Contents

Figures ........................................ v
Tables ........................................ vii

About this document ............................ ix
Who should read this document ................. ix
How this document is organized .................. ix
How to use this document ....................... ix
  Determining whether a publication is current .. ix
Conventions and terminology that are used in this document ......................... x
How to read a syntax diagram .................. xi
Prerequisite and related information .......... xi

Summary of changes for CSM Guide ............ xxi
Changes made in z/OS Version 2 Release 2 ....... xxi
Summary of changes for z/OS Version 2 Release 1 . xxi

Chapter 1. Communications storage manager (CSM) overview ................... 1
  CSM application programming interface (API) ................................ 1
  Installing, defining, and initializing CSM ................................... 2
  Monitoring CSM storage ...................................................... 2
  Diagnosing CSM problems .................................................. 4
  Formatting CSM dump information .................. 5
  Application responsibilities for using CSM ..................... 5
  Functions provided by the CSM API ...................... 6
    Creating and registering buffer pools ................ 8
    Requesting storage from CSM ................................. 8
    CSM buffer lists ................................................. 8
    Fixed buffers versus pageable buffers .............. 9
    Ownership of buffers ..................................... 10
    Storage return ................................................. 12
    Clearing data from buffers ......................... 12
    Removing registration from a pool .................. 12
    Copying data to or from a CSM buffer .......... 13
    Sharing buffers among multiple users .......... 14
    Obtaining CSM dumping information .......... 15
    Obtaining CSM resource statistics .......... 16
    CSM buffer pool expansion and contraction .... 17
    Buffer return exit routine ................ 18
    CSM recovery for normal and abnormal termination .. 19

Chapter 2. CSM macroinstructions ................ 21
  How the macroinstructions are described ........... 21
  Computing environment for the CSM API .......... 22
    IVTCSM REQUEST=ASSIGN_BUFFER ................. 23
    IVTCSM REQUEST=CHANGE_OWNER .............. 29
    IVTCSM REQUEST=COPY_DATA ................. 34
    IVTCSM REQUEST=CREATE_POOL .......... 40
    IVTCSM REQUEST=DELETE_POOL .............. 46
    IVTCSM REQUEST=DUMP_INFO ................. 50
    IVTCSM REQUEST=FIX_BUFFER .............. 52
    IVTCSM REQUEST=FREE_BUFFER .......... 57
    IVTCSM REQUEST=GET_BUFFER .............. 63
Figures

1. Example of copy data buffer list and copy results ........................................ 14
Tables

1. Buffer pools in CSM .................................................. 1
2. Valid operands for IVTCSM macroinstruction ...................... 7
3. Default values for EXPBUF .......................................... 43
4. Default values for INITBUF ........................................... 43
5. Default values for MINFREE .......................................... 44
6. IVTCSM macroinstruction return and reason codes .............. 79
7. CSM buffer list format ............................................... 81
8. IVTDATSP DSECT layout ............................................. 83
About this document

This document helps customers write application programs that use the communications storage manager (CSM). It describes the CSM application programming interface (API). It also describes the IVTCSM macro and the programming techniques for using this macro. It is intended to be used by application programs using the high performance data transfer (HPDT) interface. See the z/OS Communications Server: SNA Programmer’s LU 6.2 Guide for information about how VTAM application programs use the HPDT interface.

Who should read this document

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

You should be familiar with programming concepts and programming VTAM applications, as described in z/OS Communications Server: SNA Programming before you use the macroinstructions described in this book.

You should be familiar with the following concepts:

- Systems network architecture
- Data communications
- LU 6.2 architecture

How this document is organized

This document is divided into the following sections:

- Chapter 1, “Communications storage manager (CSM) overview,” on page 1
- Chapter 2, “CSM macroinstructions,” on page 21
- Appendix A, “IVTCSM macroinstruction return and reason codes,” on page 79
- Appendix B, “CSM DSECTs,” on page 81
- Appendix C, “Related protocol specifications,” on page 85
- Appendix D, ”Architectural specifications,” on page 109
- Appendix E, “Accessibility,” on page 111

- “Notices” on page 115 contains notices and trademarks used in this document.
- “Bibliography” on page 125 contains descriptions of the documents in the z/OS Communications Server library.

How to use this document

To use this document, you should be familiar with System/390 or zSeries host.

Determining whether a publication is current

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

- At the end of a publication's order number there is a dash followed by two digits, often referred to as the dash level. A publication with a higher dash level is more current than one with a lower dash level. For example, in the publication order number GC28-1747-07, the dash level 07 means that the publication is more current than previous levels, such as 05 or 04.
- If a hardcopy publication and a softcopy publication have the same dash level, it is possible that the softcopy publication is more current than the hardcopy publication. Check the dates shown in the Summary of Changes. The softcopy publication might have a more recently dated Summary of Changes than the hardcopy publication.
- To compare softcopy publications, you can check the last 2 characters of the publication's file name (also called the book name). The higher the number, the more recent the publication. Also, next to the publication titles in the CD-ROM booklet and the readme files, there is an asterisk (*) that indicates whether a publication is new or changed.

How to contact IBM service


Most problems can be resolved at this website, where you can submit questions and problem reports electronically, and access a variety of diagnosis information.

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

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

If you would like to provide feedback on this publication, see “Communicating your comments to IBM” on page 133.

Conventions and terminology that are used in this document

Commands in this book that can be used in both TSO and z/OS UNIX environments use the following conventions:

- When describing how to use the command in a TSO environment, the command is presented in uppercase (for example, NETSTAT).
- When describing how to use the command in a z/OS UNIX environment, the command is presented in bold lowercase (for example, netstat).
- When referring to the command in a general way in text, the command is presented with an initial capital letter (for example, Netstat).

All the exit routines described in this document are installation-wide exit routines. The installation-wide exit routines also called installation-wide exits, exit routines, and exits throughout this document.

The TPF logon manager, although included with VTAM, is an application program; therefore, the logon manager is documented separately from VTAM.
Samples used in this book might not be updated for each release. Evaluate a sample carefully before applying it to your system.

**Note:** In this information, you might see the following Shared Memory Communications over Remote Direct Memory Access (SMC-R) terminology:

- RDMA network interface card (RNIC), which is used to refer to the IBM® 10GbE RoCE Express® feature.
- Shared RoCE environment, which means that the 10GbE RoCE Express feature operates on an IBM z13™ (z13) or later system, and that the feature can be used concurrently, or shared, by multiple operating system instances. The RoCE Express feature is considered to operate in a shared RoCE environment even if you use it with a single operating system instance.

For definitions of the terms and abbreviations that are used in this document, you can view the latest IBM terminology at [the IBM Terminology website](https://www.ibm.com/support/pages/node/3262683).

**Clarification of notes**

Information traditionally qualified as Notes is further qualified as follows:

- **Note** Supplemental detail
- **Tip** Offers shortcuts or alternative ways of performing an action; a hint
- **Guideline** Customary way to perform a procedure
- **Rule** Something you must do; limitations on your actions
- **Restriction** Indicates certain conditions are not supported; limitations on a product or facility
- **Requirement** Dependencies, prerequisites
- **Result** Indicates the outcome

**How to read a syntax diagram**

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 ( بصورة) 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 an operand or value you must spell exactly as shown. In this example, you must code **OPERAND**.

**Note:** VTAM and IP commands are not case sensitive. You can code them in uppercase or lowercase. If the operand is shown in both uppercase and lowercase, the uppercase portion is the abbreviation (for example, OPERand).
If an operand or value can be abbreviated, the abbreviation is described 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.
Notes:
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:

Prerequisite and related information

z/OS Communications Server function is described in the z/OS Communications Server library. Descriptions of those documents are listed in “Bibliography” on page 125, in the back of this document.

Required information

Before using this product, you should be familiar with TCP/IP, VTAM, MVS™, and UNIX System Services.

Softcopy information

Softcopy publications are available in the following collection.

<table>
<thead>
<tr>
<th>Titles</th>
<th>Order Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM System z Redbooks Collection</td>
<td>SK3T-7876</td>
<td>The IBM Redbooks® publications selected for this CD series are taken from the IBM Redbooks inventory of over 800 books. All the Redbooks publications that are of interest to the System z® platform professional are identified by their authors and are included in this collection. The System z subject areas range from e-business application development and enablement to hardware, networking, Linux, solutions, security, parallel sysplex, and many others. For more information about the Redbooks publications, see <a href="http://www-03.ibm.com/systems/z/os/zos/zfavorites/">http://www-03.ibm.com/systems/z/os/zos/zfavorites/</a></td>
</tr>
</tbody>
</table>

Other documents

This information explains how z/OS references information in other documents.

When possible, this information uses cross-document links that go directly to the topic in reference using shortened versions of the document title. For complete titles and order numbers of the documents for all products that are part of z/OS, see [z/OS Information Roadmap](http://www.ibm.com/support/docview.wss?rs=180&id=infoRoadmap). The Roadmap describes what level of documents are published with each release of z/OS Communications Server, and also describes each z/OS publication.
To find the complete z/OS library, visit the z/OS library in IBM Knowledge Center (www.ibm.com/support/knowledgecenter/SSLTBW/welcome).

