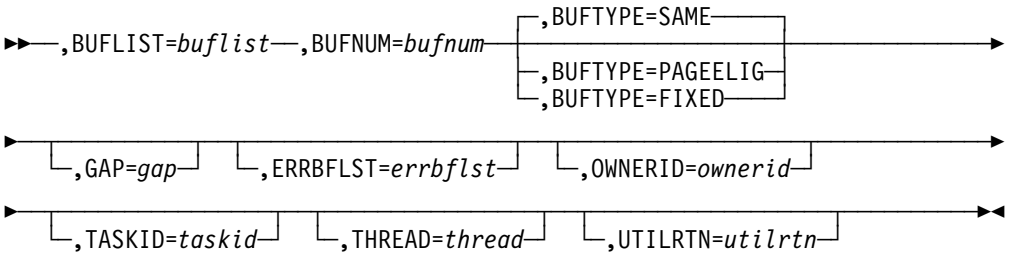
Multiple shared instances of a single buffer can be created by passing a multiple entry buffer list with the same buffer token in each entry.

Syntax

main diagram



parameters-1



Parameters

The parameters are explained as follows:

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

,BUFLIST=*buflist*
A required input parameter, of an area containing a list of buffer entries. The number of entries in the list is provided by BUFNUM. An entry in the buffer list is mapped by IVTBUFL. Some of the fields defined in IVTBUFL are required as input and some are set by CSM as output fields. Note that the buffer token representing the new buffer image is returned in the BUFL_TOKEN field as output.

The following fields in IVTBUFL are required as input for this request.

- BUFL_VERSION
- BUFL_TOKEN

The following fields in IVTBUFL are returned as output by CSM for this request.

- BUFL_TYPE
- BUFL_TOKEN

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

,BUFNUM=*bufnum*
A required input parameter, specifying the number of buffers to be logically assigned.

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

,BUFTYPE=SAME

,BUFTYPE=PAGEELIG

,BUFTYPE=FIXED
An optional parameter, specifying whether the buffer images are guaranteed to be fixed, eligible to be made pageable or have the same pageable state as the buffers represented by the input token. The default is BUFTYPE=SAME.

,BUFTYPE=SAME
Indicates that the pageable state of the buffer images will be the same as the buffers represented by the input token.

,BUFTYPE=PAGEELIG
Indicates that the buffer images are eligible to be made pageable.

,BUFTYPE=FIXED
Indicates that buffer images are guaranteed to be fixed.

,ERRBFLST=*errbflst*

An optional output parameter, specifying the number of the last buffer entry that was successfully processed when an error is detected during processing of the macroinstruction.

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

,GAP=*gap*

An optional input parameter, specifying the number of bytes used to separate buffer entries. This parameter allows the buffer entries to be in discontiguous storage. If GAP is not specified, buffer entries are contiguous.

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

,MF=S

,MF=(L,*list addr***)**

,MF=(L,*list addr,attr***)**

,MF=(L,*list addr***,0D)**

,MF=(E,*list addr***)**

,MF=(E,*list addr***,COMPLETE)**

,MF=(E,*list addr***,NOCHECK)**

,MF=(M,*list addr***)**

,MF=(M,*list addr***,COMPLETE)**

,MF=(M,*list addr***,NOCHECK)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage

area defined by the list form, and generates the macro invocation to transfer control to the service.

Use MF=M together with the list and execute forms of the macro for service routines that need to provide different options according to user-provided input. Use the list form to define a storage area; use the modify form to set the appropriate options; then use the execute form to call the service.

IBM recommends that you use the modify and execute forms of IVTCSM in the following order:

1. Use IVTCSM ...MF=(M,list-addr,COMPLETE) specifying appropriate parameters, including all required ones.
2. Use IVTCSM ...MF=(M,list-addr,NOCHECK), specifying the parameters that you want to change.
3. Use IVTCSM ...MF=(E,list-addr,NOCHECK), to execute the macro.

,*list addr*
The name of a storage area to contain the parameters. For MF=S, MF=E, and MF=M, this can be an RS-type address or an address in register (1)-(12).

,*attr*
An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

,COMPLETE
Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

,NOCHECK
Specifies that the system is not to check for required parameters and is not to supply defaults for omitted optional parameters.

,OWNERID=*ownerid*
An optional input parameter, specifying the owner to which the buffer image is to be logically assigned. If not coded, the ASID of the issuing application is assigned as the OWNERID.

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

,PLISTVER=IMPLIED_VERSION

,PLISTVER=MAX

,PLISTVER=0
An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

  If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form; in this way, MAX ensures that the parameter list does not overwrite nearby storage.
- **0**, if you use the currently available parameters.

To code: Specify one of the following:

- IMPLIED_VERSION
- MAX
- A decimal value of 0

,RETCODE=*retcode*
An optional output parameter into which the return code is to be copied from GPR 15.

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

,RSNCODE=*rsncode*
An optional output parameter into which the reason code is to be copied from GPR 0.

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

,TASKID=*taskid*
An optional input parameter that is to contain the address of a TCB. This further qualifies the ownership of a buffer to a specific task. If TASKID is not specified, the buffer is not associated with a task but is instead associated with the issuing application's ASID.

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

,THREAD=*thread*
An optional input parameter, specifying a unique identifier that is placed in the CSM trace entry to correlate trace records with the application that is requesting the buffers. It is the CSM user's responsibility to ensure that this value is different from the THREAD value specified by other users of the CSM. One way this can be achieved is by specifying an ECSA control block for THREAD.

To code: Specify the RS-type address, or address in register (2)-(12), of a 4-character field.

,UTILRTN=*utilrtn*
An optional input parameter that is issued from a utility routine. Specify the utility routine caller's address to be placed in the CSM trace entry. If this parameter is omitted, only the address of the CSM request issuer is placed in the CSM trace entry. This parameter is only relevant to the tracing process. It

should be specified only if the CSM user requires identification of the caller of a utility routine in the CSM trace entry.

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

Return Codes

The following codes can be returned to the application on this macroinstruction.

Return Code	Meaning
0	Request completed successfully.
4	Request did not complete successfully. See the following reason codes to determine the type of error encountered.

Reason Code	Meaning
2	Requested function not supported at the present time, service has not been initialized.
7	Invalid buffer token specified
8	Instance ID in the input buffer token does not match that of the buffer, possible attempt to use a buffer that has been freed.
9	Real storage unavailable to provide a fixed buffer, wait not requested.
15	Assign buffer request failed because the state of the buffer is guaranteed to be pageable.
20	BUFTYPE value specified is not valid for this request.

Return Code	Meaning
8	System error while processing the request. See the following reason codes to determine the type of error encountered.

Reason Code	Meaning
1	Unable to obtain storage for request.
6	An abend occurred while processing this request.

IVTCSM REQUEST=CHANGE_OWNER

Purpose

This macroinstruction allows the application to change the ownership of a buffer to another user.

Usage

This macroinstruction can be used to assume ownership of another user's buffers or to pass ownership to another user. The new owner must have sufficient information to address the buffers. This information can be found in the CSM buffer list. The owner of a set of buffers bears the responsibility for returning them to CSM on the IVTCSM REQUEST=FREE_BUFFER macroinstruction.

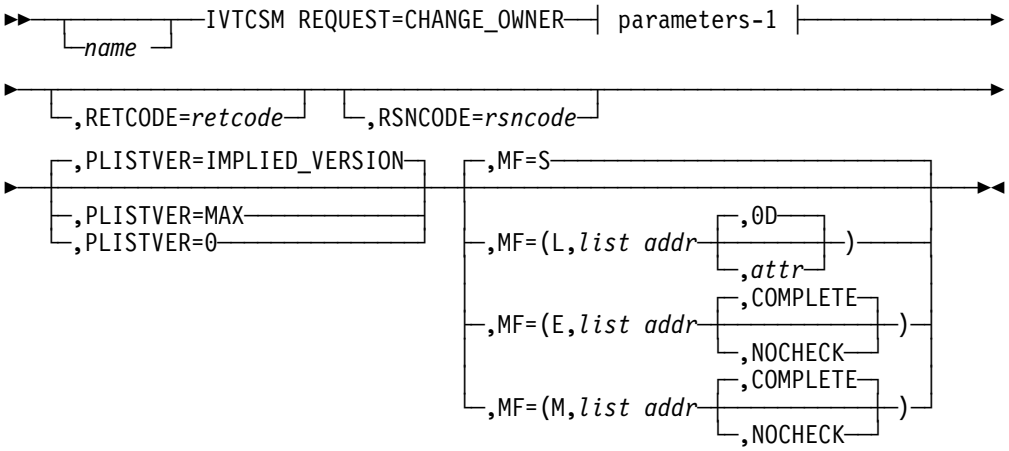
CSM associates a set of buffers that have been retrieved from a buffer pool with the OWNERID of an application. The OWNERID is the same as the application's ASID. Ownership of a buffer may be optionally qualified by specifying the TASKID parameter on the macroinstruction. The TASKID is a TCB address with the default being no task association.

Ownership of a buffer that has an associated buffer return exit is not actually changed due to an IVTCSM REQUEST=CHANGE_OWNER macroinstruction; the buffer is actually borrowed. The original owner of the buffer will be maintained so that ownership is restored when the buffer is freed. However, if the original owner's address space terminates before the buffer return exit is invoked, then buffers are returned to CSM.

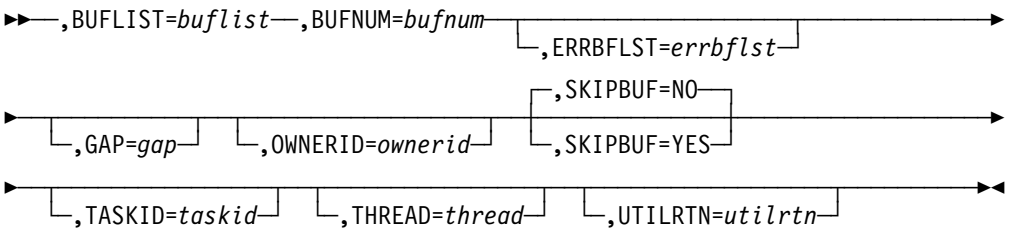
See “Ownership of Buffers” on page 10 for more information.

Syntax

main diagram



parameters-1



Parameters

The parameters are explained as follows:

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

,BUFLIST=*buflist*
A required input parameter, of an area containing a list of buffer entries. The number of entries in the list is provided by BUFNUM. Each entry in the buffer list is mapped by IVTBUFL.

The following fields in IVTBUFL are required as input for this request.

- BUFL_VERSION
- BUFL_SOURCE (Note: This field is only required when SKIPBUF=YES is specified.)
- BUFL_TOKEN

There are no fields in IVTBUFL, that are returned as output by CSM for this request.

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

,BUFNUM=*bufnum*
A required input parameter, specifying the number of buffers to change ownership.

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

,ERRBFLST=*errbflst*
An optional output parameter, containing the number of the last buffer entry that was successfully processed when an error is detected during processing of the macroinstruction.

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

,GAP=*gap*
An optional input parameter, specifying the number of bytes used to separate buffer entries. This parameter allows the buffer entries to be in discontiguous storage. If GAP is not specified, buffer entries are contiguous.

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

,MF=S

,MF=(L,*list addr***)**

,MF=(L,*list addr,attr***)**

,MF=(L,*list addr***,0D)**

,MF=(E,*list addr***)**

,MF=(E,*list addr***,COMPLETE)**

,MF=(E,*list addr***,NOCHECK)**

,MF=(M,*list addr***)**

,MF=(M,*list addr***,COMPLETE)**

,MF=(M,*list addr***,NOCHECK)**
An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

Use MF=M together with the list and execute forms of the macro for service routines that need to provide different options according to user-provided input. Use the list form to define a storage area; use the modify form to set the appropriate options; then use the execute form to call the service.

IBM recommends that you use the modify and execute forms of IVTCSM in the following order:

1. Use IVTCSM ...MF=(M,list-addr,COMPLETE) specifying appropriate parameters, including all required ones.
2. Use IVTCSM ...MF=(M,list-addr,NOCHECK), specifying the parameters that you want to change.
3. Use IVTCSM ...MF=(E,list-addr,NOCHECK), to execute the macro.

,*list addr*
The name of a storage area to contain the parameters. For MF=S, MF=E, and MF=M, this can be an RS-type address or an address in register (1)-(12).

,*attr*
An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

,COMPLETE
Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

,NOCHECK
Specifies that the system is not to check for required parameters and is not to supply defaults for omitted optional parameters.

,OWNERID=*ownerid*
An optional input parameter, specifying the owner to which the buffer is to be assigned. If not coded, the ASID of the issuing application is assigned as the OWNERID.

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

,PLISTVER=IMPLIED_VERSION

,PLISTVER=MAX

,PLISTVER=0
An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

  If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form; in this way, MAX ensures that the parameter list does not overwrite nearby storage.
- **0**, if you use the currently available parameters.

To code: Specify one of the following:

- IMPLIED_VERSION
- MAX
- A decimal value of 0

,RETCODE=*retcode*
An optional output parameter into which the return code is to be copied from GPR 15.

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

,RSNCODE=*rsncode*
An optional output parameter into which the reason code is to be copied from GPR 0.

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

,SKIPBUF=NO

,SKIPBUF=YES
An optional parameter, specifying whether all entries in the buffer list should be processed. The default is SKIPBUF=NO.

,SKIPBUF=NO
specifies that all the entries in the buffer list will be processed. No entries are skipped. The BUFL_SOURCE value is not examined.

,SKIPBUF=YES
specifies that the only entries in the buffer list that have a BUFL_SOURCE value indicating the user's non-CSM storage (BUFL_UDSPACE or BUFL_USTOR) will be skipped.

,TASKID=*taskid*
An optional input parameter that is to contain the address of a TCB. This further qualifies the ownership of a buffer to a specific task. If TASKID is not specified, the buffer is not associated with a task but is instead associated with the issuing application's ASID. id="xCMCH25">**To code:** Specify the RS-type address, or address in register (2)-(12), of a pointer field.

,THREAD=*thread*
An optional input parameter, specifying a unique identifier that is placed in the CSM trace entry to correlate trace records with the application that is requesting the buffers. It is the CSM user's responsibility to ensure that this value is different from the THREAD value specified by other users of the CSM. One way this can be achieved is by specifying an ECSA control block for THREAD.

To code: Specify the RS-type address, or address in register (2)-(12), of a 4-character field.

,UTILRTN=*utilrtn*
An optional input parameter that is issued from a utility routine. Specify the utility routine caller's address to be placed in the CSM trace entry. If this parameter is omitted, only the address of the CSM request issuer is placed in the CSM trace entry. This parameter is only relevant to the tracing process. It should be specified only if the CSM user requires identification of the caller of a utility routine in the CSM trace entry.

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

Return Codes

The following codes can be returned to the application on this macroinstruction.

Return Code	Meaning
0	Request completed successfully.
4	Request did not complete successfully. See the following reason codes to determine the type of error encountered.

Reason Code	Meaning
2	Requested function not supported at the present time, service has not been initialized.
7	Invalid buffer token specified.
8	Instance ID in the input buffer token does not match that of the buffer, possible attempt to use a buffer that has been freed.
24	ASID specified on the OWNERID parameter is not active.

IVTCSM REQUEST=COPY_DATA

Purpose

This macroinstruction allows you to copy data to or from a CSM buffer or a user data area.

Usage

This macroinstruction assists the application by isolating it from possible storage key differences between that of the requester and that of the CSM buffer. It also assists users of CSM data space buffers by isolating the requester from the addressing method used to access a data space.

This macroinstruction allows multiple source buffers to be copied to or from one or multiple target buffers. The source buffers are copied to the target buffers using the source and target buffer lengths to pack data or span data across the target buffers as required.

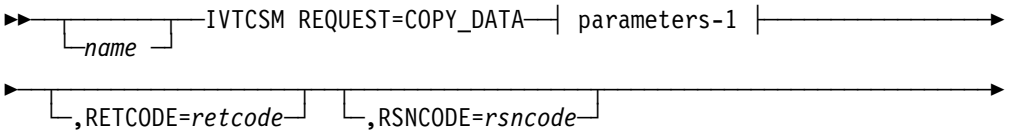
If the cumulative length of the source buffers is greater than the cumulative length of the target buffers, truncation of the source data will occur. The application can specify a character on the PADCHAR input parameter to pad the target buffers when the cumulative length of the source buffers is less than the cumulative length of the target buffers.

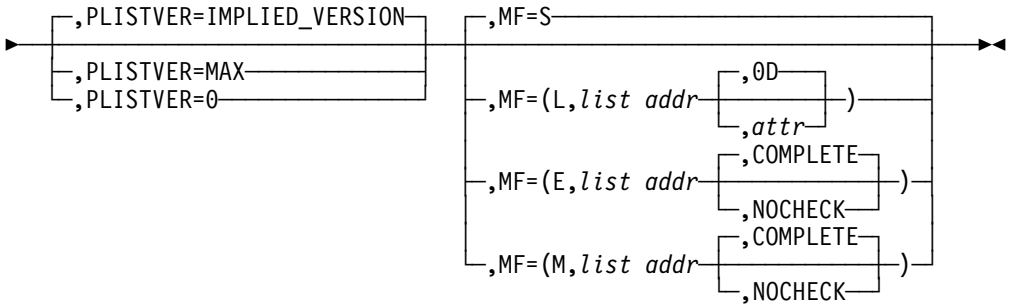
CSM accepts a source buffer list and a target buffer list as input. This is the same buffer list that is mapped by the IVTBUFL DSECT that is described in topic 89. The number of entries in each list are not required to be equal. Within each list, entries may or may not represent a CSM buffer. The BUFL_SOURCE field in the entry indicates whether the entry represents a CSM buffer. For entries representing CSM buffers, the address that is the source or target of the copy is supplied by the requester and is not required to be the actual start address of the CSM buffer. CSM validates that the specified address and length corresponds to a storage area that is within the bounds of the CSM buffer. This validation is based on the size of the buffer as determined at the time the buffer pool was created.

The application can use the COPY_DATA request to copy data to or from a non-CSM data space using the ALET provided in the buffer list entry. The ALET must be valid for the address space for which it is being used.

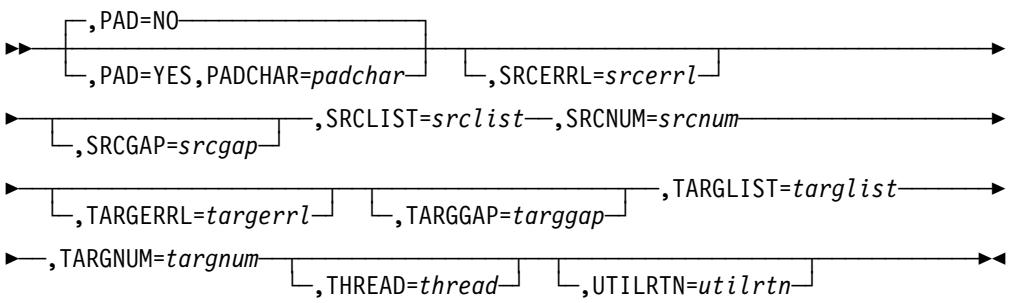
Syntax

main diagram





parameters-1



Parameters

The parameters are explained as follows:

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

,MF=S

,MF=(L,*list addr***)**

,MF=(L,*list addr,attr***)**

,MF=(L,*list addr***,0D)**

,MF=(E,*list addr***)**

,MF=(E,*list addr***,COMPLETE)**

,MF=(E,*list addr***,NOCHECK)**

,MF=(M,*list addr***)**

,MF=(M,*list addr***,COMPLETE)**

,MF=(M,*list addr***,NOCHECK)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

Use MF=M together with the list and execute forms of the macro for service routines that need to provide different options according to user-provided input. Use the list form to define a storage area; use the modify form to set the appropriate options; then use the execute form to call the service.

IBM recommends that you use the modify and execute forms of IVTCSM in the following order:

1. Use IVTCSM ...MF=(M,list-addr,COMPLETE) specifying appropriate parameters, including all required ones.
2. Use IVTCSM ...MF=(M,list-addr,NOCHECK), specifying the parameters that you want to change.
3. Use IVTCSM ...MF=(E,list-addr,NOCHECK), to execute the macro.

,*list addr*

The name of a storage area to contain the parameters. For MF=S, MF=E, and MF=M, this can be an RS-type address or an address in register (1)-(12).

,*attr*

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

,COMPLETE

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

,NOCHECK

Specifies that the system is not to check for required parameters and is not to supply defaults for omitted optional parameters.

,PAD=NO

,PAD=YES

An optional parameter that indicates if padding is to be performed. The default is PAD=NO.

,PAD=NO
indicates that padding is not performed.

,PAD=YES
indicates that padding is to be performed using the value specified by PADCHAR.

,PADCHAR=*padchar*
When PAD=YES is specified, a required input parameter, specifying the character to use as pad if the cumulative target length is greater than the cumulative source length. If PAD=YES is not specified, then no padding is performed.

To code: Specify the RS-type address, or address in register (2)-(12), of a 1-character field.

,PLISTVER=IMPLIED_VERSION

,PLISTVER=MAX

,PLISTVER=0
An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

  If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form; in this way, MAX ensures that the parameter list does not overwrite nearby storage.
- **0**, if you use the currently available parameters.

To code: Specify one of the following:

- IMPLIED_VERSION
- MAX
- A decimal value of 0

,RETCODE=*retcode*
An optional output parameter into which the return code is to be copied from GPR 15.

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

,RSNCODE=*rsncode*
An optional output parameter into which the reason code is to be copied from GPR 0.

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

,SRCERRL=*srcerrl*
An optional output parameter, specifying the number of the last buffer entry that was successfully processed in the SRCLIST.

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

,SRCGAP=*srcgap*
An optional input parameter, specifying the number of bytes used to separate buffer entries in SRCLIST. This parameter allows the buffer entries to be in discontiguous storage. If this parameter is not specified, buffer entries will be in contiguous storage.

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

,SRCLIST=*srclist*
A required input parameter, of an area containing a list of information about the buffers from which the data is to be copied. Each entry in the list describes a buffer and is mapped by IVTBUFL. The number of entries is equal to the number of buffers specified by SRCNUM. The buffer entry may represent a CSM buffer or a user data area.

The following fields in IVTBUFL are required as input for this request.

- BUFL_VERSION
- BUFL_SOURCE
- BUFL_TOKEN (Note: This field is only required if data is being copied from a CSM buffer.)
- BUFL_ALET (Note: This field is only required to access the data in a user data space.)
- BUFL_ADDR
- BUFL_SIZE

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

,SRCNUM=*srcnum*
A required input parameter, specifying the number of source buffers for the copy.

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

,TARGERRL=*targerrl*
An optional output parameter, specifying the number of the last buffer entry that was successfully processed in the TARGLIST. id="xCMCO37">**To code:** Specify the RS-type address, or address in register (2)-(12), of a fullword field.

,TARGGAP=*targgap*
An optional input parameter, specifying the number of bytes used to separate buffer entries in TARGLIST. This parameter allows the buffer entries to be in discontiguous storage. If this parameter is not specified, buffer entries will be in contiguous storage.

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

,TARGLIST=*targlist*

A required input parameter, of an area containing a list of information about the buffers that are the target of the copy operation. Each entry in the list is a buffer entry mapped by IVTBUFL. The buffer entry may represent a CSM buffer or a user data area.

The following fields in IVTBUFL are required as input for this request.

- BUFL_VERSION
- BUFL_SOURCE
- BUFL_TOKEN (Note: This field is only required if data is being copied into a CSM buffer.)
- BUFL_ALET (Note: This field is only required to copy data into a user data space.)
- BUFL_ADDR
- BUFL_SIZE

There are no fields in IVTBUFL returned as output, by CSM, for this request.

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

,TARGNUM=*targnum*

A required input parameter, specifying the number of target buffers for the copy.

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

,THREAD=*thread*

An optional input parameter, specifying a unique identifier that is placed in the CSM trace entry to correlate trace records with the application that is requesting the buffers. It is the CSM user's responsibility to ensure that this value is different from the THREAD value specified by other users of the CSM. One way this can be achieved is by specifying an ECSA control block for THREAD.

To code: Specify the RS-type address, or address in register (2)-(12), of a 4-character field.

,UTILRTN=*utilrtn*

An optional input parameter that is issued from a utility routine. Specify the utility routine caller's address to be placed in the CSM trace entry. If this parameter is omitted, only the address of the CSM request issuer is placed in the CSM trace entry. This parameter is only relevant to the tracing process. It should be specified only if the CSM user requires identification of the caller of a utility routine in the CSM trace entry.

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

Return Codes

The following codes can be returned to the application on this macroinstruction.

Return Code	Meaning
0	Request completed successfully.

4 Request did not complete successfully. See the following reason codes to determine the type of error encountered.

Reason Code	Meaning
2	Requested function not supported at the present time, service has not been initialized.
7	Invalid buffer token specified.
8	Instance ID in the input buffer token does not match that of the buffer, possible attempt to use a buffer that has been freed.
12	Address and length specified on a copy data request for a source buffer descriptor is outside the bounds of the CSM buffer represented by the specified pool token.
13	Address and length specified on a copy data request for a target buffer descriptor is outside the bounds of the CSM buffer represented by the specified pool token.
14	Copy operation resulted in truncation of source data due to insufficient buffer space provided by the target buffer list.
18	BUFL_SOURCE value is not valid for an entry in the Source buffer list (SRCLIST).
19	BUFL_SOURCE value is not valid for an entry in the Target buffer list (TRGLIST).
20	BUFTYPE value specified is not valid for this request.
21	BUFSOURC value specified is not valid for this request.
22	Source and target buffers overlap, no data has been copied.

IVTCSM REQUEST=CREATE_POOL

Purpose

This macroinstruction allows the user to register as a user of a storage pool of buffers residing in ECSA or in a data space.

The structures to maintain the storage pools are created in response to the first CREATE_POOL request by a user of CSM. For storage pools requesting a data space as the storage type, a data space is created on the first request for a pool of this type. Multiple storage pools may exist per data space.

On the create request, the caller specifies the size of the buffers in the pool to be created (4K, 16K, 32K, 60K, and 180K). Only one pool of a given size exists per storage type. Requests by other callers for a pool of the same characteristics will share the existing pool.

Usage

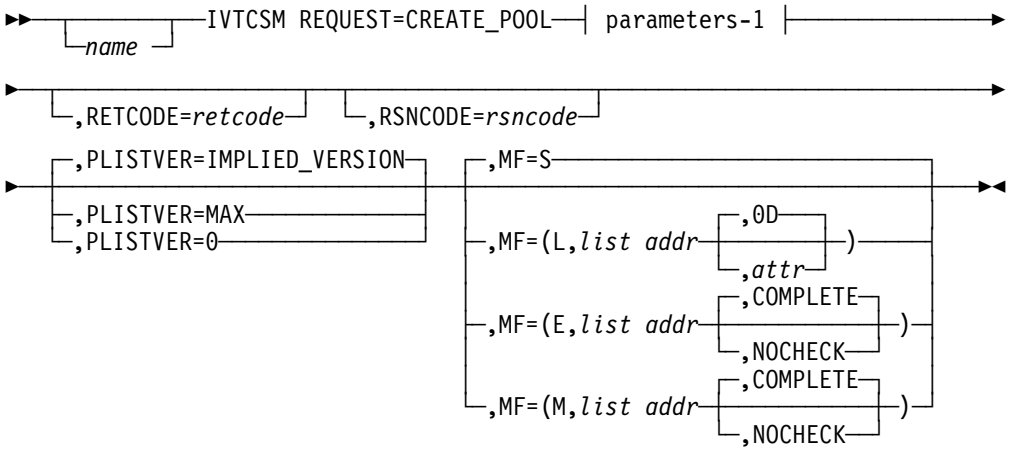
This macroinstruction should be used by an application if it subsequently will request buffers from CSM.

Environment

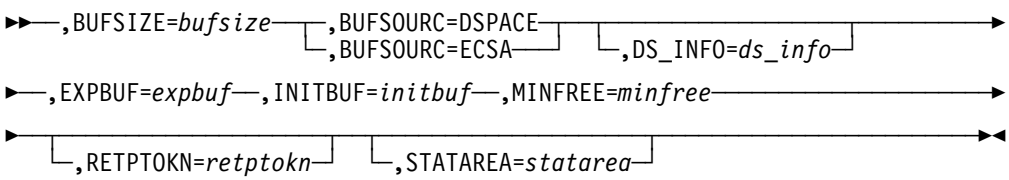
This macroinstruction must be issued in task mode; it is not allowed in cross memory mode.

Syntax

main diagram



parameters-1



Parameters

The parameters are explained as follows:

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

,BUFSIZE=*bufsize*
A required input parameter, specifying the size of the buffers in the pool to be created. Valid pool sizes are 4K, 16K, 32K, 60K and 180K. All other values specified on this parameter are rounded up to the next valid pool size. However, if BUFSIZE is greater than 180K, the CREATE_POOL request is rejected.

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

,BUFSOURC=DSPACE

,BUFSOURC=ECSA
A required parameter, specifying the source of the storage from which the buffers are to be allocated.

,BUFSOURC=DSPACE
indicates that the storage pool is to be created in data space.

,BUFSOURC=ECSA
indicates that the storage pool is to be created in ECSA.

,DS_INFO=*ds_info*
An optional output parameter that contains the address of an area containing the information required to dump CSM data spaces mapped by IVTDATSP.

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

,EXPBUF=*expbuf*
A required input parameter, specifying the number of buffers by which the pool is expanded when the number of free buffers falls below the value for MINFREE or when a GET_BUFFER request needs to be satisfied.

Valid ranges for EXPBUF are noted in the following chart. If a value outside of a range is specified, then CSM will use a default value. The default values for EXPBUF are also noted in the chart.