Relevant RFCs are listed in an appendix of the IP documents. Architectural specifications for the SNA protocol are listed in an appendix of the SNA documents.

The following table lists documents that might be helpful to readers.

<table>
<thead>
<tr>
<th>Title</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNA Formats</td>
<td>GA27-3136</td>
</tr>
<tr>
<td>TCP/IP Tutorial and Technical Overview</td>
<td>GG24-3376</td>
</tr>
<tr>
<td>Understanding LDAP</td>
<td>SG24-4986</td>
</tr>
<tr>
<td>z/OS Cryptographic Services System SSL Programming</td>
<td>SC14-7495</td>
</tr>
<tr>
<td>z/OS IBM Tivoli Directory Server Administration and Use for z/OS</td>
<td>SC23-6788</td>
</tr>
<tr>
<td>z/OS JES2 Initialization and Tuning Guide</td>
<td>SA32-0991</td>
</tr>
<tr>
<td>z/OS Problem Management</td>
<td>SC23-6844</td>
</tr>
<tr>
<td>z/OS MVS Diagnosis: Reference</td>
<td>GA32-0904</td>
</tr>
<tr>
<td>z/OS MVS Diagnosis: Tools and Service Aids</td>
<td>GA32-0905</td>
</tr>
<tr>
<td>z/OS MVS Using the Subsystem Interface</td>
<td>SA38-0679</td>
</tr>
<tr>
<td>z/OS Program Directory</td>
<td>GI11-9848</td>
</tr>
<tr>
<td>z/OS UNIX System Services Command Reference</td>
<td>SA23-2280</td>
</tr>
<tr>
<td>z/OS UNIX System Services Planning</td>
<td>GA32-0884</td>
</tr>
<tr>
<td>z/OS UNIX System Services Programming: Assembler Callable Services Reference</td>
<td>SA23-2281</td>
</tr>
<tr>
<td>z/OS UNIX System Services User's Guide</td>
<td>SA23-2279</td>
</tr>
<tr>
<td>z/OS XL C/C++ Runtime Library Reference</td>
<td>SC14-7314</td>
</tr>
<tr>
<td>z/Enterprise System and System z10 OSA-Express Customer's Guide and Reference</td>
<td>SA22-7935</td>
</tr>
</tbody>
</table>

**Redbooks publications**

The following Redbooks publications might help you as you implement z/OS Communications Server.

<table>
<thead>
<tr>
<th>Title</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM z/OS V2R1 Communications Server TCP/IP Implementation, Volume 1: Base Functions, Connectivity, and Routing</td>
<td>SG24-8096</td>
</tr>
</tbody>
</table>
Where to find related information on the Internet

z/OS
This site provides information about z/OS Communications Server release availability, migration information, downloads, and links to information about z/OS technology
http://www.ibm.com/systems/z/os/zos/

z/OS Internet Library
Use this site to view and download z/OS Communications Server documentation
www.ibm.com/systems/z/os/zos/bkserv/

IBM Communications Server product
The primary home page for information about z/OS Communications Server

IBM Communications Server product support
Use this site to submit and track problems and search the z/OS Communications Server knowledge base for Technotes, FAQs, white papers, and other z/OS Communications Server information

IBM Communications Server performance information
This site contains links to the most recent Communications Server performance reports.
http://www.ibm.com/support/docview.wss?uid=swg27005524

IBM Systems Center publications
Use this site to view and order Redbooks publications, Redpapers™, and Technotes
IBM Systems Center flashes

Search the Technical Sales Library for Techdocs (including Flashes, presentations, Technotes, FAQs, white papers, Customer Support Plans, and Skills Transfer information)

http://www.ibm.com/support/techdocs/atsmastr.nsf

Tivoli® NetView® for z/OS

Use this site to view and download product documentation about Tivoli NetView for z/OS

http://www.ibm.com/support/knowledgecenter/SSZJDU/welcome

RFCs

Search for and view Request for Comments documents in this section of the Internet Engineering Task Force website, with links to the RFC repository and the IETF Working Groups web page

http://www.ietf.org/rfc.html

Internet drafts

View Internet-Drafts, which are working documents of the Internet Engineering Task Force (IETF) and other groups, in this section of the Internet Engineering Task Force website

http://www.ietf.org/ID.html

Information about web addresses can also be found in information APAR II11334.

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

DNS websites

For more information about DNS, see the following USENET news groups and mailing addresses:

USENET news groups
comp.protocols.dns.bind

BIND mailing lists
https://lists.isc.org/mailman/listinfo

BIND Users
- Subscribe by sending mail to bind-users-request@isc.org.
- Submit questions or answers to this forum by sending mail to bind-users@isc.org.

BIND 9 Users (This list might not be maintained indefinitely.)
- Subscribe by sending mail to bind9-users-request@isc.org.
- Submit questions or answers to this forum by sending mail to bind9-users@isc.org.

The z/OS Basic Skills Information Center

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

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

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

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

Summary of changes for CSM Guide

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

Changes made in z/OS Version 2 Release 2

This document contains information previously presented in z/OS Communications Server: CSM Guide, SC27-3647-00, which supported z/OS Version 2 Release 1.

Changed information

• For 64-bit enablement of the TCP/IP stack, see the following topics:
  – “Monitoring CSM storage” on page 2
  – “Formatting CSM dump information” on page 5
  – “Obtaining CSM dumping information” on page 15

Summary of changes for z/OS Version 2 Release 1

For specifics on the enhancements for z/OS Version 2, Release 1, see the following publications:

• z/OS Summary of Message and Interface Changes
• z/OS Introduction and Release Guide
• z/OS Planning for Installation
• z/OS Migration
Chapter 1. Communications storage manager (CSM) overview

This chapter describes CSM and its capabilities, and includes the following sections:

- "CSM application programming interface (API)"
- "Installing, defining, and initializing CSM" on page 2
- "Monitoring CSM storage" on page 2
- "Diagnosing CSM problems" on page 4
- "Application responsibilities for using CSM" on page 5
- "Functions provided by the CSM API" on page 6

The communications storage manager (CSM) is a VTAM component that allows authorized host applications to share data with VTAM and other CSM users without the need to physically copy the data.

CSM is provided as part of the high performance data transfer (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, the following central processing unit (CPU) resources are conserved:

- CPU cycles
- Memory bus
- Cache

CSM application programming interface (API)

CSM includes an application programming interface (API) that provides a way to:

- Obtain and return CSM buffers
- Change ownership of buffers
- Copy buffers
- Manage CSM buffers

Requirement: 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 as listed in Table 1.

<table>
<thead>
<tr>
<th>Storage types</th>
<th>Buffer sizes</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-bit-backed data space</td>
<td>4 KB, 16 KB, 32 KB, 60 KB, 180KB</td>
</tr>
<tr>
<td>64-bit-backed data space</td>
<td>4 KB, 16 KB, 32 KB, 60 KB, 180KB</td>
</tr>
<tr>
<td>ECSA</td>
<td>4 KB, 16 KB, 32 KB, 60 KB, 180KB</td>
</tr>
</tbody>
</table>
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 associates 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. When this occurs, 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 z/OS Communications Server: New Function Summary 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. After CSM is initialized, it persists for the life of the system.

**Monitoring CSM storage**

System operators can issue the DISPLAY CSM command to monitor CSM storage usage.

**Rule:** CSM messages always start with the message prefix IVT.

For a complete list of messages issued by CSM, see the z/OS Communications Server: SNA Messages.

The DISPLAY CSM command yields the following information:
- Amount of storage allocated to each pool
- Amount of storage allocated to each user of the pool
- If OWNERID=ALL is specified, the cumulative storage allocated to each user across all pools
- If OWNERID is not specified,
  - The highest level of fixed storage that is obtained since the last DISPLAY CSM command was issued without the OWNERID parameter.
  - The highest level of fixed storage that is obtained since the IPL.
  - The highest level of ECSA storage that is obtained since the last DISPLAY CSM command was issued without the OWNERID parameter.
  - The highest level of ECSA storage that is obtained since the IPL.
- The highest level of HVCOMM storage that is obtained since the last
  DISPLAY CSM command was issued without the OWNERID parameter.
- The highest level of HVCOMM storage that is obtained since the IPL.
- The names of CSM data spaces.

- The maximum amount of fixed, ECSA, and HVCOMM storage that can be
  allocated by CSM and current values of fixed, ECSA, and HVCOMM storage.

Use the DISPLAY CSM command to identify a user of the pool that is consuming
inordinate amounts of storage. This can happen if an application fails to free
buffers that it obtained from CSM. The report of storage allocated to a user is
based on the value of the user's owner_ID parameter. This is the OWNERID
operand on the DISPLAY CSM command. CSM uses the application's address
space identifier (ASID) as the OWNERID.

In some cases, the sum of the total of the storage allocated to all users of a pool
might be greater than the total amount of storage allocated to a pool. This is
cased by multiple buffer owners, which is a result from the creation of shared
instances using the [TVTCMS REQUEST=ASSIGN_BUFFER on page 23]
macroinstruction. The OWNERID information indicates the amount of storage that
the user must free to enable the storage to be returned to the buffer pool. Issue the
MODIFY CSM command to increase or decrease CSM storage limits without
requiring a re-IPL.