POOL SIZE	VALID RANGE	DEFAULT
4K	1-256	16
16K	1-256	8
32K	1-128	4
60K	1-68	4
180K	1-22	2

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

,INITBUF=*initbuf*

A required input parameter, specifying the initial number of buffers to be created in the storage pool. If zero is specified, the base pool will only be created to represent the requester as a user of the pool. In this case, the pool will be expanded on the first GET_BUFFER macroinstruction based on the specification for EXPBUF.

Note that the pool will not contract if the number of buffers currently available is not at a certain value. The value is determine as the higher of INITBUF or MINFREE+(2*EXPBUF).

Valid values for INITBUF are 0–9999. If a value outside of this range is specified, then CSM will use a default value. The default values for INITBUF are noted in the following chart.

POOL SIZE	DEFAULT
4K	64
16K	32
32K	16
60K	16
180K	2

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

,MF=S

,MF=(L,*list addr***)**

,MF=(L,*list addr,attr***)**

,MF=(L,*list addr***,0D)**

,MF=(E,*list addr***)**

,MF=(E,*list addr***,COMPLETE)**

,MF=(E,*list addr***,NOCHECK)**

,MF=(M,*list addr***)**

,MF=(M,*list addr***,COMPLETE)**

,MF=(M,*list addr***,NOCHECK)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

Use MF=M together with the list and execute forms of the macro for service routines that need to provide different options according to user-provided input. Use the list form to define a storage area; use the modify form to set the appropriate options; then use the execute form to call the service.

IBM recommends that you use the modify and execute forms of IVTCSM in the following order:

1. Use IVTCSM ...MF=(M,list-addr,COMPLETE) specifying appropriate parameters, including all required ones.
2. Use IVTCSM ...MF=(M,list-addr,NOCHECK), specifying the parameters that you want to change.
3. Use IVTCSM ...MF=(E,list-addr,NOCHECK), to execute the macro.

,*list addr*
The name of a storage area to contain the parameters. For MF=S, MF=E, and MF=M, this can be an RS-type address or an address in register (1)-(12).

,*attr*
An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

,COMPLETE
Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

,NOCHECK
Specifies that the system is not to check for required parameters and is not to supply defaults for omitted optional parameters.

,MINFREE=*minfree*
A required input parameter, specifying the minimum number of buffers to be free in the pool at any time. The storage pool will be expanded if the number of free buffers falls below this limit.

Valid values for MINFREE are 0-9999. If a value outside of this range is specified, then CSM will use a default value. The default values for MINFREE are noted in the following chart.

POOL SIZE	DEFAULT
4K	8
16K	4
32K	2
60K	2
180K	1

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

,PLISTVER=IMPLIED_VERSION

,PLISTVER=MAX

,PLISTVER=0

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

  If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form; in this way, MAX ensures that the parameter list does not overwrite nearby storage.
- **0**, if you use the currently available parameters.

To code: Specify one of the following:

- IMPLIED_VERSION
- MAX
- A decimal value of 0

,RETCODE=*retcode*

An optional output parameter into which the return code is to be copied from GPR 15.

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

,RETPTOKN=*retptokn*

An optional output parameter, of an area in which the application is to receive a token representing this user of this pool. This token must be supplied as input on the IVTCSM REQUEST=DELETE_POOL and IVTCSM REQUEST=GET_BUFFER macroinstructions, via the POOLTOKN parameter, associated with this pool

To code: Specify the RS-type address, or address in register (2)-(12), of a 10-character field.

,RSNCODE=*rsncode*
An optional output parameter into which the reason code is to be copied from GPR 0.

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

,STATAREA=*statarea*
An optional output parameter that contains the address an area containing the resource statistics mapped by IVTSTATA.

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

Return Codes

The following codes can be returned to the application on this macroinstruction.

Return Code	Meaning
0	Request completed successfully.
4	Request did not complete successfully. See the following reason codes to determine the type of error encountered.

Reason Code	Meaning
3	Specified buffer size is large than supported size.
4	Buffer pool cannot be expanded to satisfy request.
21	BUFSOURC value is not valid for this request.
23	Unable to create the specified pool. Creation of the pool would cause the ECSA maximum limit to be exceeded.

Return Code	Meaning
8	System error while processing the request. See the following reason codes to determine the type of error encountered.

Reason Code	Meaning
1	Unable to obtain storage for request.
2	Schedule SRB fail for PC routine.
3	Unable to create ALET for data space.
4	Error encountered while creating the data space.
5	Unable to create another data space. Number of data spaces exceeds the maximum.
6	An abend occurred while processing this request.

IVTCSM REQUEST=DELETE_POOL

Purpose

This macroinstruction allows the user to indicate that it is no longer a registered user of the storage pool.

Since each pool may have multiple users, a storage pool will not be deleted until all buffers have been returned by all users and delete requests have been received for each corresponding create request.

Usage

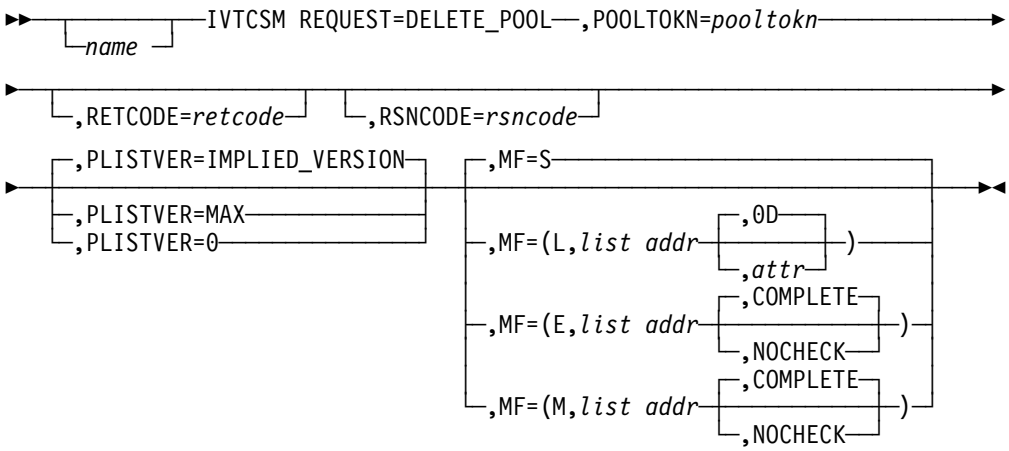
This macroinstruction should be used only if an IVTCSM REQUEST=CREATE_POOL macroinstruction was previously issued by the application.

Environment

This macroinstruction must be issued in task mode; it is not allowed in cross memory mode.

Syntax

main diagram



Parameters

The parameters are explained as follows:

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

,MF=S

,MF=(L,*list addr***)**

,MF=(L,*list addr,attr***)**

,MF=(L,*list addr***,0D)**

,MF=(E,*list addr***)**

,MF=(E,*list addr***,COMPLETE)**

,MF=(E,*list addr***,NOCHECK)**

,MF=(M,*list addr***)**

,MF=(M,*list addr***,COMPLETE)**

,MF=(M,*list addr***,NOCHECK)**
An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

Use MF=M together with the list and execute forms of the macro for service routines that need to provide different options according to user-provided input. Use the list form to define a storage area; use the modify form to set the appropriate options; then use the execute form to call the service.

IBM recommends that you use the modify and execute forms of IVTCSM in the following order:

1. Use IVTCSM ...MF=(M,list-addr,COMPLETE) specifying appropriate parameters, including all required ones.
2. Use IVTCSM ...MF=(M,list-addr,NOCHECK), specifying the parameters that you want to change.
3. Use IVTCSM ...MF=(E,list-addr,NOCHECK), to execute the macro.

,*list addr*
The name of a storage area to contain the parameters. For MF=S, MF=E, and MF=M, this can be an RS-type address or an address in register (1)-(12).

,*attr*

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

,COMPLETE

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

,NOCHECK

Specifies that the system is not to check for required parameters and is not to supply defaults for omitted optional parameters.

,PLISTVER=IMPLIED_VERSION

,PLISTVER=MAX

,PLISTVER=0

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

  If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form; in this way, MAX ensures that the parameter list does not overwrite nearby storage.
- **0**, if you use the currently available parameters.

To code: Specify one of the following:

- IMPLIED_VERSION
- MAX
- A decimal value of 0

,POOLTOKN=*pooltokn*

A required input parameter, of a token representing this user of this pool. This must be the token provided to the application on the associated IVTCSM REQUEST=CREATE_POOL macroinstruction.

To code: Specify the RS-type address, or address in register (2)-(12), of a 10-character field.

,RETCODE=*retcode*

An optional output parameter into which the return code is to be copied from GPR 15.

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

,RSNCODE=*rsncode*

An optional output parameter into which the reason code is to be copied from GPR 0.

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

Return Codes

The following codes can be returned to the application on this macroinstruction.

Return Code	Meaning
0	Request completed successfully.
4	Request did not complete successfully. See the following reason codes to determine the type of error encountered.

Reason Code	Meaning
2	Requested function not supported at the present time, service has not been initialized.
6	pool token specified not valid.

8 System error while processing the request.

Reason Codes:	Meaning
6	An abend occurred while processing this request.

IVTCSM REQUEST=DUMP_INFO

Purpose

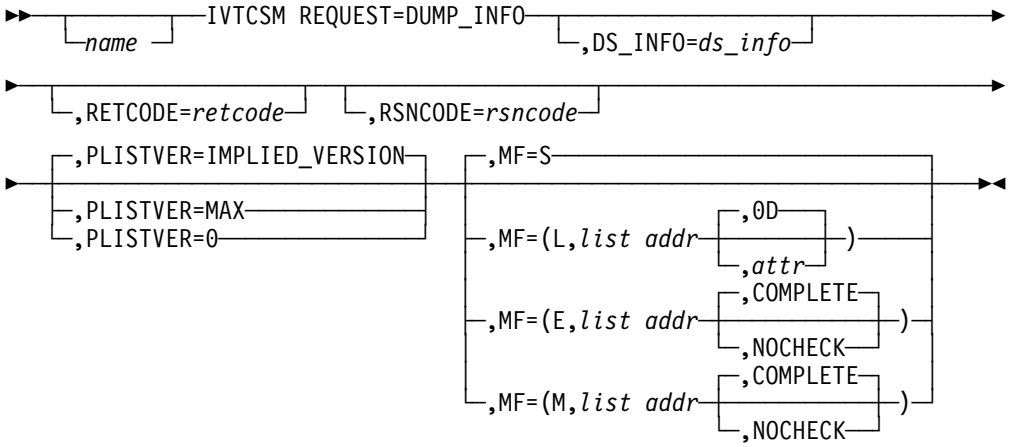
This macroinstruction requests the address of the information required to include CSM data space information in a dump.

Usage

CSM returns the address of the requested information in the address provided on the DS_INFO parameter. This information is mapped by the IVTDATSP DSECT as described on page 90.

Syntax

main diagram



Parameters

The parameters are explained as follows:

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

,DS_INFO=*ds_info*
An optional output parameter that contains the address of an area containing the information required to dump CSM data spaces mapped by IVTDATSP.

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

,MF=S

,MF=(L,*list addr***)**

,MF=(L,*list addr,attr***)**

,MF=(L,*list addr***,0D)**

,MF=(E,*list addr***)**

,MF=(E,*list addr***, COMPLETE)**

,MF=(E,*list addr***,NOCHECK)**

,MF=(M,*list addr***)**

,MF=(M,*list addr***, COMPLETE)**

,MF=(M,*list addr***,NOCHECK)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

Use MF=M together with the list and execute forms of the macro for service routines that need to provide different options according to user-provided input. Use the list form to define a storage area; use the modify form to set the appropriate options; then use the execute form to call the service.

IBM recommends that you use the modify and execute forms of IVTCSM in the following order:

1. Use IVTCSM ...MF=(M,list-addr,COMPLETE) specifying appropriate parameters, including all required ones.
2. Use IVTCSM ...MF=(M,list-addr,NOCHECK), specifying the parameters that you want to change.
3. Use IVTCSM ...MF=(E,list-addr,NOCHECK), to execute the macro.

,*list addr*

The name of a storage area to contain the parameters. For MF=S, MF=E, and MF=M, this can be an RS-type address or an address in register (1)-(12).

,*attr*

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

,COMPLETE
Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

,NOCHECK
Specifies that the system is not to check for required parameters and is not to supply defaults for omitted optional parameters.

,PLISTVER=IMPLIED_VERSION

,PLISTVER=MAX

,PLISTVER=0
An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

  If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form; in this way, MAX ensures that the parameter list does not overwrite nearby storage.
- **0**, if you use the currently available parameters.

To code: Specify one of the following:

- IMPLIED_VERSION
- MAX
- A decimal value of 0

,RETCODE=*retcode*
An optional output parameter into which the return code is to be copied from GPR 15.

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

,RSNCODE=*rsncode*
An optional output parameter into which the reason code is to be copied from GPR 0.

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

Return Codes

The following codes can be returned to the application on this macroinstruction.

Return Code	Meaning
0	Request completed successfully.
4	Request did not complete successfully. See the following reason codes to determine the type of error encountered.

Reason Code	Meaning
2	Requested function not supported at the present time, service has not been initialized.

IVTCSM REQUEST=FIX_BUFFER

Purpose

This macroinstruction allows an application to change the pageable state of a buffer to be guaranteed to be fixed.

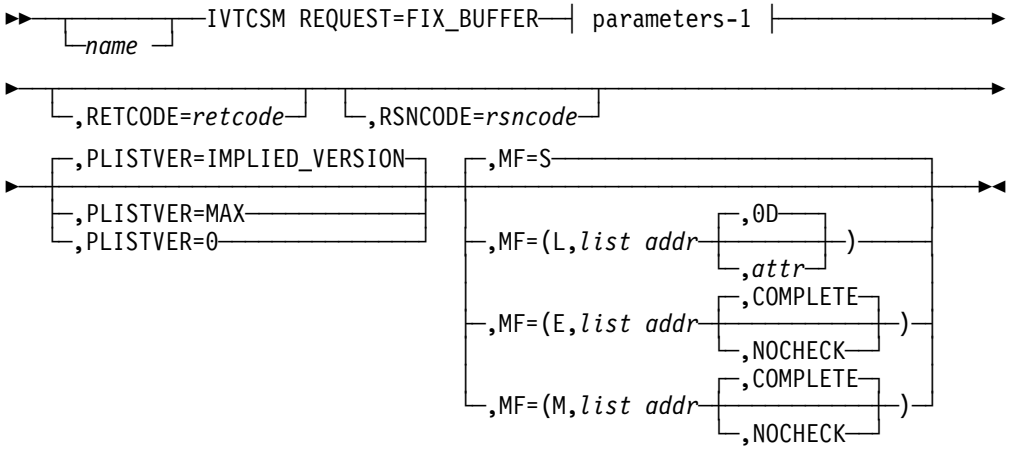
Usage

If a buffer is guaranteed to be pageable or eligible to be pagefreed, an application can use this macroinstruction to make the buffer guaranteed to be fixed.