The CSM monitor function is available to monitor CSM buffers between many
components of z/OS Communications Server. IBM Software Support uses this
function to help diagnose CSM storage problems.

This function is controlled using the Modify CSM command with the MONITOR
operand. Valid options are MONITOR=YES, MONITOR=NO and
MONITOR=DYNAMIC. If you choose MONITOR=DYNAMIC, CSM buffer
monitoring is dynamically activated and deactivated. CSM dynamically activates
CSM buffer monitoring when CSM storage usage reaches 80% of defined limits of
the fixed storage or ECSA storage. It dynamically deactivates CSM buffer
monitoring when CSM storage usage returns to 75% of defined limits of the fixed
storage and ECSA storage. Use the Display CSM command with the MONITOR
operand to display the function status.

Use the DISPLAY CSMUSE command to evaluate the use of storage managed by
CSM. Although this command is similar to the DISPLAY CSM command, it
provides a lower level of detail for storage usage; therefore, this command is
different than that of the DISPLAY CSM command. This command is primarily
intended for IBM service. However, it can also be beneficial to the user. The
display output provides detailed information about each CSM storage pool. The
detailed information describes storage as it corresponds to an identifier, which is
referred to as a monitor ID. Monitor IDs describe specific z/OS Communications
Server components. When CSM storage is associated with (or isolated to) a specific
monitor ID, then IBM service can correlate the monitor ID to a component (usage
or function) of z/OS Communications Server. This information can be useful when
evaluating how z/OS Communications Server is using system storage or to help
diagnose storage growth. See the [z/OS Communications Server: IP and SNA Codes]
for the complete description of monitor IDs. The critical level storage usage is 90% or
higher of ECSA MAX, FIXED MAX, or HVCOMM MAX values specified in
CSM parmlib IVTPRM00. If the trend of CSM storage usage is approaching 85% or
higher of ECSA MAX, FIXED MAX, or HVCOMM MAX storage values, CSM
indicates a constrained level of storage usage. The normal level storage usage is 80% or below of ECSA MAX, FIXED MAX, or HVCOMM MAX storage values.

CSM issues messages when CSM storage limits are at a constrained level, critical level or when limits are exceeded. In this case, the system operator can issue the MODIFY CSM command to increase the amount of fixed, ECSA or HVCOMM storage available for CSM.

For more information about these CSM commands, see the z/OS Communications Server: SNA Operation.

Use the DISPLAY TRL command to isolate a storage problem to a specific device. For more information, see z/OS Communications Server: SNA Diagnosis Vol 1, Techniques and Procedures. For more detail on the DISPLAY TRL command, see the z/OS Communications Server: SNA 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 the z/OS 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" on page 74 macroinstruction.

---

**Diagnosing CSM problems**

Use the CSM option on the VTAM internal trace (VIT) or, when VTAM is not active, the GTF trace facility to obtain trace output of application requests to CSM.

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 controls 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 varies based on the number of buffer operations requested. Because the information required to trace one IVTCSM request can 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 to on, the trace record is continued. If VIT is not active, multiple trace records for an IVTCSM request could be interspersed with trace records of IVTCSM requests from other users. A unique trace record number correlates the continuation trace records for each IVTCSM request.

For more information about tracing events over the CSM API, see z/OS Communications Server: SNA Diagnosis Vol 1, Techniques and Procedures.
Formatting CSM dump information

Interactive Problem Control System (IPCS) dump formatters provide the following services for displaying CSM information in a dump:

- Find and display CSM data structures including all pools and their extents.
- Find and display CSM data structures for a buffer pool based on the size and source, ECSA, data space, and HVCOMM storage.
- Find and display a buffer based on the input buffer token.
- Find all buffers based on an input OWNERID.
- Find all buffers based on input COMPID with the OWNERID or without OWNERID for all CSM pools.
- Display the summary of all COMPIDs for all CSM pools. For more information, see techniques and procedures information in z/OS Communications Server: SNA Diagnosis Vol 1, Techniques and Procedures.

All information is displayed with a header that identifies the contents followed by hexadecimal contents only; no field identification is provided.

Restriction: Formatting options that display information in buffers can provide the requested data only when the required storage areas are included in the dump.

Application responsibilities for using CSM

Requirement: An application must be authorized in order to use the CSM API. This is described in z/OS Communications Server: SNA Programming.

As system-authorized applications, all programs using CSM should be written to 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 see the guidelines described in z/OS Communications Server: SNA Programmer's LU 6.2 Guide.

Guidelines:

- Data in CSM storage should be modified only by the original requester of the buffers. The original requester is the application that obtained the storage using the "IVTCSM REQUEST=GET_BUFFER" on page 63 macroinstruction. All other applications are considered borrowers of the buffers and must treat the data as read-only. For information about 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" on page 34 macroinstruction allows data to be copied into or out of CSM storage. The authorized invoker can be in any key. Using this service might 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" on page 57 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" for more information. The RESOURCE_STATS request is described on page 74.

Generally, 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 CSM user needs in order to create and use storage pools. The IVTCSM macroinstruction provides the interface for applications issuing requests to CSM.

Use the IVTCSM macroinstruction to issue the following types of requests:

- "IVTCSM REQUEST=ASSIGN_BUFFER" on page 23
- "IVTCSM REQUEST=CHANGE_OWNER" on page 29
- "IVTCSM REQUEST=CREATE_POOL" on page 40
- "IVTCSM REQUEST=DELETE_POOL" on page 46
- "IVTCSM REQUEST=DUMP_INFO" on page 50
- "IVTCSM REQUEST=FREE_BUFFER" on page 57
- "IVTCSM REQUEST=GET_BUFFER" on page 63
- "IVTCSM REQUEST=PAGE_BUFFER" on page 69
- "IVTCSM REQUEST=RESOURCE_STATS" on page 74

See Chapter 2, “CSM macroinstructions,” for the complete description and syntax of each request.

Table 2 contains a cross reference of IVTCSM requests and their valid input and output parameters.
### Table 2. Valid operands for IVTCSM macroinstruction

<table>
<thead>
<tr>
<th>REQUEST</th>
<th>CREATEPOOL</th>
<th>DELETEPOOL</th>
<th>GETBUFFER</th>
<th>FREEBUFFER</th>
<th>ASSIGNBUFFER</th>
<th>CHANGEOWNER</th>
<th>FIXBUFFER</th>
<th>PAGEBUFFER</th>
<th>COPYDATA</th>
<th>DUMPINFO</th>
<th>RESOURCES</th>
<th>STATS</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLISTVER</td>
<td>Oi</td>
<td>Oi</td>
<td>Oi</td>
<td>Oi</td>
<td>Oi</td>
<td>Oi</td>
<td>Oi</td>
<td>Oi</td>
<td>Oi</td>
<td>Oi</td>
<td>Oi</td>
<td>Oi</td>
</tr>
<tr>
<td>BACK</td>
<td>Oi</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BUFLIST</td>
<td></td>
<td>Ri</td>
<td>Ri</td>
<td>Ri</td>
<td>Ri</td>
<td>Ri</td>
<td>Ri</td>
<td>Ri</td>
<td>Ri</td>
<td>Ri</td>
<td>Ri</td>
<td></td>
</tr>
<tr>
<td>BUFSIZE</td>
<td></td>
<td></td>
<td>Ri</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BUFSIZERC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BUFTYPE</td>
<td></td>
<td>Ri</td>
<td>Ri</td>
<td>Oi</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLEAR</td>
<td></td>
<td>Oi</td>
<td>Oi</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DS_INFO</td>
<td></td>
<td>Oo</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Oo</td>
</tr>
<tr>
<td>ERRBFLIST</td>
<td></td>
<td>Oo</td>
<td>Oo</td>
<td>Oo</td>
<td>Oo</td>
<td>Oo</td>
<td>Oo</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXPBUF</td>
<td></td>
<td>Ri</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FREERTN</td>
<td></td>
<td>Oi</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FREETO</td>
<td></td>
<td>Oi</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GAP</td>
<td></td>
<td>Oi</td>
<td>Oi</td>
<td>Oi</td>
<td>Oi</td>
<td>Oi</td>
<td>Oi</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INITBUF</td>
<td></td>
<td>Oi</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MF</td>
<td>Ri</td>
<td>Ri</td>
<td>Ri</td>
<td>Ri</td>
<td>Ri</td>
<td>Ri</td>
<td>Ri</td>
<td>Ri</td>
<td>Ri</td>
<td>Ri</td>
<td>Ri</td>
<td>Ri</td>
</tr>
<tr>
<td>MINFREE</td>
<td></td>
<td>Ri</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OWNERID</td>
<td></td>
<td>Oi</td>
<td>Oi</td>
<td>Oi</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PAD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Oi</td>
</tr>
<tr>
<td>POOLTOKN</td>
<td></td>
<td>Ri</td>
<td>Ri</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PADCHAR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RETCODE</td>
<td></td>
<td>Oo</td>
<td>Oo</td>
<td>Oo</td>
<td>Oo</td>
<td>Oo</td>
<td>Oo</td>
<td>Oo</td>
<td>Oo</td>
<td>Oo</td>
<td>Oo</td>
<td></td>
</tr>
<tr>
<td>RETPTOKN</td>
<td></td>
<td>Oo</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RSNCODE</td>
<td></td>
<td>Oo</td>
<td>Oo</td>
<td>Oo</td>
<td>Oo</td>
<td>Oo</td>
<td>Oo</td>
<td>Oo</td>
<td>Oo</td>
<td>Oo</td>
<td>Oo</td>
<td></td>
</tr>
<tr>
<td>SKIPBUF</td>
<td></td>
<td>Oi</td>
<td>Oi</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SRCERRL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Oo</td>
</tr>
<tr>
<td>SRCGAP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Oi</td>
</tr>
<tr>
<td>SRCLIST</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Ri</td>
</tr>
<tr>
<td>SRCNUM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Ri</td>
</tr>
<tr>
<td>STATAREA</td>
<td></td>
<td>Oo</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Oo</td>
</tr>
</tbody>
</table>
Table 2. Valid operands for IVTCSM macroinstruction (continued)

<table>
<thead>
<tr>
<th>REQUEST</th>
<th>CREAT POOL</th>
<th>DELE POOL</th>
<th>GET BUFFER</th>
<th>FREE BUFFER</th>
<th>ASSIGN BUFFER</th>
<th>CHAN OWNER</th>
<th>FIX BUFFER</th>
<th>PAGE BUFFER</th>
<th>COPY DATA</th>
<th>DUMP INFO</th>
<th>RES STATS</th>
</tr>
</thead>
<tbody>
<tr>
<td>TARGERRL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TARGGAP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TARGLIST</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TARGNUM</td>
<td>Ri</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TASKID</td>
<td>Oi</td>
<td>Oi</td>
<td>Oi</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>THREAD</td>
<td>Oi</td>
<td>Oi</td>
<td>Oi</td>
<td>Oi</td>
<td>Oi</td>
<td>Oi</td>
<td>Oi</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UTILRTN</td>
<td>Oi</td>
<td>Oi</td>
<td>Oi</td>
<td>Oi</td>
<td>Oi</td>
<td>Oi</td>
<td>Oi</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WAIT</td>
<td>Oi</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Creating and registering buffer pools

Upon application request, CSM can create buffer pools based on the size and types listed in [Table 1 on page 1]. A total of 15 CSM buffer pools can be created, one for each storage type and buffer size.

The structures that maintain the storage pools are created as a result of the first IVTCSM REQUEST=CREATE_POOL macroinstruction issued by a CSM user (this could be V1AM or a host application). The pool might or might 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, which the application uses on subsequent requests buffer requests 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.

**Guideline:** Where possible, programs should use CSM data space instead of ECSA. Furthermore, use of 64-bit backed CSM data space is preferred to 31-bit backed CSM data space. Use data space instead of ECSA and 64-bit backed data space instead of 31-bit backed data space to benefit overall system performance.

Requesting storage from CSM

**Requirement:** Before an application can retrieve storage from CSM, it must be registered as a user of a CSM buffer pool. See “Creating and registering buffer pools” for a description.
Table 1 on page 1 lists a summary of the available CSM buffer pools. CSM returns a pool token in the RETPTOKN parameter, which the application uses on subsequent buffer requests 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 after they are freed by a user.

Buffer return exit routines can receive control after buffers are released by a FREE_BUFFER request. See “Buffer return exit routine” on page 18 for more information.

Restriction: GET_BUFFER requests that exceed the storage available in CSM are rejected. CSM users should monitor CSM storage use and take action to prevent critical shortages that might jeopardize the application or system operation. See “Obtaining CSM resource statistics” on page 16 for more information.

CSM buffer lists

The following requests require that the application provide the buffer list address:

- ‘IVTCSM REQUEST=ASSIGN_BUFFER’ on page 23
- ‘IVTCSM REQUEST=CHANGE_OWNER’ on page 29
- ‘IVTCSM REQUEST=FIX_BUFFER’ on page 52
- ‘IVTCSM REQUEST=FREE_BUFFER’ on page 57
- ‘IVTCSM REQUEST=GET_BUFFER’ on page 63
- ‘IVTCSM REQUEST=PAGE_BUFFER’ on page 69

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 81. 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.

Tip: Each buffer list entry can be contiguous to the previous entry with the number of entries in the list defined by the BUFNUM operand on the request. Use the GAP parameter to separate entries.

Fixed buffers versus pageable buffers

Applications can specify buffers 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 fixed into real storage. 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 use the FIX_BUFFER request to make the buffers fixed at a later time.

Guaranteeing that a buffer is pageable
An application can specify BUFTYPE=PAGEABLE on the GET_BUFFER and PAGE_BUFFER requests to classify buffers as guaranteed to be made pageable.

Restriction: This function can be issued for a buffer consisting of only one image. For information about creating multiple images of a buffer, see “IVTCSM REQUEST=ASSIGN_BUFFER” on page 23.

Making a buffer eligible to be paged
An application can specify BUFTYPE=PAGEELIG on the GET_BUFFER, ASSIGN_BUFFER, and PAGE_BUFFER requests in order to classify buffers as eligible to be paged. CSM can maintain a status of eligible to be paged. 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 can remain fixed unless CSM determines that the storage should be made pageable. This avoids the unnecessary overhead of fixing and freeing storage from a system perspective.

This function avoids consuming fixed storage for data that is being held in a buffer for possible use at a later time.

Requirement: If an eligible to be paged buffer must be fixed later, the application must issue the FIX_BUFFER request.

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” on page 23
- “IVTCSM REQUEST=GET_BUFFER” on page 63
- “IVTCSM REQUEST=CHANGE_OWNER” on page 29

The owner is the application responsible for returning buffers to CSM using the “IVTCSM REQUEST=FREE_BUFFER” on page 57 macroinstruction. CSM assigns initial buffer ownership to the original requester of the buffers. To override this assignment, specify another user’s ASID as the OWNERID on the “IVTCSM REQUEST=GET_BUFFER” on page 63 macroinstruction. To pass on ownership of CSM buffers to another, issue an “IVTCSM REQUEST=CHANGE_OWNER” on page 29 macroinstruction.

To associate ownership of a buffer to a task, specify a TCB address on the TASKID parameter on the following requests:

- “IVTCSM REQUEST=ASSIGN_BUFFER” on page 23
- “IVTCSM REQUEST=GET_BUFFER” on page 63
- “IVTCSM REQUEST=CHANGE_OWNER” on page 29

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 is 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" on page 63 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" on page 57 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.

Guideline: The receiving application should consider this as read-only storage and should not modify the contents.

If written as a cooperative set of processes, applications might determine that it is acceptable to modify the data if the original application does not require the original data returned unmodified.

Restriction: 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

Any user that can address the buffers by using the buffer tokens provided by CSM can issue the CHANGE_OWNER request. This includes the following scenarios:

- Application A passes buffer tokens to application B. Application B issues the "IVTCSM REQUEST=CHANGE_OWNER" on page 29 macroinstruction.
- Application A uses the "IVTCSM REQUEST=CHANGE_OWNER" on page 29 macroinstruction to pass ownership of the buffers to application B.

Restriction: 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" on page 57 macroinstruction.

Example:

The High Performance data transfer (HPDT) interface demonstrates how two CSM users change ownerships.

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](page 57) macroinstruction to release buffers back to CSM.

CSM returns the buffers to the original requester if all of the following conditions exist:

- The original requester specified a buffer return exit address on the FREERTN parameter of the [IVTCSM REQUEST=GET_BUFFER](page 63) macroinstruction.
- The original requester is active at the time the [IVTCSM REQUEST=FREE_BUFFER](page 57) macroinstruction is issued.
- The application issuing [IVTCSM REQUEST=FREE_BUFFER](page 57) 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.

Optionally, the requester can specify that the buffer is to be cleared when issuing the FREE_BUFFER request. See [“Clearing data from buffers”](page 57) for more information.

**Requirement:** 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 ensures 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](page 57)
- [IVTCSM REQUEST=GET_BUFFER](page 63)

This passes secure data to another user such that any residual data is eliminated when that buffer has returned to the pool.

**Notes:**

- Specifying CLEAR=YES on a [IVTCSM REQUEST=GET_BUFFER](page 63) macroinstruction overrides a CLEAR=NO specification on a [IVTCSM REQUEST=FREE_BUFFER](page 57) macroinstruction.
- Specifying CLEAR=YES does 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**

Because each pool can 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](page 46) macroinstruction.
Copying data to or from a CSM buffer

Applications can use the “IVTCSM REQUEST=COPY_DATA” on page 34 macroinstruction to copy data to or from a CSM buffer or a user data area. The authorized invoker can be in any key. This request might reduce the impact to the application by reducing storage key mismatches when CSM storage must be accessed. It also assists 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” on page 34 macroinstruction allows multiple source buffers to be copied to or from one or more 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, the source data is truncated. 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. This is shown in Figure 1 on page 14. The number of entries in each list does not have to be equal. Within each list, entries might or might 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 correspond 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.