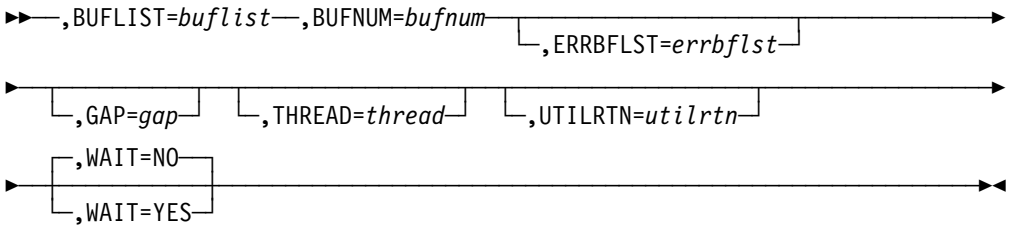
See “Fixed Buffers Versus Pageable Buffers” on page 10 for more information.

Syntax

main diagram



parameters-1



Parameters

The parameters are explained as follows:

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

,BUFLIST=*buflist*
: A required input parameter, of an area in which the application program is to provide a list of buffer entries. The number of entries in the list is equal to the value specified by the BUFNUM parameter. A entry in the list is mapped by IVTBUFL.

  The following fields in IVTBUFL are required as input for this request.

- BUFL_VERSION
- BUFL_TOKEN

The following field in IVTBUFL is returned as output by CSM for this request.

- BUFL_TYPE

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

,BUFNUM=*bufnum*

A required input parameter, specifying the number of buffers to be made guaranteed to be fixed.

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

,ERRBFLST=*errbflst*

An optional output parameter, containing the number of the last buffer entry that was successfully processed when an error is detected during processing of the macroinstruction.

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

,GAP=*gap*

An optional input parameter, specifying the number of bytes used to separate buffer entries. This parameter allows the buffer entries to be in discontiguous storage. If GAP is not specified, buffer entries are in contiguous storage.

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

,MF=S

,MF=(L,*list addr***)**

,MF=(L,*list addr,attr***)**

,MF=(L,*list addr***,0D)**

,MF=(E,*list addr***)**

,MF=(E,*list addr***,COMPLETE)**

,MF=(E,*list addr***,NOCHECK)**

,MF=(M,*list addr***)**

,MF=(M,*list addr***,COMPLETE)**

,MF=(M,*list addr***,NOCHECK)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

Use MF=M together with the list and execute forms of the macro for service routines that need to provide different options according to user-provided input. Use the list form to define a storage area; use the modify form to set the appropriate options; then use the execute form to call the service.

IBM recommends that you use the modify and execute forms of IVTCSM in the following order:

1. Use IVTCSM ...MF=(M,list-addr,COMPLETE) specifying appropriate parameters, including all required ones.
2. Use IVTCSM ...MF=(M,list-addr,NOCHECK), specifying the parameters that you want to change.
3. Use IVTCSM ...MF=(E,list-addr,NOCHECK), to execute the macro.

,*list addr*

The name of a storage area to contain the parameters. For MF=S, MF=E, and MF=M, this can be an RS-type address or an address in register (1)-(12).

,*attr*

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

,COMPLETE

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

,NOCHECK

Specifies that the system is not to check for required parameters and is not to supply defaults for omitted optional parameters.

,PLISTVER=IMPLIED_VERSION

,PLISTVER=MAX

,PLISTVER=0

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

  If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form; in this way, MAX ensures that the parameter list does not overwrite nearby storage.
- **0**, if you use the currently available parameters.

To code: Specify one of the following:

- IMPLIED_VERSION
- MAX
- A decimal value of 0

,RETCODE=*retcode*

An optional output parameter into which the return code is to be copied from GPR 15.

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

,RSNCODE=*rsncode*

An optional output parameter into which the reason code is to be copied from GPR 0.

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

,THREAD=*thread*

An optional input parameter, specifying a unique identifier that is placed in the CSM trace entry to correlate trace records with the application that is requesting the buffers. It is the CSM user's responsibility to ensure that this value is different from the THREAD value specified by other users of the CSM. One way this can be achieved is by specifying an ECSA control block for THREAD.

To code: Specify the RS-type address, or address in register (2)-(12), of a 4-character field.

,UTILRTN=*utilrtn*

An optional input parameter that is issued from a utility routine. Specify the utility routine caller's address to be placed in the CSM trace entry. If this parameter is omitted, only the address of the CSM request issuer is placed in the CSM trace entry. This parameter is only relevant to the tracing process. It should be specified only if the CSM user requires identification of the caller of a utility routine in the CSM trace entry.

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

,WAIT=NO

,WAIT=YES
An optional parameter, specifying whether or not the request should wait for fixed storage to become available. The default is WAIT=NO.

,WAIT=NO
Specifies that this macroinstruction completes without waiting for fixed storage to become available.

,WAIT=YES
Specifies that this macroinstruction will not complete until fixed storage becomes available. If fixed storage is not available, users will be suspended until enough fixed storage is available to satisfy the request.

Return Codes

The following codes can be returned to the application on this macroinstruction.

Return Code	Meaning
0	Request completed successfully.
4	Request did not complete successfully. See the following reason codes to determine the type of error encountered.

Reason Code	Meaning
2	Requested function not supported at the present time, service has not been initialized.
7	Specified buffer token is not valid.
8	Instance ID in the input buffer token does not match that of the buffer, possible attempt to use a buffer that has been freed.
9	Real storage unavailable to provide a fixed buffer, wait not requested.

8 System error while processing the request.

Reason Codes: Meaning
6 An abend occurred while processing this request.

IVTCSM REQUEST=FREE_BUFFER

Purpose

This macroinstruction allows an application to return one or more buffers to a storage pool. It is also used to logically return a buffer that has been assigned to multiple owners. The buffer is returned to CSM when the owner of the last buffer image returns it to CSM and a buffer return exit routine was not specified during the initial allocation of the buffer.

Usage

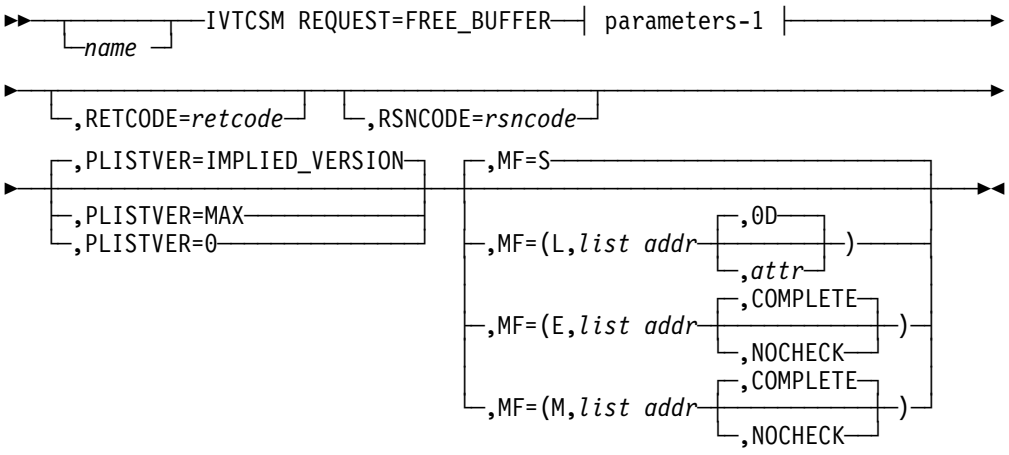
An application may specify the address of a buffer return exit routine that is to receive control when the IVTCSM REQUEST=FREE_BUFFER macroinstruction is issued. See “Buffer Return Exit Routine” on page 18 for more information. An application may optionally specify that the buffer return exit address specified when the buffer was obtained is to be overridden, allowing a buffer to be freed back to CSM that was obtained specifying a free routine address. This option is requested by specifying FREETO=CSM; it must be invoked in this manner only by the requester of the buffer that specified a free routine on the GET_BUFFER request. If others use this option, the buffer will not be returned to the original owner of the buffer.

The application may optionally specify that the buffer obtained is to be cleared when it is returned to the pool on a FREE_BUFFER request. This allows secure data to be cleared after use.

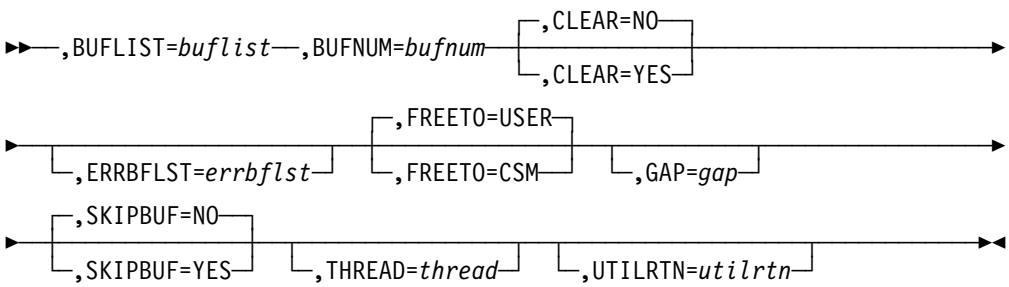
All IVTCSM REQUEST=GET_BUFFER|ASSIGN_BUFFER macroinstructions must have a corresponding FREE_BUFFER request before the buffer is considered available for reallocation by CSM, or before a buffer return exit routine is invoked for a buffer obtained specifying a buffer return exit routine. This is necessary to ensure that all users have finished using the buffer.

Syntax

main diagram



parameters-1



Parameters

The parameters are explained as follows:

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

,BUFLIST=*buflist*
A required input parameter, of an area containing a list of buffer entries. The number of entries in the list is specified by BUFNUM. An entry in the list is mapped by IVTBUFL.

The following fields in IVTBUFL are required as input for this request.

- BUFL_VERSION
- BUFL_SOURCE (Note: This field is only required when SKIPBUF=YES is specified.)
- BUFL_TOKEN

There are no fields in IVTBUFL returned as output by CSM for this request.

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

,BUFNUM=*bufnum*
A required input parameter, specifying the number of buffer entries in the list.

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

,CLEAR=NO

,CLEAR=YES
An optional parameter, specifying whether the buffer is to be cleared when returned to storage pool. The default is CLEAR=NO.

,CLEAR=NO
specifies that the buffer is not cleared when returned to the storage pool. If the buffer was originally allocated with a CLEAR value of YES, then CLEAR=NO is ignored by CSM and the buffer will be cleared when returned to the storage pool.

,CLEAR=YES
specifies that the buffer is to be cleared. Specifying CLEAR=YES will not cause a buffer to be cleared that is returned via a user-specified free

routine. However, if CLEAR=YES is specified, the buffer is cleared in the event that it is returned to the storage pool.

,ERRBFLST=*errbflst*

An optional output parameter, specifying the number of the last buffer entry that was successfully processed when an error is detected during processing of the macroinstruction.

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

,FREETO=USER

,FREETO=CSM

An optional parameter, allowing the FREERTN parameter on the IVTCSM REQUEST=GET_BUFFER macroinstruction to be overridden. The default is FREETO=USER.

,FREETO=USER

specifies that the buffer is to be returned to the free routine specified on the GET_BUFFER request.

,FREETO=CSM

specifies that the free routine address provided when the buffer was obtained is to be overridden and the buffer is to be returned to the storage pool. This option should only be used by the original owner of the buffer.

,GAP=*gap*

An optional input parameter, specifying the number of bytes used to separate buffer entries. This parameter allows the buffer entries to be in discontiguous storage. If GAP is not specified, buffer entries are not contiguous.

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

,MF=S

,MF=(L,*list addr***)**

,MF=(L,*list addr,attr***)**

,MF=(L,*list addr***,0D)**

,MF=(E,*list addr***)**

,MF=(E,*list addr***,COMPLETE)**

,MF=(E,*list addr***,NOCHECK)**

,MF=(M,*list addr***)**

,MF=(M,*list addr***,COMPLETE)**

,MF=(M,*list addr***,NOCHECK)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

Use MF=M together with the list and execute forms of the macro for service routines that need to provide different options according to user-provided input. Use the list form to define a storage area; use the modify form to set the appropriate options; then use the execute form to call the service.

IBM recommends that you use the modify and execute forms of IVTCSM in the following order:

1. Use IVTCSM ...MF=(M,list-addr,COMPLETE) specifying appropriate parameters, including all required ones.
2. Use IVTCSM ...MF=(M,list-addr,NOCHECK), specifying the parameters that you want to change.
3. Use IVTCSM ...MF=(E,list-addr,NOCHECK), to execute the macro.

,*list addr*

The name of a storage area to contain the parameters. For MF=S, MF=E, and MF=M, this can be an RS-type address or an address in register (1)-(12).

,*attr*

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

,COMPLETE

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

,NOCHECK

Specifies that the system is not to check for required parameters and is not to supply defaults for omitted optional parameters.

,PLISTVER=IMPLIED_VERSION

,PLISTVER=MAX

,PLISTVER=0
An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

  If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form; in this way, MAX ensures that the parameter list does not overwrite nearby storage.
- **0**, if you use the currently available parameters.

To code: Specify one of the following:

- IMPLIED_VERSION
- MAX
- A decimal value of 0

,RETCODE=*retcode*
An optional output parameter into which the return code is to be copied from GPR 15.

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

,RSNCODE=*rsncode*
An optional output parameter into which the reason code is to be copied from GPR 0.

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

,SKIPBUF=NO

,SKIPBUF=YES
An optional parameter, specifying whether all entries in the buffer list should be processed. The default is SKIPBUF=NO.

,SKIPBUF=NO
specifies that all the entries in the buffer list will be processed. No entries are skipped. The BUFL_SOURCE value is not examined.

,SKIPBUF=YES
specifies that the only entries in the buffer list that have a BUFL_SOURCE value indicating the user's non-CSM storage (BUFL_UDSPACE or BUFL_USTOR) will be skipped.

,THREAD=*thread*

An optional input parameter, specifying a unique identifier that is placed in the CSM trace entry to correlate trace records with the application that is requesting the buffers. It is the CSM user's responsibility to ensure that this value is different from the THREAD value specified by other users of the CSM. One way this can be achieved is by specifying an ECSA control block for THREAD.

To code: Specify the RS-type address, or address in register (2)-(12), of a 4-character field.

,UTILRTN=*utilrtn*

An optional input parameter that is issued from a utility routine. Specify the utility routine caller's address to be placed in the CSM trace entry. If this parameter is omitted, only the address of the CSM request issuer is placed in the CSM trace entry. This parameter is only relevant to the tracing process. It should be specified only if the CSM user requires identification of the caller of a utility routine in the CSM trace entry.

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

Return Codes

The following codes can be returned to the application on this macroinstruction.

Return Code	Meaning
0	Request completed successfully.
4	Request did not complete successfully. See the following reason codes to determine the type of error encountered.

Reason Code	Meaning
2	Requested function not supported at the present time, service has not been initialized.
7	Pool token specified is not valid.
8	Instance ID in the input buffer token does not match that of the buffer, possible attempt to use a buffer that has been freed.

8	System error while processing the request.

Reason Codes: Meaning

6 An abend occurred while processing this request.

IVTCSM REQUEST=GET_BUFFER

Purpose

This macroinstruction allows an application to request one or more buffers of a given size from the CSM storage pool.