Optionally, a user data area that is involved in the copy data operation can be access list entry token (ALET)-qualified to allow this area to reside in a data space.
Sharing buffers among multiple users

The "IVTCSM REQUEST=ASSIGN_BUFFER" on page 23 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 function allows specific areas of the buffer to be allocated to different owners. This function can be used to give multiple users read access to the same data.

Tip: No serialization is provided to prevent concurrent updates by users.

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

Requirement: 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.

Restriction: A request to create a new image of a buffer that is in a guaranteed to be pageable state is not permitted. 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 regardless of the
fact that the user might be interested in only a piece of the buffer. The application can specify the BUFTYPE parameter as described in “Fixed buffers versus pageable buffers” on page 9.

Obtaining CSM dumping information

To debug problems associated with the application’s use of CSM buffers, it might be necessary to include CSM storage in a dump.

In order to include CSM storage in a dump, an application can use the following information:

- **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 4 KB - 2 GB
  - Data space STOKENs and ALETs provided by IVTCSM REQUEST=DUMP_INFO.

- **Dumping buffers in a high virtual common (HVCOMM) storage above the bar buffer pool:**
  - Dump of common storage area (CSA) is required. When you dump CSA, CSA will include CSM data structures, pool structures, and buffer headers and buffers in ECSA buffer pools.

  **Tip:** If you dump the buffers in HVCOMM storage pools, CSA will include CSM data structures, pool structures, and buffer headers and buffers in ECSA buffer pools.

  **Guideline:** Applications should 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 GB contents.

For ease of dumping CSM buffers during testing, an application can use ECSA buffers because this area can be included in a dump using the subpool and key. After the application is debugged, data space buffers can 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” on page 40
- “IVTCSM REQUEST=DUMP_INFO” on page 50

The application can perform the following actions:

- Specify the address of an area on the DS_INFO parameter where CSM places 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 83.
- Request the information during initialization processing and use the information throughout normal processing.

Dumping the entire data space can require a long processing time and a large amount of external recording media. Because of this, you might want to limit the amount of area dumped based on address ranges within the data space believed to be pertinent to the user. For example, use the ALETs returned by the IVTCSM REQUEST=DUMP_INFO macroinstruction to 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 can 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 the information that is 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” on page 40
- “IVTCSM REQUEST=RESOURCE_STATS” on page 74

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, the following bits are defined:

**Critical**
Indicates that CSM storage usage is at 90% of defined limits or higher.

**Constrained**
Indicates that CSM storage usage is higher than 80% of defined limits and is approaching the constrained level (85% of defined limits). If this bit is set, the application should determine if the critical bit is also set.

**Note:** If neither bit is set, the resource usage is considered to be normal.

**Restriction:** GET_BUFFER requests that exceed the storage available in CSM are rejected.

CSM users should monitor CSM storage use and take action to prevent critical shortages that might jeopardize the application or system operation. Actions might include:

- Freeing buffers that are no longer needed
Selecting a different storage source for buffer pools

Limiting usage of fixed storage

IVTSTATA DSECT maps CSM resource statistics information. See "CSM resource status area (IVTSTATA)" on page 83.

CSM buffer pool expansion and contraction

The MINFREE, INITBUF, and EXPBUF specifications in the CSM parmlib member, IVTPRM00, determine CSM buffer pools expansion and contraction. 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" on page 40 macroinstruction. If buffer pool tuning specifications are not available from either the CSM parmlib member or application request, system defaults are used.

This section describes how CSM buffer pool expansion and contraction are performed based on application settings. For more information about the CSM parmlib member, see the z/OS Communications Server: New Function Summary.

Number of buffers when buffer pool is created using application settings

When the first CREATE_POOL request is received, CSM creates a pool of buffers. The INITBUF parameter specifies the initial number of buffers created in that pool.

Requirement: INITBUF is required on the CREATE_POOL request.

The range for INITBUF is 0 - 9999. If 0 is specified, only the base pool structure is created. In this case, the pool is 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).

<table>
<thead>
<tr>
<th>Pool size</th>
<th>Initial number of buffers (INITBUF)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4096</td>
<td>64</td>
</tr>
<tr>
<td>16384</td>
<td>32</td>
</tr>
<tr>
<td>32768</td>
<td>16</td>
</tr>
<tr>
<td>61440</td>
<td>16</td>
</tr>
<tr>
<td>184320</td>
<td>2</td>
</tr>
</tbody>
</table>

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 is 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" on page 63 macroinstruction due to inline pool expansion. In the event that pool expansion must complete to satisfy a
request for buffers, the application can optionally wait for pool expansion to complete by specifying WAIT=EXPAND or WAIT=YES on the "IVTCSM REQUEST=GET_BUFFER" on page 63 macroinstruction.

MINFREE parameter:

**Requirement:** 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 buffer sizes in the pool).

<table>
<thead>
<tr>
<th>Pool size</th>
<th>Minimum buffers that are free (MINFREE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4096</td>
<td>8</td>
</tr>
<tr>
<td>16384</td>
<td>4</td>
</tr>
<tr>
<td>32768</td>
<td>2</td>
</tr>
<tr>
<td>61440</td>
<td>2</td>
</tr>
<tr>
<td>184320</td>
<td>1</td>
</tr>
</tbody>
</table>

EXPBUF parameter:

**Requirement:** 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.

<table>
<thead>
<tr>
<th>Pool size</th>
<th>Valid range</th>
<th>Number of buffers to expand pool (EXPBUF)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4096</td>
<td>1 - 256</td>
<td>16</td>
</tr>
<tr>
<td>16384</td>
<td>1 - 256</td>
<td>8</td>
</tr>
<tr>
<td>32768</td>
<td>1 - 128</td>
<td>4</td>
</tr>
<tr>
<td>61440</td>
<td>1 - 68</td>
<td>4</td>
</tr>
<tr>
<td>184320</td>
<td>1 - 22</td>
<td>2</td>
</tr>
</tbody>
</table>

**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 does not contract when the number of buffers is 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 after they are freed by a user. By default, buffers are returned to CSM when the current owner issues the "IVTCSM REQUEST=FREE_BUFFER" on page 57.
macroinstruction. An application that provides the address of its buffer return exit
on the FREERTN parameter of the "IVTCSM REQUEST=GET_BUFFER" on page 63
macroinstruction receives 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 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. For 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 sample output shows 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 might 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 FREE_BUFFER
request. This 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 might not always be possible for
abnormal termination.

To ensure that buffers are not lost as a result of 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 cleanup is performed if task is not an MVS started task (for
example, batch).

**Memory Resource Termination Manager**

Memory cleanup is performed whenever address space memory
termination occurs.

The resource managers perform the following actions:
• 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
• 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 are 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 CSM user wants 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, users must ensure that their application processing is properly synchronized with any applications to which they are passing buffers.
Chapter 2. CSM macroinstructions

This chapter describes all varieties of the IVTCSM macroinstruction and includes the following sections:

- "How the macroinstructions are described"
- "Computing environment for the CSM API" on page 22

Separate descriptions are included for each value of the REQUEST parameter. Macroinstruction descriptions are arranged alphabetically.

How the macroinstructions are described

Each macroinstruction description includes the following information:

- 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 uses the following format:

```
main diagram
```

The macroinstructions are coded in the same format as assembler instructions, using name, operation, and operand fields. See the IBM High Level Assembler Language Reference for MVS and VM for complete information about 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 uppercase bold type must be coded as they appear in the syntax. For parameter values that are shown in lowercase 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.

Tip: 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, \texttt{MF=\{\ldots\}}, to define the parameter list structure. The \texttt{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_Xopername. If macroinstruction-defined values are associated with opername, the constant for a particular value, valuenam, is listaddr_Xopername_valuenam.

**Restriction:** 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.

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 API**

This section describes the environment where the IVTCSM macro is issued.

**Environment**

The following requirements are for the caller:

**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 can 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 provides for buffer reclamation at the following tasks:

- End-of-memory
• End-of-Job-Step-Task
• Optionally, at abnormal end-of-task

See “CSM recovery for normal and abnormal termination” on page 19 for more information.

Restrictions

The caller has the following restrictions:
• 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 the following registers:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Error reason code from the requested function</td>
</tr>
<tr>
<td>1</td>
<td>Used as work register by the system</td>
</tr>
<tr>
<td>2-13</td>
<td>Unchanged</td>
</tr>
<tr>
<td>14</td>
<td>Used as work register by the system</td>
</tr>
<tr>
<td>15</td>
<td>Return code from requested function</td>
</tr>
</tbody>
</table>

**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 can be optionally qualified by specifying a TASKID on the macroinstruction. The TASKID is a TCB address with the default being no task association.

Upon 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**

```
IVTCSM REQUEST — ASSIGN_BUFFER

parameters-1

- RETCODE — retcode
- RSNCODE — rsncode

- PLISTVER — IMPLIED_VERSION
- PLISTVER — MAX
- PLISTVER — 0

- MF — S

- MF — (L — list addr — attr)
- MF — (E — list addr — COMPLETE)
- MF — (M — list addr — NOCHECK)
```
Parameters

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: The buffer token representing the new buffer image is returned in the BUFL_TOKEN field as output.