Usage

For the IVTCSM REQUEST=GET_BUFFER macroinstruction, CSM allocates buffers from a pre-existing pool and returns information to the requester needed to address the buffer. This includes the ALET of a buffer that resides in a data space. The value specified on the POOLTOKN parameter must be the same value returned on the RETPTOKN parameter of the IVTCSM REQUEST=CREATE_POOL macroinstruction.

The application has the option of requesting buffers that are guaranteed to be fixed, guaranteed to be pageable or eligible to be made pageable. A pageable buffer can be obtained and used when fixed buffers are unavailable, and fixed at a later time using the IVTCSM REQUEST=FIX_BUFFER macroinstruction.

Ownership of the buffers is assigned to the requesting address space by default. This can be overridden by specifying OWNERID. The OWNERID is the ASID of the address space. Ownership of a buffer can be optionally qualified for a given task by specifying TASKID. The TASKID is a TCB address.

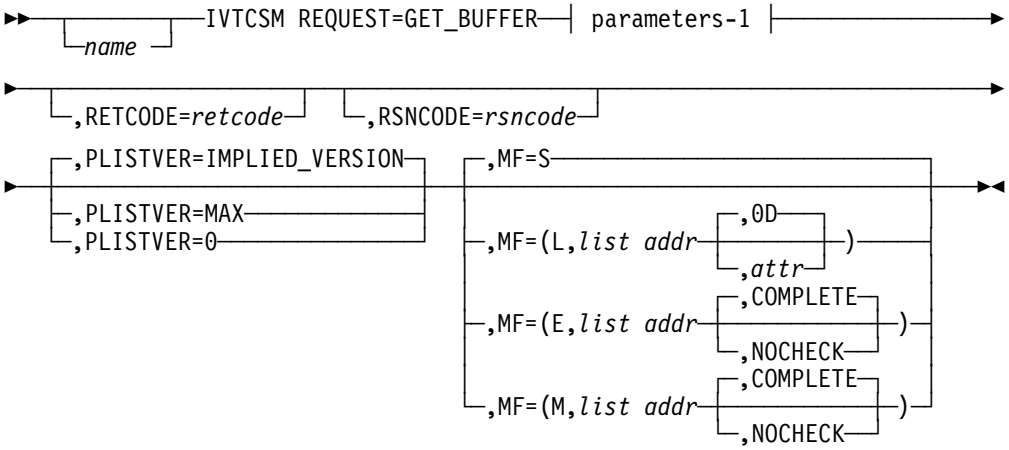
A buffer token is returned with each buffer. The buffer token is the means by which this buffer is known to CSM. This token must be used with all other requests to CSM for the associated buffer.

The application can also specify a free routine address that is to receive control when the IVTCSM REQUEST=FREE_BUFFER macroinstruction is issued for the buffer. The default is that the buffers are to be returned to CSM.

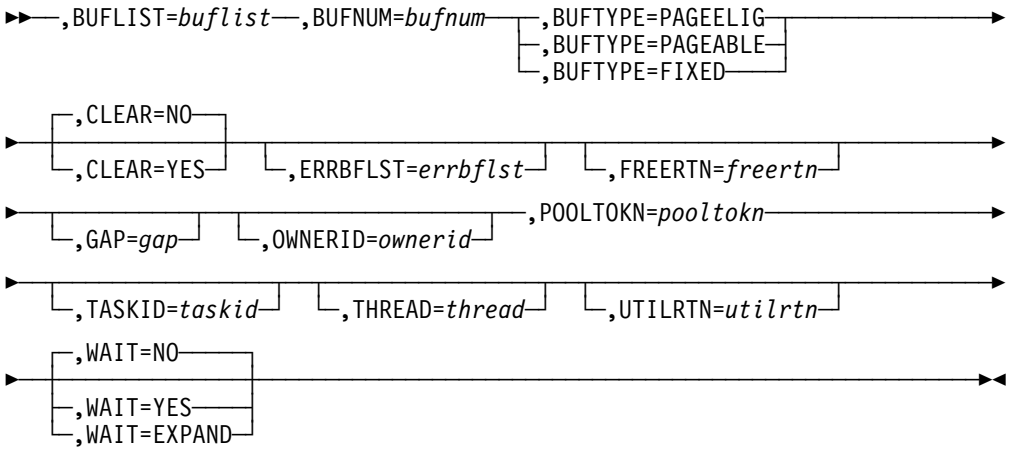
The application can also specify that the buffer obtained is to be cleared when it is returned to the pool. This provides for secure data to be cleared once processing is complete.

Syntax

main diagram



parameters-1



Parameters

The parameters are explained as follows:

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

,BUFLIST=*buflist*
A required input parameter, of an area containing a list of buffer entries. The number of entries in the list is equal to the value specified by the BUFNUM parameter. An entry in the list is mapped by IVTBUFL.

The following field in IVTBUFL is required as input for this request.

- BUFL_VERSION

The following fields in IVTBUFL are returned as output by CSM for this request.

- BUFL_SOURCE
- BUFL_TYPE
- BUFL_TOKEN
- BUFL_ALET (Note: This field is returned only if the buffer was allocated from a data space.)
- BUFL_ADDR
- BUFL_SIZE

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

,BUFNUM=*bufnum*
A required input parameter, specifying the number of buffers to be obtained.

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

,BUFTYPE=PAGEELIG

,BUFTYPE=PAGEABLE

,BUFTYPE=FIXED
A required parameter, specifying whether the buffers are to be guaranteed to be fixed, guaranteed to be pageable or eligible to be made pageable.

,BUFTYPE=PAGEELIG
indicates that the buffers are eligible to be made pageable.

,BUFTYPE=PAGEABLE
indicates that the buffers are to be guaranteed to be pageable.

,BUFTYPE=FIXED
indicates that buffers are to be guaranteed to be fixed.

,CLEAR=NO

,CLEAR=YES
An optional parameter, specifying whether the buffer is to be cleared when returned to the storage pool. The default is CLEAR=NO.

,CLEAR=NO
specifies that the buffer is not cleared when returned to the buffer pool

,CLEAR=YES
specifies that the buffer is cleared. Specifying CLEAR=YES will not cause a buffer to be cleared that is returned via a user-specified free routine. However, if CLEAR=YES is specified, the buffer is cleared in the event that it is returned to the storage pool.

,ERRBFLST=*errbflst*
An optional output parameter, containing the number of the last buffer entry that was successfully processed when an error is detected during processing of the macroinstruction.

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

,FREERTN=*freertn*
An optional input parameter that is to contain the address of an application routine that is to receive control when the buffer is freed. This allows the buffer to be passed to another application or product such as VTAM and to receive the buffer back when the receiver is finished. The free routine is scheduled for execution in the address space of the original owner of the buffer. See “Buffer Return Exit Routine” on page 18 for more information.

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

,GAP=*gap*
An optional input parameter, specifying the number of bytes used to separate buffer entries. This parameter allows the buffer entries to be in discontiguous storage. If GAP is not specified, buffer entries are contiguous.

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

,MF=S

,MF=(L,*list addr***)**

,MF=(L,*list addr,attr***)**

,MF=(L,*list addr***,0D)**

,MF=(E,*list addr***)**

,MF=(E,*list addr***,COMPLETE)**

,MF=(E,*list addr***,NOCHECK)**

,MF=(M,*list addr***)**

,MF=(M,*list addr***,COMPLETE)**

,MF=(M,*list addr***,NOCHECK)**
An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

Use MF=M together with the list and execute forms of the macro for service routines that need to provide different options according to user-provided input. Use the list form to define a storage area; use the modify form to set the appropriate options; then use the execute form to call the service.

IBM recommends that you use the modify and execute forms of IVTCSM in the following order:

1. Use IVTCSM ...MF=(M,list-addr,COMPLETE) specifying appropriate parameters, including all required ones.
2. Use IVTCSM ...MF=(M,list-addr,NOCHECK), specifying the parameters that you want to change.
3. Use IVTCSM ...MF=(E,list-addr,NOCHECK), to execute the macro.

,*list addr*
The name of a storage area to contain the parameters. For MF=S, MF=E, and MF=M, this can be an RS-type address or an address in register (1)-(12).

,*attr*
An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

,COMPLETE
Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

,NOCHECK
Specifies that the system is not to check for required parameters and is not to supply defaults for omitted optional parameters.

,OWNERID=*ownerid*
An optional input parameter, specifying the owner of the buffer being obtained.

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

,PLISTVER=IMPLIED_VERSION

,PLISTVER=MAX

,PLISTVER=0
An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

  If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form; in this way, MAX ensures that the parameter list does not overwrite nearby storage.
- **0**, if you use the currently available parameters.

To code: Specify one of the following:

- IMPLIED_VERSION
- MAX
- A decimal value of 0

,POOLTOKN=*pooltokn*

A required input parameter, of the token representing this user of this pool. This must be the token provided to the application on the associated IVTCSM REQUEST=CREATE_POOL macroinstruction.

To code: Specify the RS-type address, or address in register (2)-(12), of a 10-character field.

,RETCODE=*retcode*

An optional output parameter into which the return code is to be copied from GPR 15.

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

,RSNCODE=*rsncode*

An optional output parameter into which the reason code is to be copied from GPR 0.

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

,TASKID=*taskid*

An optional input parameter that is to contain the address of a TCB. This further qualifies the ownership of a buffer to a specific task. If TASKID is not specified, the buffer is not associated with a task but is instead associated with the issuing application's ASID.

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

,THREAD=*thread*

An optional input parameter, specifying a unique identifier that is placed in the CSM trace entry to correlate trace records with the application that is requesting the buffers. It is the CSM user's responsibility to ensure that this value is different from the THREAD value specified by other users of the CSM. One way this can be achieved is by specifying an ECSA control block for THREAD.

To code: Specify the RS-type address, or address in register (2)-(12), of a 4-character field.

,UTILRTN=*utilrtn*

An optional input parameter that is issued from a utility routine. Specify the utility routine caller's address to be placed in the CSM trace entry. If this parameter is omitted, only the address of the CSM request issuer is placed in the CSM trace entry. This parameter is only relevant to the tracing process. It should be specified only if the CSM user requires identification of the caller of a utility routine in the CSM trace entry.

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

,WAIT=NO

,WAIT=YES

,WAIT=EXPAND

An optional parameter, specifying whether or not the request should wait for buffers to become available. The default is WAIT=NO.

,WAIT=NO
specifies that this macroinstruction completes without waiting for an available buffer.

,WAIT=YES
specifies that this macroinstruction will not complete until all buffers become available. If buffers are not available, users will be suspended until enough buffers become available to satisfy the request.

,WAIT=EXPAND
specifies that this macroinstruction will wait for pool expansion to complete. If enough buffers are not available to satisfy the request, users will be suspended until expansion completes.

Return Codes

The following codes can be returned to the application on this macroinstruction.

Return Code	Meaning
0	Request completed successfully.
4	Request did not complete successfully. See the following reason codes to determine the type of error encountered.

Reason Code	Meaning
2	Requested function not supported at the present time, service has not been initialized.
4	Buffer pool cannot be expanded to satisfy request.
5	No available buffers in pool, wait not requested.
6	Pool token specified is not valid.
9	Real storage unavailable to provide a fixed buffer, wait not requested.
11	A problem has been detected with the pool associated with the CSM request. The user should free all buffers when finished using them and issue a delete pool request to terminate usage of this pool. To allocate new buffers, a new pool must be created by issuing a new create pool request.
16	Instance ID in the input pooltoken does not match that of the user, possible attempt to allocate buffers after issuing a DELETE_POOL request.
17	Extent has been overlaid. Reissue the request.
20	BUFTYPE value specified is not valid for this request.
24	ASID specified on the OWNERID parameter is not active.

8 System error while processing the request. See the following reason codes to determine the type of error encountered.

Reason Code	Meaning
1	Unable to obtain storage for request.
3	Unable to create ALET for data space.
4	Error encountered, while creating the data space.
5	Unable to create another data space. Number of data spaces exceeds the maximum.
6	An abend occurred while processing this request.

IVTCSM REQUEST=PAGE_BUFFER

Purpose

This macroinstruction allows an application to change the pageable state of a buffer.

Usage

An application can use this macroinstruction to make the buffer guaranteed to be pageable or eligible to be paged.

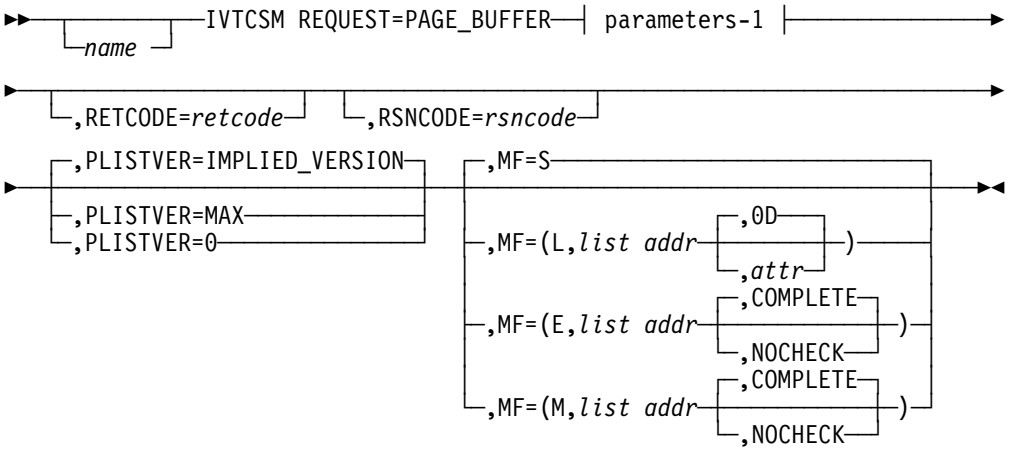
When BUFTYPE=PAGEELIG is specified on this macroinstruction, the buffer is marked as eligible to be paged. The buffer is not physically unfixed unless CSM requires real storage to satisfy another CSM request. This avoids the potential overhead of unnecessary fixing and freeing of storage.

This macroinstruction may be used to avoid consuming real storage for data that is being held in a buffer for possible use at a later time.

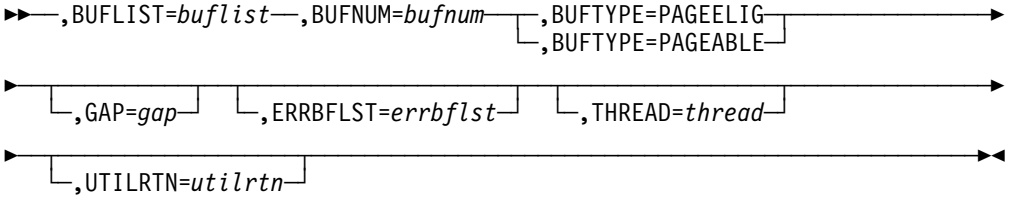
When BUFTYPE=PAGEABLE is specified on this macroinstruction, the buffer is marked as guaranteed to be pageable. This macroinstruction may be used when a system service requires pageable storage on input. This macroinstruction can be issued only for a buffer consisting of one image. This restriction guarantees that a user of a buffer that has multiple images can successfully issue a FIX_BUFFER request if necessary. Fixing a buffer requires that the entire buffer be fixed, even if the user is interested in only a piece of the buffer.