The following fields in IVTBUFL are required 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=
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 is 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=**
An optional input parameter that specifies the macro form.

**MF=S**  Specifies 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.

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

**MF=E**  Specifies 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.

**MF=M**  Use 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.

**,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 input string 1 - 60 characters in length 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.

**Guidelines:** 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.

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

**IMPLIED_VERSION**
- The lowest version that allows all parameters specified on the request
to be processed. If you omit the PLISTVER parameter,
IMPLIED_VERSION is the default.

**MAX**
- Code this if you want the parameter list to be the largest size currently
possible. This size might increase from release to release and affect the
amount of storage that your program needs.

If you can tolerate the size change, you should 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**
- Code this if you use the currently available parameters.

To code, specify one of the following values:
- 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.
   This can be achieved 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 relevant only 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 be initialized.

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

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

BUFTYPE value specified is not valid for this request.

Assign buffer request failed because CSM reached the maximum number of image buffers of the single CSM buffer.

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

<table>
<thead>
<tr>
<th>Reason code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Unable to obtain storage for request.</td>
</tr>
<tr>
<td>6</td>
<td>An abend occurred while processing this request.</td>
</tr>
</tbody>
</table>

**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 can 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 is 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**
**Parameters**

**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 for this request.
• BUFL_VERSION
• BUFL_SOURCE

Rule: This field is required only when SKIPBUF=YES is specified.
• BUFL_TOKEN

No fields in IVTBUFL 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.

\(,\text{BUFNUM} = \text{bufnum}\)
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.

\(,\text{ERRBFLST} = \text{errbflst}\)
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.

\(,\text{GAP} = \text{gap}\)
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.

\(,\text{MF} = \)
An optional input parameter that specifies the macro form.

\(\text{MF=S}\) Specifies 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.

\(\text{MF=L}\) Specifies the list form of the macro. Use the list form together with the execute form of the macro for applications that require reentrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

\(\text{MF=E}\) Specifies 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.

\(\text{MF=M}\)
Use 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.
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 input string 1-60 characters in length that forces 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.

Guidelines: 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.

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

IMPLIED_VERSION
The lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED_VERSION is the default.

MAX
Code this if you want the parameter list to be the largest size currently possible. This size might increase from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, it is recommended 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
Code this if you use the currently available parameters.

To code, specify one of the following values:
- 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=
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 are 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) is 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.
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 relevant only 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

<table>
<thead>
<tr>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
<tr>
<td>4</td>
</tr>
</tbody>
</table>

### Reason code

<table>
<thead>
<tr>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>24</td>
</tr>
</tbody>
</table>

**IVTCSM REQUEST=COPY_DATA**

**Purpose**

Use this macroinstruction 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, the source data is truncated. 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 on page “CSM buffer list entry (IVTBUFL)” on page 81**. The number of entries in each list are not required to be equal. Within each list, entries might or might 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

```
IVTCSM REQUEST —— COPY_DATA

parameters-1

, RETCODE —— retcode
, RSNCODE —— rsncode

, PLISTVER —— IMPLIED_VERSION
, PLISTVER —— MAX
, PLISTVER —— 0

, MF —— S

, MF —— (L, list addr)

, MF —— (E, list addr)

, MF —— (M, list addr)
```

Chapter 2. CSM macroinstructions 35
Parameters

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=
An optional input parameter that specifies the macro form.

MF=S 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.

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

MF=E 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.

MF=M Use 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.

,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 input string 1 - 60 characters in length 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.

Guidelines:
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.

PAD=
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=
An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form.

When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms.

The values are:

IMPLIED_VERSION
The lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED_VERSION is the default.

MAX
Specify when you want the parameter list to be the largest size currently possible. This size might increase from release to release and affect the amount of storage that your program needs. If you can tolerate the size change, you should 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 Specify when you use the currently available parameters.

To code, specify one of the following values:
- IMPLIED_VERSION
- MAX
- A decimal value of 0

\texttt{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).

\texttt{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).

\texttt{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.

\texttt{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 are in contiguous storage.

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

\texttt{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 can 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 (Required only if data is being copied from a CSM buffer.)
- BUFL_ALET (Required only if 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.

\texttt{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. 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 are 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 can 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 (Required only if data is being copied into a CSM buffer.)
- BUFL_ALET (Required only if required to copy data into a user data space.)
- BUFL_ADDR
- BUFL_SIZE

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

To code, specify the RS-type address, or address in register (2)-(12), of a fullword 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 relevant only 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  
BUFLSOURCE value is not valid for an entry in the Source buffer list (SRCLIST).

19  
BUFLSOURCE 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**

Use this macroinstruction to allow the user to register as a user of a storage pool of buffers residing in ECSA or 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 can exist per data space. Data space storage pools are further qualified as 31-bit backed (when fixed, the real storage frame containing the page is below the 2-gigabyte real storage bar) or 64-bit backed (when fixed, the real storage frame containing the page can be on or above the 2-gigabyte real storage bar). If a 64-bit backed pool is requested and cannot be
created because the machine is not executing in z/Architecture® mode, the request is converted to a 31-bit backed request for the corresponding pool size.

On the create request, the caller specifies the size of the buffers in the pool to be created (4096, 16384, 32768, 61440, and 184320), and for dataspace pools, whether or not the pool is 31-bit backed or 64-bit backed. Only one pool of a given size exists per storage type. Requests by other callers for a pool of the same characteristics share the existing pool. The EXPBUF, INITBUF, and MINFREE values for each pool are each set to the largest value specified by any single user sharing a pool. Therefore, these values can also be adjusted downward when a user discontinues sharing a pool (that is, the user issues a DELETE_POOL request).

Usage

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

Environment

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

Syntax

main diagram

```plaintext
1VTCMS REQUEST---CREATE_POOL

parameters-1

, RETCODE = retcode
, RSNCODE = rsncode

, PLISTVER = IMPLIED_VERSION

, PLISTVER = MAX
, PLISTVER = 0

, MF = S

, MF = (L, list addr, attr, COMPLETE)
, MF = (E, list addr, NOCHECK, COMPLETE)
, MF = (M, list addr, NOCHECK)
```
Parameters

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.

,BACK=location
An optional parameter that applies to BUFSOURCE=DSPACE. Specifies whether or not the data space storage, when fixed, can be backed above the 2-gigabyte real storage bar. BACK=31 forces the storage to be backed below the 2-gigabyte bar and BACK=64 allows the storage to be backed on or above the 2-gigabyte bar. BACK=31 is the default. If a 64-bit backed pool is requested and cannot be created because the machine is not executing in z/Architecture mode, the request is converted to a 31-bit backed request for the corresponding pool size.

,BUFSIZE=bufsize
A required input parameter, specifying the size of the buffers in the pool to be created. Valid pool sizes are 4096, 16384, 32768, 61440, and 184320. All other values specified on this parameter are rounded up to the next valid pool size. However, if BUFSIZE is greater than 184320, the CREATE_POOL request is rejected.

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

,BUFSOURC=
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.

\texttt{,\texttt{EXPBUF}=\texttt{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 listed in Table 3. If a value outside of a range is specified, then CSM uses a default value. The default values for EXPBUF are also listed in Table 3.

\textbf{Table 3. Default values for EXPBUF}

<table>
<thead>
<tr>
<th>Pool size</th>
<th>Valid range</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>4096</td>
<td>1 - 256</td>
<td>16</td>
</tr>
<tr>
<td>16384</td>
<td>1 - 256</td>
<td>8</td>
</tr>
<tr>
<td>32768</td>
<td>1 - 128</td>
<td>4</td>
</tr>
<tr>
<td>61440</td>
<td>1 - 68</td>
<td>4</td>
</tr>
<tr>
<td>184320</td>
<td>1 - 22</td>
<td>2</td>
</tr>
</tbody>
</table>

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

\texttt{,\texttt{INITBUF}=\texttt{initbuf}}

A required input parameter, specifying the initial number of buffers to be created in the storage pool. If 0 is specified, the base pool is created only to represent the requester as a user of the pool. In this case, the pool is expanded on the first GET_BUFFER macroinstruction based on the specification for EXPBUF.

\textbf{Guideline:} The pool does not contract if the number of buffers currently available is below a certain value. The value is determined as the higher of INITBUF or MINFREE+(2*EXPBUF).

Valid values for INITBUF are in the range 0 - 9999. If a value outside of this range is specified, then CSM uses a default value. The default values for INITBUF are listed in Table 4.

\textbf{Table 4. Default values for INITBUF}

<table>
<thead>
<tr>
<th>Pool size</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>4096</td>
<td>64</td>
</tr>
<tr>
<td>16384</td>
<td>32</td>
</tr>
<tr>
<td>32768</td>
<td>16</td>
</tr>
<tr>
<td>61440</td>
<td>16</td>
</tr>
<tr>
<td>184320</td>
<td>2</td>
</tr>
</tbody>
</table>

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

\texttt{,\texttt{MF}=}

An optional input parameter that specifies the macro form.