Syntax

main diagram



parameters-1



Parameters

The parameters are explained as follows:

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

,BUFLIST=*buflist*
A required input parameter, of an area containing a list of buffer entries. The number of entries in the list is specified by BUFNUM. An entry in the buffer list is mapped by IVTBUFL.

The following fields in IVTBUFL are required as input for this request.

- BUFL_VERSION
- BUFL_TOKEN

The following field in IVTBUFL is returned as output by CSM for this request.

- BUFL_TYPE

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

,BUFNUM=*bufnum*
A required input parameter, specifying the number of buffers to be made pageable or eligible to be paged.

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

,BUFTYPE=PAGEELIG

,BUFTYPE=PAGEABLE
A required parameter, specifying whether the buffers are to be guaranteed to be pageable or eligible to be made pageable.

,BUFTYPE=PAGEELIG
indicates that the buffers are eligible to be made pageable.

,BUFTYPE=PAGEABLE
indicates that the buffers are to be guaranteed to be pageable.

,ERRBFLST=*errbflst*
An optional output parameter, containing the number of the last buffer entry that was successfully processed when an error is detected during processing of the macroinstruction.

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

,GAP=*gap*
An optional input parameter, specifying the number of bytes used to separate buffer entries. This parameter allows the buffer entries to be in discontiguous storage. If GAP is not specified, buffer entries are contiguous.

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

,MF=S

,MF=(L,*list addr***)**

,MF=(L,*list addr,attr***)**

,MF=(L,*list addr***,0D)**

,MF=(E,*list addr***)**

,MF=(E,*list addr***,COMPLETE)**

,MF=(E,*list addr***,NOCHECK)**

,MF=(M,*list addr***)**

,MF=(M,*list addr***,COMPLETE)**

,MF=(M,*list addr***,NOCHECK)**

An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

Use MF=M together with the list and execute forms of the macro for service routines that need to provide different options according to user-provided input. Use the list form to define a storage area; use the modify form to set the appropriate options; then use the execute form to call the service.

IBM recommends that you use the modify and execute forms of IVTCSM in the following order:

1. Use IVTCSM ...MF=(M,list-addr,COMPLETE) specifying appropriate parameters, including all required ones.
2. Use IVTCSM ...MF=(M,list-addr,NOCHECK), specifying the parameters that you want to change.
3. Use IVTCSM ...MF=(E,list-addr,NOCHECK), to execute the macro.

,*list addr*
The name of a storage area to contain the parameters. For MF=S, MF=E, and MF=M, this can be an RS-type address or an address in register (1)-(12).

,*attr*
An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

,COMPLETE
Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

,NOCHECK
Specifies that the system is not to check for required parameters and is not to supply defaults for omitted optional parameters.

,PLISTVER=IMPLIED_VERSION

,PLISTVER=MAX

,PLISTVER=0
An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

  If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form; in this way, MAX ensures that the parameter list does not overwrite nearby storage.
- **0**, if you use the currently available parameters.

To code: Specify one of the following:

- IMPLIED_VERSION
- MAX
- A decimal value of 0

,RETCODE=*retcode*
An optional output parameter into which the return code is to be copied from GPR 15.

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

,RSNCODE=*rsncode*

An optional output parameter into which the reason code is to be copied from GPR 0.

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

,THREAD=*thread*

An optional input parameter, specifying a unique identifier that is placed in the CSM trace entry to correlate trace records with the application that is requesting the buffers. It is the CSM user's responsibility to ensure that this value is different from the THREAD value specified by other users of the CSM. One way this can be achieved is by specifying an ECSA control block for THREAD.

To code: Specify the RS-type address, or address in register (2)-(12), of a 4-character field.

,UTILRTN=*utilrtn*

An optional input parameter that is issued from a utility routine. Specify the utility routine caller's address to be placed in the CSM trace entry. If this parameter is omitted, only the address of the CSM request issuer is placed in the CSM trace entry. This parameter is only relevant to the tracing process. It should be specified only if the CSM user requires identification of the caller of a utility routine in the CSM trace entry.

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

Return Codes

The following codes can be returned to the application on this macroinstruction.

Return Code	Meaning
0	Request completed successfully.
4	Request did not complete successfully. See the following reason codes to determine the type of error encountered.

Reason Code	Meaning
2	Requested function not supported at the present time, service has not been initialized.
7	Pool token specified is not valid.
8	Instance ID in the input pool token does not match that of the buffer, possible attempt to use a buffer that has been freed.
10	Request to make a buffer pageable denied, more than one image of the buffer exists.
20	BUFTYPE value specified is not valid for this request.

8 System error while processing the request

Reason Codes:	Meaning
1	Unable to obtain storage for request.
6	An abend occurred while processing this request.

IVTCSM REQUEST=RESOURCE_STATS

Purpose

This macroinstruction requests that the address of the information required to monitor the usage of ECSA, data space, and fixed storage be returned.

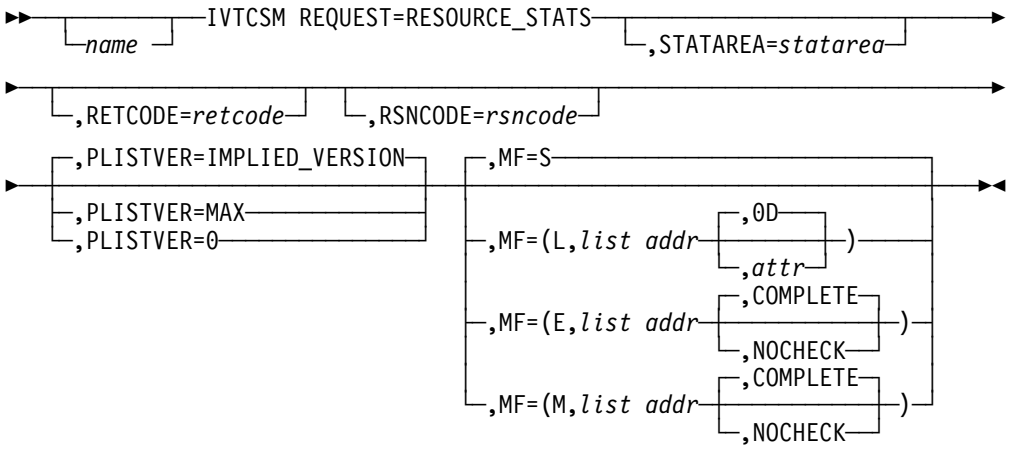
Usage

When this macroinstruction is issued, CSM returns the address of a 4-byte area containing the status of ECSA, data space, and fixed storage resources. Applications can issue this macroinstruction during initialization processing and use the address throughout normal processing. The status information contained in this area indicates whether the use of a resource is normal, constrained, or critical. If a resource usage is determined to be constrained or critical, users of CSM can take action to prevent critical shortages that might jeopardize the application's or system's operation, including:

- Selecting a different storage source for buffer pools
- Limiting usage of fixed storage.

Syntax

main diagram



Parameters

The parameters are explained as follows:

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

,MF=S

,MF=(L,*list addr***)**

,MF=(L,*list addr,attr***)**

,MF=(L,*list addr***,0D)**

,MF=(E,*list addr***)**

,MF=(E,*list addr***,COMPLETE)**

,MF=(E,*list addr***,NOCHECK)**

,MF=(M,*list addr***)**

,MF=(M,*list addr***,COMPLETE)**

,MF=(M,*list addr***,NOCHECK)**
An optional input parameter that specifies the macro form.

Use MF=S to specify the standard form of the macro, which builds an inline parameter list and generates the macro invocation to transfer control to the service. MF=S is the default.

Use MF=L to specify the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter may be coded with the list form of the macro.

Use MF=E to specify the execute form of the macro. Use the execute form together with the list form of the macro for applications that require reentrant code. The execute form of the macro stores the parameters into the storage area defined by the list form, and generates the macro invocation to transfer control to the service.

Use MF=M together with the list and execute forms of the macro for service routines that need to provide different options according to user-provided input. Use the list form to define a storage area; use the modify form to set the appropriate options; then use the execute form to call the service.

IBM recommends that you use the modify and execute forms of IVTCSM in the following order:

1. Use IVTCSM ...MF=(M,list-addr,COMPLETE) specifying appropriate parameters, including all required ones.
2. Use IVTCSM ...MF=(M,list-addr,NOCHECK), specifying the parameters that you want to change.
3. Use IVTCSM ...MF=(E,list-addr,NOCHECK), to execute the macro.

,*list addr*
The name of a storage area to contain the parameters. For MF=S, MF=E, and MF=M, this can be an RS-type address or an address in register (1)-(12).

,*attr*

An optional 1- to 60-character input string that you use to force boundary alignment of the parameter list. Use a value of 0F to force the parameter list to a word boundary, or 0D to force the parameter list to a doubleword boundary. If you do not code *attr*, the system provides a value of 0D.

,COMPLETE

Specifies that the system is to check for required parameters and supply defaults for omitted optional parameters.

,NOCHECK

Specifies that the system is not to check for required parameters and is not to supply defaults for omitted optional parameters.

,PLISTVER=IMPLIED_VERSION

,PLISTVER=MAX

,PLISTVER=0

An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:

- **IMPLIED_VERSION**, which is the lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED_VERSION is the default.
- **MAX**, if you want the parameter list to be the largest size currently possible. This size might grow from release to release and affect the amount of storage that your program needs.

  If you can tolerate the size change, IBM recommends that you always specify PLISTVER=MAX on the list form of the macro. Specifying MAX ensures that the list-form parameter list is always long enough to hold all the parameters you might specify on the execute form; in this way, MAX ensures that the parameter list does not overwrite nearby storage.
- **0**, if you use the currently available parameters.

To code: Specify one of the following:

- IMPLIED_VERSION
- MAX
- A decimal value of 0

,RETCODE=*retcode*

An optional output parameter into which the return code is to be copied from GPR 15.

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

,RSNCODE=*rsncode*

An optional output parameter into which the reason code is to be copied from GPR 0.

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

,STATAREA=*statarea*

An optional output parameter that contains the address of an area containing the resource statistics mapped by IVTSTATA.

The information returned is in a non-fetch protected area and can be accessed while executing in any storage key.

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

Return Codes

The following codes can be returned to the application on this macroinstruction.

Return Code	Meaning
0	Request completed successfully.
4	Request did not complete successfully. See the following reason codes to determine the type of error encountered.

Reason Code	Meaning
2	Requested function not supported at the present time, service has not been initialized.

Appendix A. Return and Reason Codes for the IVTCSM Macroinstruction

This chapter describes return and reason codes that can be returned to an application issuing the IVTCSM macroinstruction. For all macroinstructions that invoke CSM, the application can examine return codes in register 15 and reason codes in register 0.

0 Request completed successfully.

4 Request did not complete successfully, see reason code to identify the type of error encountered.

Reason Codes: Meaning

1 Requested function not supported.

2 Requested function not supported at the present time, service has not been initialized.

3 Specified buffer size is larger than supported size.

4 Buffer pool cannot be expanded to satisfy request.

5 No available buffers in pool, wait not requested.

6 Pool token specified is not valid.

7 Buffer token specified is not valid.

8 Instance ID in the input buffer token does not match that of the buffer, possible attempt to use a buffer that has been freed.

9 Real storage unavailable to provide a fixed buffer, wait not requested.

10 Request to make a buffer pageable denied, more than one image of the buffer exists.

11 A problem has been detected with the pool associated with the CSM request. The user should free all buffers when finished using them and issue a delete pool request to terminate usage of this pool. To allocate new buffers, a new pool must be created by issuing a new create pool request.

12 Address and length specified on a copy data request for a source buffer descriptor is outside the bounds of the CSM buffer represented by the specified pool token.

13 Address and length specified on a copy data request for a target buffer descriptor is outside the bounds of the CSM buffer represented by the specified pool token.

14 Copy operation resulted in truncation of source data due to insufficient buffer space provided by the target buffer list.

15 Assign buffer request failed because the state of the buffer is guaranteed to be pageable.

16 Instance ID in the input pool token does not match that of the user, possible attempt to allocate buffers after issuing a DELETE_POOL request.

17 Extent has been overlaid. Reissue the request.

18 BUFL_SOURCE value is not valid for an entry in the Source buffer list (SRCLIST).

19 BUFL_SOURCE value is not valid for an entry in the Target buffer list (TRGLIST).

20 BUFTYPE value specified is not valid for this request

21 BUFSOURC value is not valid for this request.

22	Source and target buffers overlap, no data has been copied.
23	Unable to create the specified pool. Creation of the pool would cause the ECSA maximum limit to be exceeded.
24	ASID specified on the OWNERID parameter is not active.

8 System error while processing the request

Reason Codes: Meaning

1	Unable to obtain storage for request.
2	Schedule SRB fail for PC routine.
3	Unable to create ALET for data space.
4	Error encountered, while creating the data space.
5	Unable to create another data space. Number of data spaces exceeds the maximum.
6	An abend occurred while processing this request.

Appendix B. CSM DSECTs

This appendix contains the CSM DSECTs. For general information on the use and purpose of DSECTs, see the *OS/390 eNetwork Communications Server: SNA Programming*.

The DSECTs are shown as an abbreviated form of an assembler listing. The first column indicates the offsets within the DSECT and is the “LOC” column of an assembler listing. The object code, address columns and statement number columns of the listing, however, are not included. The source statements and comments are found next to the offset column. All numbers in the offset column are in hexadecimal.

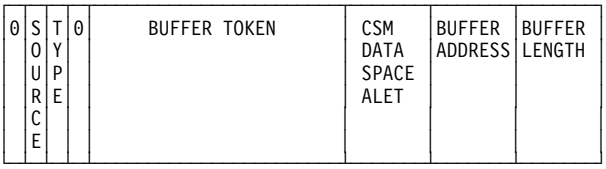
CSM Buffer List Entry (IVTBUFL)

The DSECT IVTBUFL maps an entry in the CSM buffer list which can be indicated by the BUFLIST, SRCLIST or TARGLIST parameters on the IVTCSM macroinstruction. The buffer list pointed to by the BUFLIST parameter is required on the following IVTCSM requests:

- IVTCSM REQUEST=ASSIGN_BUFFER
- IVTCSM REQUEST=CHANGE_OWNER
- IVTCSM REQUEST=FIX_BUFFER
- IVTCSM REQUEST=FREE_BUFFER
- IVTCSM REQUEST=GET_BUFFER
- IVTCSM REQUEST=PAGE_BUFFER.

The buffer list pointed to by SRCLIST and TARGLIST is required on the IVTCSM REQUEST=COPY_DATA macroinstruction.

The format of the CSM buffer list is as follows.