\textbf{MF=S} Specifies 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.
**MF=L**  Specifies the list form of the macro. Use the list form together with the execute form of the macro for applications that require re-entrant code. The list form defines an area of storage that the execute form uses to store the parameters. Only the PLISTVER parameter can be coded with the list form of the macro.

**MF=E**  Specifies the execute form of the macro. Use the execute form together with the list form of the macro for applications that require re-entrant 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.

**MF=M**  Use 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.

,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 input string 1 - 60 characters in length 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.

**Guidelines:** 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.

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

   Valid values for MINFREE are in the range 0 - 9999. If a value outside of this range is specified, then CSM uses a default value. The default values for MINFREE are listed in Table 5.

   **Table 5. Default values for MINFREE**

<table>
<thead>
<tr>
<th>Pool size</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>4096</td>
<td>8</td>
</tr>
</tbody>
</table>

   z/OS V2R2.0 Communications Server: CSM Guide
Table 5. Default values for MINFREE (continued)

<table>
<thead>
<tr>
<th>Pool size</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>16384</td>
<td>4</td>
</tr>
<tr>
<td>32768</td>
<td>2</td>
</tr>
<tr>
<td>61440</td>
<td>2</td>
</tr>
<tr>
<td>184320</td>
<td>1</td>
</tr>
</tbody>
</table>

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

,PLISTVER=

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

**IMPLIED_VERSION**

The lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED_VERSION is the default.

**MAX**

Specify when you want the parameter list to be the largest size currently possible. This size might increase from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, you should 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  Specify when you use the currently available parameters.

To code, specify one of the following values:

- 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, with 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).

\texttt{\_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:

<table>
<thead>
<tr>
<th>Return code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Request completed successfully.</td>
</tr>
<tr>
<td>4</td>
<td>Request did not complete successfully. See the following reason codes to determine the type of error encountered.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Reason code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Specified buffer size is large than supported size.</td>
</tr>
<tr>
<td>4</td>
<td>Buffer pool cannot be expanded to satisfy request.</td>
</tr>
<tr>
<td>21</td>
<td>BUFSOURC value is not valid for this request.</td>
</tr>
<tr>
<td>23</td>
<td>Unable to create the specified pool. Creation of the pool would cause the ECSA maximum limit to be exceeded.</td>
</tr>
<tr>
<td>8</td>
<td>System error while processing the request. See the following reason codes to determine the type of error encountered.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Reason code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Unable to obtain storage for request.</td>
</tr>
<tr>
<td>2</td>
<td>Schedule SRB fail for PC routine.</td>
</tr>
<tr>
<td>3</td>
<td>Unable to create ALET for data space.</td>
</tr>
<tr>
<td>4</td>
<td>Error encountered while creating the data space.</td>
</tr>
<tr>
<td>5</td>
<td>Unable to create another data space. Number of data spaces exceeds the maximum.</td>
</tr>
<tr>
<td>6</td>
<td>An abend occurred while processing this request.</td>
</tr>
</tbody>
</table>

**IVTCSM REQUEST=DELETE_POOL**

**Purpose**

Use this macroinstruction to allow the user to indicate that it is no longer a registered user of the storage pool.

Because each pool might have multiple users, a storage pool is not 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

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=
An optional input parameter that specifies the macro form.

MF=S Specifies 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.

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

MF=M  Use 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.

\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 input string 1 - 60 characters in length 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.

Guidelines:

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.

\PLISTVER=
  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:

IMPLIES_VERSION
  The lowest version that allows all parameters specified on the request
to be processed. If you omit the PLISTVER parameter,
IMPLIES_VERSION is the default.

MAX
  Specify when you want the parameter list to be the largest size
  currently possible. This size might increase from release to release and
  affect the amount of storage that your program needs.
  If you can tolerate the size change, you should 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  Specify when you use the currently available parameters.

To code, specify one of the following values:
  • IMPLIED_VERSION
  • MAX
  • A decimal value of 0

,POOLTOKEN=pooltoken
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 code  Meaning
  6  An abend occurred while processing this request.
**IVTCSM REQUEST=DUMP_INFO**

**Purpose**

Use this macroinstruction to request 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 “CSM data space information (IVTDATSP)” on page 83.

**Syntax**

**main diagram**

```
IVTCSM REQUEST=DUMP_INFO, DS_INFO=ds_info

,RETCODE=retcode, RSCODE=rsncode

,PLISTVER=IMPLIED_VERSION

,PLISTVER=MAX, PLISTVER=0

,MF=S

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

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

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

**Parameters**

- **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=**
  
  An optional input parameter that specifies the macro form.
MF=S  Specifies 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.

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

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

\textit{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).

\textit{attr}

   An optional input string 1 - 60 characters in length 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 \textit{attr}, the system provides a value of 0D.

\textit{COMPLETE}

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

\textit{NOCHECK}

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

Guidelines:

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.

\textit{PLISTVER=}

   An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:
IMPLIED_VERSION
The lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED_VERSION is the default.

MAX Code this value if you want the parameter list to be the largest size currently possible. This size might increase from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, 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 Code this value if you use the currently available parameters.

To code, specify one of the following values:
• 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
Use this macroinstruction to allow 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 page freed, an application can use this macroinstruction to make the buffer guaranteed to be fixed.

See “Fixed buffers versus pageable buffers” on page 9 for more information.

Syntax

main diagram

```
name
IVTCSM REQUEST — FIX_BUFFER

parameters-1

, RETCODE — retcode
, RSNCODE — rsncode

, PLISTVER — IMPLIED_VERSION
 , PLISTVER — MAX
 , PLISTVER — 0

, MF = S

, MF = (L, list addr
, OD
, attr

, MF = (E, list addr
, COMPLETE

, MF = (M, list addr
, NOCHECK

, COMPLETE

, NOCHECK
```
Parameters

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. An 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=
An optional input parameter that specifies the macro form.
MF=S  Specifies 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.

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

MF=E  Specifies 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.

MF=M  Use 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.

_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 input string 1 - 60 characters in length 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.

Guidelines: 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.

 PLISTVER=
An optional input parameter that specifies the version of the macro. PLISTVER determines which parameter list the system generates. PLISTVER is an optional input parameter on all forms of the macro, including the list form. When using PLISTVER, specify it on all macro forms used for a request and with the same value on all of the macro forms. The values are:
**IMPLIED_VERSION**
The lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED_VERSION is the default.

**MAX** Specify when you want the parameter list to be the largest size currently possible. This size might increase from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, you should 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** Specify when you use the currently available parameters.

To code, specify one of the following values:
- 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=rsnrcode**
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 relevant only 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=**
An optional parameter, specifying whether or not the request should wait for fixed storage to become available. The default is WAIT=NO.
Specifies that this macroinstruction completes without waiting for fixed storage to become available.

Specifies that this macroinstruction is not complete until fixed storage becomes available. If fixed storage is not available, users are 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:

<table>
<thead>
<tr>
<th>Return code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Request completed successfully.</td>
</tr>
<tr>
<td>4</td>
<td>Request did not complete successfully. See the following reason codes to determine the type of error encountered:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Reason code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Requested function not supported at the present time, service has not been initialized.</td>
</tr>
<tr>
<td>7</td>
<td>Specified buffer token is not valid.</td>
</tr>
<tr>
<td>8</td>
<td>Instance ID in the input buffer token does not match that of the buffer, possible attempt to use a buffer that has been freed.</td>
</tr>
<tr>
<td>9</td>
<td>Real storage unavailable to provide a fixed buffer, wait not requested.</td>
</tr>
<tr>
<td>15</td>
<td>Fix buffer request failed because the state of the buffer is guaranteed to be pageable.</td>
</tr>
<tr>
<td>8</td>
<td>System error while processing the request.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Reason code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>An abend occurred while processing this request.</td>
</tr>
<tr>
<td>8</td>
<td>Page fix failed.</td>
</tr>
</tbody>
</table>

**IVTCSM REQUEST=FREE_BUFFER**

**Purpose**

Use this macroinstruction to allow 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 can 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 might 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 requestor of the buffer that specified a free routine on the GET_BUFFER request. If others use this option, the buffer is not returned to the original owner of the buffer.

The application can 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**