Byte (hex)	Contents
00	Version
01	Indicates the buffer source
	X'80' ECSA
	X'40' Data space
	X'20' User data space
	X'10' User storage area other than a data space
02	Indicates the state of the buffers
	X'80' Fixed
	X'40' Pageable
	X'20' Eligible to be page freed
03	0
04 - 0F	Buffer token
10 - 13	ALET for the data space

14 - 17 Address of buffer
18 - 1B Length of buffer

```
  LOC    SOURCE STATEMENT
000000 IVTBUFL  DSECT                          BUFFER DESCRIPTOR
000000 BUFL_VERSION   DS    X                  VERSION OF BUFFER DESCRIPTOR
       BUFL_VERSIONC  EQU   X'00'              VERSION 0
000001 BUFL_SOURCE    DS    X                  BUFFER SOURCE
       BUFL_CECSA     EQU   X'80'              INDICATES THAT THE STORAGE
       *                                       IS IN CSM ECSA
       BUFL_CDSPACE   EQU   X'40'              INDICATES THAT THE STORAGE
       *                                       IS IN CSM DATA SPACE
       BUFL_UDSPACE   EQU   X'20'              INDICATES THAT THE STORAGE
       *                                       IS IN A USER DATA SPACE
       BUFL_USTOR     EQU   X'10'              INDICATES THAT THE STORAGE
       *                                       IS A USER'S STORAGE OTHER THAN
       *                                       A DATA SPACE
000002 BUFL_TYPE      DS    X                  BUFFER TYPE
       BUFL_FIXED     EQU   X'80'              INDICATES THAT THE STORAGE IS
       *                                       IN A GUARANTEED TO BE FIXED
       *                                       STATE
       BUFL_PAGEABLE  EQU   X'40'              INDICATES THAT THE STORAGE IS
       *                                       IN A GUARANTEED TO BE PAGEABLE
       *                                       STATE
       BUFL_PAGEELIG  EQU   X'20'              INDICATES THAT THE STORAGE IS
       *                                       ELIGIBLE TO BE PAGEFREED BY
       *                                       CSM
000003                DS    XL1                RESERVED
000004 BUFL_TOKEN     DS    XL12               CSM BUFFER TOKEN
000010 BUFL_ALET      DS    F                  DATA SPACE ALET
000014 BUFL_ADDR      DS    A                  POINTER TO BUFFER
000018 BUFL_SIZE      DS    F                  THE SIZE OF THE ALLOCATED BUFFER
       *                                       ON GET_BUFFER REQUESTS, THE DATA
       *                                       LENGTH ON COPY_DATA REQUESTS
00001C BUFL_END       DS    0F                 END OF IVTBUFL
```

CSM Data Space Information (IVTDATSP)

IVTDATSP maps the information required to include CSM data space buffers in a user dump. The address of this information can be optionally returned to the user on the IVTCSM REQUEST=CREATE_POOL and IVTCSM REQUEST=DUMP_INFO macroinstructions.

The following describes the layout of the IVTDATSP DSECT.

DATS	NUMBER OF DATA SPACES	LENGTH OF DATA SPACE ENTRIES	DATA SPACE ENTRIES MAPPED BY DATSP_ENT

Byte (hex)	Field name	Contents
00—03	DATSP_ACRO	DATS
04—07	DATSP_SNUM	Number of data spaces
08—0B	DATSP_SLEN	Length of a data space information entry
0C—	DATSP_SINF	All of the data space entries. The number of entries is determined by DATSP_SNUM. Each entry is separately mapped by the DATSP_ENT DSECT and includes the following information:

Byte (hex)	Contents
00—07	STOKEN for the data space
08—0F	Data space name
10—	ALET for the data space

```
  LOC    SOURCE STATEMENT
000000 IVTDATSP DSECT                      CSM DATA SPACE INFORMATION
000000 DATSP_ACRO DS    CL4                Eyecatcher 'DATS'
000004 DATSP_SNUM DS    F                  NUMBER OF DATA SPACES
000008 DATSP_SLEN DS    F                  LENGTH OF A DATA SPACE
       *                                     INFORMATION ENTRY
00000C DATSP_SINF DS    5XL20              CONTAINS ALL THE DATA SPACE
       *                                     INFORMATION ENTRIES.  THESE
       *                                     ENTRIES ARE MAPPED BY THE
       *                                     DATSP_ENT DSECT.
       *                                     THERE WILL BE ONE ENTRY FOR
       *                                     EACH DATA SPACE INDICATED BY
       *                                     DATSP_SNUM, AND EACH ENTRY
       *                                     MUST BE SEPARATELY MAPPED BY
       *                                     THE DATSP_ENT DSECT.
000000 DATSP_ENT  DSECT                    DATA SPACE INFORMATION ENTRY
000000 DATSP_TOKN DS    XL8                STOKEN FOR THE DATA SPACE
000008 DATSP_NAME DS    XL8                DATA SPACE NAME
000010 DATSP_ALET DS    F                  ALET FOR THE DATA SPACE
```

CSM Resource Status Area (IVTSTATA)

IVTSTATA maps the information required to monitor the usage of CSM resources such as ECSA, data space and fixed storage. The address of the information can be optionally returned to the user on the IVTCSM REQUEST=CREATE_POOL and IVTCSM REQUEST=RESOURCE_STATS macroinstructions.

For each resource type, two bits are defined: one to indicate the usage of the resource is constrained and one to indicate the usage is critical. If neither bit is set, the usage of the resource is considered to be normal. This allows the users of CSM to take action based on resource usage to prevent critical shortages that might jeopardize the application's or system's operation. Possible user actions might include selecting a different storage source for buffer pools or limiting usage of fixed storage. For information on other actions that an installation may consider to resolve resource shortages, see “Monitoring CSM” on page 4.

```
  LOC    SOURCE STATEMENT
000000 IVTSTATA DSECT                      CSM resource status area
000000 STATA_ACRO  DS    CL4               Eyecatcher 'STAT'
000004 STATA_LEN   DS    XL2               Number of bytes of resource
       *                                     statistics
000006 STATA_FLAG  DS    X                 Status flag
       STATA_ESTAT EQU   X'C0'             ECSA storage status
```

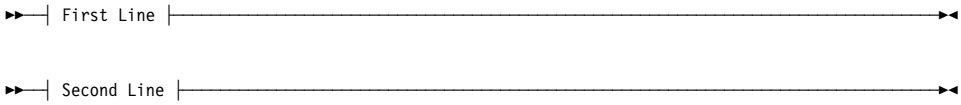
```
         STATA_ECRIT EQU   X'80'               ECSA storage critical
         STATA_ECONS EQU   X'40'               ECSA storage constrained
         *
         STATA_DSTAT EQU   X'30'               Data space storage status
         STATA_DCRIT EQU   X'20'               Data space storage
         *                                     critical
         STATA_DCONS EQU   X'10'               Data space storage
         *                                     constrained
         *
         STATA_FSTAT EQU   X'0C'               Fixed storage status
         STATA_FCRIT EQU   X'08'               Fixed storage critical
         STATA_FCONS EQU   X'04'               Fixed storage constrained
000007               DS    X                  Reserved
```

Appendix C. How to Read the Syntax Diagrams

This section describes how to read the syntax diagrams used in this book.

- Read the diagrams from left-to-right, top-to-bottom, following the main path line. Each diagram begins on the left with double arrowheads (►►) and ends on the right with two arrowheads facing each other (►◄).

- If a diagram is longer than one line, the first line ends with a single arrowhead (►) and the second line begins with a single arrowhead.



- Required operands and values appear on the main path line.

  You must code required operands and values.

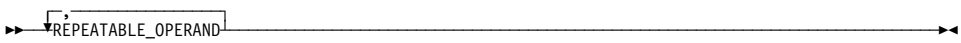
  If there is more than one mutually exclusive required operand or value to choose from, they are stacked vertically in alphanumeric order.

- Optional operands and values appear below the main path line.

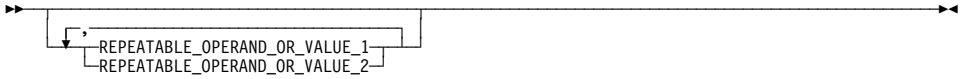
  You can choose not to code optional operands and values.

  If there is more than one mutually exclusive optional operand or value to choose from, they are stacked vertically in alphanumeric order below the main path line.

- An arrow returning to the left above an operand or value on the main path line means that the operand or value can be repeated. The comma means that each operand or value must be separated from the next by a comma.



- An arrow returning to the left above a group of operands or values means more than one can be selected or a single one can be repeated.



- A word in all uppercase is a operand or value you must spell exactly as shown. In this example, you must code ***OPERAND***.

  VTAM commands are not case sensitive. You can code them in uppercase or lowercase.

If an operand or value can be abbreviated, the abbreviation is discussed in the text associated with the syntax diagram.

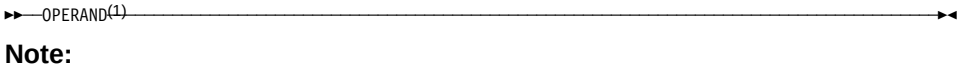
- If a diagram shows a character that is not alphanumeric (such as parentheses, periods, commas, and equal signs), you must code the character as part of the syntax. In this example, you must code ***OPERAND=(001,0.001)***.

- If a diagram shows a blank space, you must code the blank space as part of the syntax. In this example, you must code ***OPERAND=(001 FIXED)***.

- Default operands and values appear above the main path line. VTAM uses the default if you omit the operand entirely.

- A word in all lowercase italics is a *variable*. Where you see a variable in the syntax, you must replace it with one of its allowable names or values, as defined in the text.

- References to syntax notes appear as numbers enclosed in parentheses above the line. Do not code the parentheses or the number.



Note:
1 An example of a syntax note.

- Some diagrams contain *syntax fragments*, which serve to break up diagrams that are too long, too complex, or too repetitious. Syntax fragment names are in mixed case and are shown in the diagram and in the heading of the fragment. The fragment is placed below the main diagram.

Syntax Fragment:

Parameter Descriptions

Upper case parameter values represent macro-defined values that must be coded exactly as shown. Parameter values that are in lower case italics represent variable data.

Specify a location that is to be source of input data or the target of output data. The location must be defined in a manner that is consistent with the indicated data type. If you wish to pass the storage address in general purpose register 2-12, code a valid expression for the register within parentheses. Otherwise, code an expression that is valid as a storage operand on RS-type instructions.

Exception: If a register is specified for RETCODE or RSNCODE, the output is loaded straight into the register.

Appendix D. Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make them available in all countries in which IBM operates. 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. Subject to IBM's valid intellectual property or other legally protectable rights, any functionally equivalent product, program, or service may be used instead of the IBM product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, are the responsibility of the user.

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
500 Columbus Avenue
Thornwood, NY 10594
USA

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement.

This document is not intended for production use and is furnished as is without any warranty of any kind, and all warranties are hereby disclaimed including the warranties of merchantability and fitness for a particular purpose.

IBM is required to include the following statements in order to distribute portions of this document and the software described herein to which contributions have been made by The University of California.

Portions herein © Copyright 1979, 1980, 1983, 1986, Regents of the University of California. Reproduced by permission. Portions herein were developed at the Electrical Engineering and Computer Sciences Department at the Berkeley campus of the University of California under the auspices of the Regents of the University of California.

Portions of this publication relating to RPC are Copyright © Sun Microsystems, Inc., 1988, 1989.

Some portions of this publication relating to X Window System** are Copyright © 1987, 1988 by Digital Equipment Corporation, Maynard, Massachusetts, and the Massachusetts Institute Of Technology, Cambridge, Massachusetts. All Rights Reserved.

Some portions of this publication relating to X Window System are Copyright © 1986, 1987, 1988 by Hewlett-Packard Corporation.

Permission to use, copy, modify, and distribute the M.I.T., Digital Equipment Corporation, and Hewlett-Packard Corporation portions of this software and its documentation for any purpose without fee is hereby granted, provided that the

above copyright notice appears in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the names of M.I.T., Digital, and Hewlett-Packard not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission. M.I.T., Digital, and Hewlett-Packard make no representation about the suitability of this software for any purpose. It is provided “as is” without express or implied warranty.

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

ACF/VTAM
Advanced Peer-to-Peer Networking
AD/Cycle
AIX
AIX/ESA
AnyNet
APPN
AS/400
BookManager
C/370
CICS
DB2
DFSMS
DFSMS/MVS
ESCON
ES/9000
ES/9370
EtherStreamer
Extended Services
FFST
GDDM
Hardware Configuration Definition
IBM
LANStreamer
Library Reader
MVS/ESA
MVS/SP
MVS/XA
NetView
Nways
OpenEdition
OS/2
OS/390
PS/2
RACF
RETAIN
RISC System/6000
RS/6000
SAA
System/360
System/370
System/390
VTAM
3090

The following terms are trademarks of other companies:

ATM is a trademark of Adobe Systems, Incorporated.

BSC is a trademark of BusiSoft Corporation.

CSA is a trademark of Canadian Standards Association.

DCE is a trademark of The Open Software Foundation.

HYPERchannel is a trademark of Network Systems Corporation.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

Microsoft, Windows, and the Windows 95 logo are trademarks or registered trademarks of Microsoft Corporation.

Other company, product, and service names, which may be denoted by a double asterisk (**), may be trademarks or service marks of others.

Bibliography

eNetwork Communications Server for OS/390 V2R5 Publications

Following are descriptions of the books in the eNetwork Communications Server for OS/390 V2R5 library. The books are arranged in the following categories:

- Softcopy Information
- Marketing Information
- Planning
- Installation, Resource Definition, and Tuning
- Operation
- Customization
- Writing Application Programs
- Diagnosis
- Messages and Codes
- APPC Application Suite.
- Multiprotocol Transport Networking (MPTN) Architecture publications

The complete set of unlicensed books in this section can be ordered using a single order number, SBOF-7011.

Softcopy Information

IBM Networking Softcopy Collection Kit CD-ROM(SK2T-6012).

The softcopy library contains softcopy versions of the licensed and unlicensed books for eNetwork Communications Server for OS/390 V2R5.

All of the unlicensed and licensed books described in this section are available in softcopy on this CD-ROM. These softcopy files can be read using any of the IBM BookManager READ programs. They can also be read with the IBM Library Reader program shipped on this CD.

The CD also contains softcopy of the unlicensed books of many other products.

Marketing Information

A Networking Overview and the following IBM Networking Previews are available:

- VTAM
- TCP/IP

Ask your IBM marketing representative for more information.

Planning

OS/390 eNetwork Communications Server: SNA Planning and Migration Guide (SC31-8622). This guide helps you upgrade to eNetwork Communications Server for OS/390 V2R5. It includes:

- Installation procedures
- Planning to upgrade
  - Upward and downward compatibility
  - Software and hardware requirements
  - Storage requirements
  - Impacts of new functions and enhancements performed without changes to user interfaces
  - Changes to installation process
- Upgrading user interfaces
  - Changes to start options
  - Changes to buffer pools
  - Changes to definition statements
  - Changes to IBM-supplied default user-definable tables and modules
  - Changes to user-definable table macroinstructions
  - Changes to commands
  - Changes to messages
  - Changes to SNA application programming interface
  - Changes to installation-wide exit routines
  - Changes to control blocks
- Implementing optional functions and enhancements introduced in eNetwork Communications Server for OS/390 V2R5.
  - Overview of each new function and enhancement introduced since VTAM V4R4
  - Pointers to other books in the library where implementation details can be found.

OS/390 eNetwork Communications Server: IP Planning and Migration Guide (SC31-8512). This book is intended to help you plan for TCP/IP whether you are migrating from a previous version or installing TCP/IP for the first time. This book also identifies the suggested and required modifications needed to enable you to use the enhanced functions provided with TCP/IP.

Installation, Resource Definition, Configuration, and Tuning

Program Directory. These documents are shipped with the product tape and explains the steps for installing VTAM and TCP/IP.

OS/390 eNetwork Communications Server: IP Configuration (SC31-8513). This book is for people who

want to configure, customize, administer, and maintain TCP/IP. Familiarity with MVS operating system, TCP/IP protocols, and IBM Time Sharing Option (TSO) is recommended.

OS/390 eNetwork Communications Server: SNA Network Implementation (SC31-8563). This book presents the major concepts involved in implementing a SNA network, and includes:

- Buffer pools, slowdown, pacing, storage considerations
- Implementation considerations
- Sample major node definitions
- Migration considerations
- Tables and filters
- TSO, VCNS, and other programs that run with VTAM
- Tuning procedures
- VTAM start options.

Use this book in conjunction with the *OS/390 eNetwork Communications Server: SNA Resource Definition Reference*

OS/390 eNetwork Communications Server: SNA Resource Definition Reference (SC31-8565). This book describes each VTAM definition statement, start option, and macroinstruction for user tables. It also describes NCP definition statements that affect VTAM. The information includes:

- IBM-supplied default tables (logon mode and USS)
- Major node definitions
- User-defined tables and filters
- VTAM start options.

If you are unfamiliar with the major concepts involved in implementing a SNA network, use this book in conjunction with the *OS/390 eNetwork Communications Server: SNA Network Implementation*.

OS/390 eNetwork Communications Server: SNA Resource Definition Samples (SC31-8566). This book contains sample definitions to help you implement VTAM functions in your networks, and includes sample major node definitions. Use this book in conjunction with the *OS/390 eNetwork Communications Server: SNA Network Implementation* and *OS/390 eNetwork Communications Server: SNA Resource Definition Reference*

OS/390 eNetwork Communications Server: AnyNet SNA over TCP/IP (SC31-8578). This guide provides information to help you install, configure, use, and diagnose SNA over TCP/IP.

OS/390 eNetwork Communications Server: AnyNet Sockets over SNA (SC31-8577). This guide provides information to help you install, configure, use, and diagnose Sockets over SNA. It also provides information to help you prepare application programs to use sockets over SNA.

Operation

OS/390 eNetwork Communications Server: IP User's Guide (GC31-8514). This book is for people who want to use TCP/IP for data communication activities such as FTP and Telnet. Familiarity with MVS operating system and IBM Time Sharing Option (TSO) is recommended.

OS/390 eNetwork Communications Server: Operation (SC31-8567). This book serves as a reference for programmers and operators requiring detailed information about specific operator commands. The information includes:

- VTAM commands and start options
- Logon manager commands
- DISPLAY output examples (messages received)
- VSCS commands.

OS/390 eNetwork Communications Server: Operation Quick Reference (SX75-0121). This book contains essential information about VTAM operator commands.

High Speed Access Services User's Guide (GC31-8676).

Customization

OS/390 eNetwork Communications Server: SNA Customization (LY43-0110). This book enables you to customize VTAM, and includes:

- Communication network management (CNM) routing table
- Logon-interpret routine requirements
- Logon manager installation-wide exit routine for the CLU search exit
- TSO/VTAM installation-wide exit routines
- VTAM installation-wide exit routines:
  - Command verification exit (ISTCMMND)
  - Configuration services XID exit (ISTEXCCS) with description of IBM-supplied default exit
  - Directory services management exit (ISTEXCDM)
  - Generic resource resolution exit (ISTEXCGR)
  - Performance monitor exit (ISTEXCPM)
  - SDDLU exit (ISTEXCSD) with description of IBM-supplied default exit
  - Session accounting exit (ISTAUCAG)
  - Session authorization exit (ISTAUCAT)

  - Session management exit (ISTEXCAA) with example
  - TPRINT processing exit (ISTRAEUE)
  - USERVAR exit (ISTEXCUV) with description of IBM-supplied default exit
  - Virtual route pacing window size calculation exit (ISTPUCWC)
  - Virtual route selection exit (ISTEXCVR).

OS/390 eNetwork Communications Server: IP Network Print Facility (SC31-8522). This book is for system programmers and network administrators who need to prepare their network to route VTAM, JES2, or JES3 printer output to remote printers using TCP/IP.

Writing Application Programs

OS/390 eNetwork Communications Server: IP API Guide (SC31-8516). This book describes the syntax and semantics of program source code necessary to write your own application programming interface (API) into TCP/IP. You can use this interface as the communication base for writing your own client or server application. You can also use this book to adapt your existing applications to communicate with each other using sockets over TCP/IP.

OS/390 eNetwork Communications Server: IP CICS Sockets Guide (SC31-8521). This book is for people who want to set up, write application programs for, and diagnose problems with the socket interface for CICS using TCP/IP for MVS.

OS/390 eNetwork Communications Server: IP IMS Sockets Guide (SC31-8546). This book is for programmers who want application programs that use the IMS TCP/IP application development services provided by IBM TCP/IP for MVS.

OS/390 eNetwork Communications Server: IP Programmer's Reference (SC31-8515). This book describes the syntax and semantics of a set of high-level application functions that you can use to program your own applications in a TCP/IP environment. These functions provide support for application facilities, such as user authentication, distributed databases, distributed processing, network management, and device sharing. Familiarity with the MVS operating system, TCP/IP protocols, and IBM Time Sharing Option (TSO) is recommended.

OS/390 eNetwork Communications Server: SNA Programming (SC31-8573). This book describes how to use VTAM macroinstructions to send data to and receive data from (1) a terminal in either the same or a different domain, or (2) another application program in either the same or a different domain. The information includes:

- API concepts
  - Cryptography
  - RUs and exchanges
  - Session establishment and termination
- BIND area format
- Communication Network Management Interface
- Dictionary of VTAM macroinstructions
- OPEN or CLOSE errors
- Operating system differences
- Program Operator Coding requirements
- RAPI DSECTs and control block mappings (ACB, ADSP, BLENT, CV29, EXLST, MTS, NIB, NIB DEVCHAR, NIB PROC, RH, RPL, RPL RTNCD=FDB2=FDBK=DSECT)
- RAPI global variables
- Vector lists
- RPL-based macroinstructions
- RPL RTNCD,FDB2 codes
- User exit routines.

OS/390 eNetwork Communications Server: SNA Programmers LU 6.2 Guide (SC31-8581). This book describes how to use the VTAM LU 6.2 application programming interface for host application programs. This book applies to programs that use only LU 6.2 sessions or that use LU 6.2 sessions along with other session types. (Only LU 6.2 sessions are covered in this book.) The information includes:

- VTAM's implementation of the LU 6.2 architecture
- Design considerations for LU 6.2 application programs
- Negotiating session limits with partner LUs
- BIND image and response
- Allocating and deallocating conversations
- FMH-5 and PIP data
- Conversation states
- Sending and receiving data
- Using high performance data transfer (HPDT)
- Session- and conversation-level security and data encryption
- Register usage
- Sync point services
- LU 6.2 global variables
- Vector lists
- Sense codes for FMH-7 and UNBIND
- RCPRI,RCSEC codes
- User exit routines.

OS/390 eNetwork Communications Server: SNA Programmers LU 6.2 Reference (SC31-8568). This book provides reference material for the VTAM LU 6.2 programming interface for host application programs. The information includes:

- APPCCMD macroinstructions
- Primary and secondary return codes (RCPRI, RCSEC)
- DSECTs
- Examples of using VTAM's LU 6.2 API

- Register usage

OS/390 eNetwork Communications Server: CSM Guide (SC31-8575). This book describes how applications use the communications storage manager. The information includes:

- Creating and deleting buffer pools
- Obtaining and freeing buffers
- Return codes and reason codes
- DSECTs

OS/390 eNetwork Communications Server: CMIP Services and Topology Agent Guide (SC31-8576). This book describes the Common Management Information Protocol (CMIP) programming interface for application programmers to use in coding CMIP application programs. The book provides guide and reference information about CMIP services and the VTAM topology agent and includes the following topics:

- Management information base (MIB) API functions
- CMIP message strings
- Special CMIP message strings
- Read queue exit routine
- Sample CMIP application program
- VTAM resources as CMIP objects
- Naming conventions for objects
- VTAM resources and OSI states
- Attributes to object cross-reference
- ASN.1 syntax for CMIP messages
- GDMO table format
- ACYAPHDH header file.

Diagnosis

OS/390 eNetwork Communications Server: IP Diagnosis (SC31-8521). This book explains how to diagnose TCP/IP problems and how to determine whether a specific problem is in the TCP/IP product code. It explains how to gather information for and describe problems to the IBM Software Support Center.

OS/390 eNetwork Communications Server: SNA Diagnosis (LY43-0079). This book helps you identify a VTAM problem, classify it, and collect information about it before you call the IBM Support Center. The information collected includes traces, dumps, and other problem documentation. The information includes:

- Command syntax for running traces and collecting and analyzing dumps
- VIT entries
- Procedures for collecting documentation (VTAM, TSO)
- VTAM internal trace and VIT analysis tool
- FFST Probes
- Channel programs
- Flow diagrams
- Procedures for locating buffer pools
- CPCB operation codes
- Storage and control block ID codes
- Offset names and locations for VTAM buffer pools.

OS/390 eNetwork Communications Server: Data Areas Volume 1 (LY43-0111). This book describes VTAM data areas and can be used to read a VTAM dump. It is intended for IBM programming service representatives and customer personnel who are diagnosing problems with VTAM.

OS/390 eNetwork Communications Server: Data Areas Volume 2 (LY43-0112). This book describes VTAM data areas and can be used to read a VTAM dump. It is intended for IBM programming service representatives and customer personnel who are diagnosing problems with VTAM.

Messages and Codes

OS/390 eNetwork Communications Server: SNA Messages (SC31-8569). This book describes the following types of messages and other associated information:

- Messages:
  - ELM messages for logon manager
  - IKT messages for TSO/VTAM
  - IST messages for VTAM network operators
  - ISU messages for sockets-over-SNA
  - IVT messages for the communications storage manager
  - IUT messages
  - USS messages
- Other information that displays in VTAM messages:
  - Command and RU types in VTAM messages
  - Node and ID types in VTAM messages
- Supplemental message-related information:
  - Message additions, deletions, and changes
  - Message flooding prevention
  - Message groups and subgroups
  - Message routing and suppression including descriptor codes, routing codes, and suppression levels for ELM, IKT, IST, and ISU messages
  - Message text and description formats
  - Message text of MSGLVL option messages including general information on the MSGLVL option
  - Message text of all VTAM network operator messages including variable field lengths

OS/390 eNetwork Communications Server: IP Messages Volume 1 (SC31-8517). This volume contains TCP/IP messages beginning with EZA.

OS/390 eNetwork Communications Server: IP Messages Volume 2 (SC31-8570). This volume contains TCP/IP messages beginning with EZB.

OS/390 eNetwork Communications Server: IP Messages Volume 3 (SC31-8674). This volume contains TCP/IP messages beginning with EZY, EZZ, and SNM.

OS/390 eNetwork Communications Server: IP and SNA Codes (SC31-8571). This book describes codes and other information that display in CS/390 messages:

- Sense codes including VTAM sense code hints, SNA sense field values for RPL-based macroinstructions, and 3270 SNA and non-SNA device sense fields
- Return codes for macroinstructions including ACB OPEN and CLOSE macroinstruction error fields, RTNCD-FDB2 return code combinations, and LU 6.2 RCPRI-RCSEC return codes
- Data link control (DLC) status codes
- Status codes including resource status and session state codes
- Wait state event codes and IDs
- Abend codes
- ATM network-generated cause and diagnostic codes

APPC Application Suite

OS/390 eNetwork Communications Server: APPC Application Suite User's Guide (GC31-8619). This book documents the end-user interface (concepts, commands, and messages) for the AFTP, ANAME, and APING facilities of the APPC application suite. Although its primary audience is the end user, administrators and application programmers may also find it useful.

OS/390 eNetwork Communications Server: APPC Application Suite Administration (SC31-8620). This book contains the information that administrators need to configure the APPC application suite and to manage the APING, ANAME, AFTP, and A3270 servers.

OS/390 eNetwork Communications Server: APPC Application Suite Programming (SC31-8621). This book provides the information application programmers need to add the functions of the AFTP and ANAME APIs to their application programs.

Multiprotocol Transport Networking (MPTN) Architecture Publications

Following are selected publications for MPTN:

Networking Blueprint Executive Overview (GC31-7057)

Multiprotocol Transport Networking: Technical Overview (GC31-7073)

Multiprotocol Transport Networking: Formats (GC31-7074)

OS/390 Publications

For information on OS/390 and other products, refer to *OS/390 Information Roadmap* (GC28-1727-04).

Index

H

I

K

M

N

O

P

R

S

T

U

V

Communicating Your Comments to IBM

OS/390 eNetwork Communications Server
CSM Guide
Version 2 Release 5

Publication No. SC31-8575-00

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM. Whichever method you choose, make sure you send your name, address, and telephone number if you would like a reply.

Feel free to comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. However, the comments you send should pertain to only the information in this manual and the way in which the information is presented. To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer.

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

If you are mailing a readers' comment form (RCF) from a country other than the United States, you can give the RCF to the local IBM branch office or IBM representative for postage-paid mailing.

- If you prefer to send comments by mail, use the RCF at the back of this book.
- If you prefer to send comments by FAX, use this number:

  1-800-227-5088(US and Canada)
- If you prefer to send comments electronically, use this network ID:
  - USIB2HPD@VNET.IBM.COM
  - USIB2HPD at IBMMAIL

Make sure to include the following in your note:

- Title and publication number of this book
- Page number or topic to which your comment applies.

Readers' Comments — We'd Like to Hear from You

OS/390 eNetwork Communications Server
CSM Guide
Version 2 Release 5

Publication No. SC31-8575-00

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	□	□	□	□	□

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	□	□	□	□	□
Complete	□	□	□	□	□
Easy to find	□	□	□	□	□
Easy to understand	□	□	□	□	□
Well organized	□	□	□	□	□
Applicable to your tasks	□	□	□	□	□

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? □ Yes □ No

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

Name _____ Address _____

Company or Organization _____

Phone No. _____

Readers' Comments — We'd Like to Hear from You
SC31-8575-00

Cut or Fold
Along Line

Fold and Tape | **Please do not staple** | Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Information Development
Department CGMD / Bldg 500
P.O. Box 12195
Research Triangle Park, NC 27709-9990

Fold and Tape | **Please do not staple** | Fold and Tape

SC31-8575-00

Cut or Fold
Along Line

Program Number: 5647-A01

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC31-8575-00

Spine information:

OS/390 eNetwork Communications Server CSM Guide *Version 2 Release 5*