```
main diagram

IVTCSM REQUEST = FREE_BUFFER

parameters-1

,RETCODE = retcode
,RSNCODE = rsncode

,PLISTVER = IMPLIED_VERSION
,PLISTVER = MAX
,PLISTVER = 0

,MF = S

,MF = (L, list addr, attr, COMPLETE)
,MF = (E, list addr, NOCHECK, COMPLETE)
,MF = (M, list addr, NOCHECK)
```
Parameters

**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** (Required only when SKIPBUF=YES is specified.)
- **BUFL_TOKEN**

No fields in IVTBUFL 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 buffer entries in the list.

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

**CLEAR=**
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 is cleared when returned to the storage pool.

**CLEAR=YES**
Specifies that the buffer is to be cleared. Specifying CLEAR=YES does not cause a buffer to be cleared that is returned by 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=
An optional parameter, allowing the FREERTN parameter on the IVTC SM
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 be used only 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=
An optional input parameter that specifies the macro form.

MF=S  Specifies 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.

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

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

,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 input string 1 - 60 characters in length 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.

Guidelines: 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.

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

IMPLIED_VERSION
The lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED_VERSION is the default.

MAX
Code this value if you want the parameter list to be the largest size currently possible. This size might increase 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
Code this value if you use the currently available parameters

To code, specify one of the following values:
• 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).

,RSN_CODE=rsn_code
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=
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 are 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) are 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 relevant only 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:

<table>
<thead>
<tr>
<th>Return code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Request completed successfully.</td>
</tr>
<tr>
<td>4</td>
<td>Request did not complete successfully. See the following reason codes to determine the type of error encountered.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Reason code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Requested function not supported at the present time, service has not been initialized.</td>
</tr>
<tr>
<td>7</td>
<td>Pool token specified is not valid.</td>
</tr>
<tr>
<td>8</td>
<td>Instance ID in the input buffer token does not match that of the buffer, possible attempt to use a buffer that has been freed.</td>
</tr>
</tbody>
</table>

| 8           | System error while processing the request. |
Unable to obtain storage for request.

An abend occurred while processing this request.

**IVTCSM REQUEST=GET_BUFFER**

**Purpose**

Use 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" on page 40 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" on page 52 macroinstruction. For data space buffers, the pool token provided on the GET_BUFFER invocation is used to determine whether the returned buffer is backed by 31-bit or 64-bit real storage. If BACK=64 was specified on the CREATE_POOL invocation and the machine supports 64-bit backed storage, then a 64-bit backed buffer is returned. If BACK=31 was specified or taken as the default, or BACK=64 was specified but the machine is not executing in z/Architecture mode, then a 31-bit backed buffer is returned.

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 after processing is complete.

**Syntax**
main diagram

parameters-1

parameters-1

Parameters

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.
\texttt{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.

- \texttt{BUFL\_VERSION}

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

- \texttt{BUFL\_SOURCE}
- \texttt{BUFL\_TYPE}
- \texttt{BUFL\_TOKEN}
- \texttt{BUFL\_ALET} (Returned only if the buffer was allocated from a data space.)
- \texttt{BUFL\_ADDR}
- \texttt{BUFL\_SIZE}

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

\texttt{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.

\texttt{BUFTYPE=}

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

- \texttt{BUFTYPE=PAGEELIG}
  Indicates that the buffers are eligible to be made pageable.

- \texttt{BUFTYPE=PAGEABLE}
  Indicates that the buffers are to be guaranteed to be pageable.

- \texttt{BUFTYPE=FIXED}
  Indicates that buffers are to be guaranteed to be fixed.

\texttt{CLEAR=}

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

- \texttt{CLEAR=NO}
  Specifies that the buffer is not cleared when returned to the buffer pool

- \texttt{CLEAR=YES}
  Specifies that the buffer is cleared. Specifying CLEAR=YES does 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.

\texttt{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.

\texttt{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=
An optional input parameter that specifies the macro form.

MF=S  Specifies 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.

MF=L  Specifies 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 be coded with the list form of the macro.

MF=E  Specifies 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.

MF=M  Use 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.

,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 input string 1 - 60 characters in length 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.
Note: Use the modify and execute forms of IVTCSM in the following order:

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

\( \text{OWNERID} = \text{ownerid} \)

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.

\( \text{PLISTVER} = \)

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

**IMPLIED_VERSION**

The lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED_VERSION is the default.

**MAX**

Specify when you want the parameter list to be the largest size currently possible. This size might increase from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, you should 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

Specify when you use the currently available parameters.

To code, specify one of the following values:

- IMPLIED_VERSION
- MAX
- A decimal value of 0

\( \text{POOLTOKN} = \text{pooltokn} \)

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.

\( \text{RETCODE} = \text{retcode} \)

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).

\( \text{RSNCODE} = \text{rsncode} \)

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 relevant only 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=
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 does not complete until all buffers become available. If buffers are not available, users are suspended until enough buffers become available to satisfy the request.

,WAIT=EXPAND
Specifies that this macroinstruction waits for pool expansion to complete. If enough buffers are not available to satisfy the request, users are 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.

25 CSM is waiting for the buffers.

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**

Use this macroinstruction to allow 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 can 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 be used when a system service requires pageable storage on input. This macroinstruction can be issued for a buffer consisting of only 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**

```
+------------------------+-------+
| IVTCSM REQUEST = PAGE_BUFFER | name |
+------------------------+-------+
|                       |       |
|                    +-------+
|                    |   RETCODE = retcode |
|                    |   RSNCODE = rsncode |
|                       |       |
|                       |     PLISTVER = IMPLIED_VERSION |
|                       |     PLISTVER = MAX |
|                       |     PLISTVER = 0 |
|                       |       |
|                       | MF = S |
|                       |       |
|                       | MF = (L, list addr, attr) |
|                       |      | COMPLETE |
|                       |       |
|                       | MF = (E, list addr) |
|                       |      | NOCHECK |
|                       |       |
|                       | MF = (M, list addr) |
|                       |      | NOCHECK |
```

70 z/OS V2R2.0 Communications Server: CSM Guide
Parameters

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=
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=
An optional input parameter that specifies the macro form.

MF=S  Specifies 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

MF=L  Specifies 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 be coded with the
list form of the macro.

MF=E  Specifies 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.

MF=M  Use this value 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.

,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 input string 1 - 60 characters in length 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.

Note: 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.

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

**IMPLIED_VERSION**

The lowest version that allows all parameters specified on the request
to be processed. If you omit the PLISTVER parameter,
IMPLIED_VERSION is the default.

**MAX** Code this value if you want the parameter list to be the largest size
currently possible. This size might increase from release to release and
affect the amount of storage that your program needs.

If you can tolerate the size change, 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** Code this value if you use the currently available parameters.

To code, specify one of the following values:

- 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).

`RSCODE=rsocode`

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 relevant only 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**

<table>
<thead>
<tr>
<th>Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Request completed successfully.</td>
</tr>
<tr>
<td>4</td>
<td>Request did not complete successfully. See the following reason codes to determine the type of error encountered.</td>
</tr>
</tbody>
</table>

**Reason code**

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

**IVTCSM REQUEST=RESOURCE_STATS**

**Purpose**

Use this macroinstruction to request the address of the information that is required to monitor the usage of ECSA, data space, and fixed storage.

**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**
Parameters

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=
An optional input parameter that specifies the macro form.

MF=S  Specifies 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.

MF=L  Specifies 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 be coded with the list form of the macro.

MF=E  Specifies 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.

MF=M  Code this value 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.
**,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 input string 1 - 60 characters in length 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.

**Note:** 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.

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

**IMPLIED_VERSION**  
The lowest version that allows all parameters specified on the request to be processed. If you omit the PLISTVER parameter, IMPLIED_VERSION is the default.

**MAX**  
Code this value if you want the parameter list to be the largest size currently possible. This size might increase from release to release and affect the amount of storage that your program needs.

If you can tolerate the size change, you should 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  
Code this value if you use the currently available parameters.

To code, specify one of the following values:
- 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. IVTCSM macroinstruction return and reason codes

This appendix describes return and reason codes that can be returned to an application issuing the IVTCSM macroinstruction. These codes are listed in Table 6.

Guideline: For all macroinstructions that invoke CSM, the application can examine return codes in register 15 and reason codes in register 0.

Table 6. IVTCSM macroinstruction return and reason codes

<table>
<thead>
<tr>
<th>Return code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Request completed successfully.</td>
</tr>
<tr>
<td>4</td>
<td>Request did not complete successfully. See the following reason codes and meanings to identify the type of error encountered:</td>
</tr>
<tr>
<td>1</td>
<td>Requested function not supported.</td>
</tr>
<tr>
<td>2</td>
<td>Requested function not supported at the present time, service has not been initialized.</td>
</tr>
<tr>
<td>3</td>
<td>Specified buffer size is larger than supported size.</td>
</tr>
<tr>
<td>4</td>
<td>Buffer pool cannot be expanded to satisfy request.</td>
</tr>
<tr>
<td>5</td>
<td>No available buffers in pool, wait not requested.</td>
</tr>
<tr>
<td>6</td>
<td>Pool token specified is not valid.</td>
</tr>
<tr>
<td>7</td>
<td>Buffer token specified is not valid.</td>
</tr>
<tr>
<td>8</td>
<td>Instance ID in the input buffer token does not match that of the buffer, possible attempt to use a buffer that has been freed.</td>
</tr>
<tr>
<td>9</td>
<td>Real storage unavailable to provide a fixed buffer, wait not requested.</td>
</tr>
<tr>
<td>10</td>
<td>Request to make a buffer pageable denied, more than one image of the buffer exists.</td>
</tr>
<tr>
<td>11</td>
<td>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, issue a new create pool request to create a new pool.</td>
</tr>
<tr>
<td>12</td>
<td>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.</td>
</tr>
<tr>
<td>13</td>
<td>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.</td>
</tr>
<tr>
<td>14</td>
<td>Copy operation resulted in truncation of source data due to insufficient buffer space provided by the target buffer list.</td>
</tr>
<tr>
<td>15</td>
<td>Assign buffer request failed because the state of the buffer is guaranteed to be pageable.</td>
</tr>
<tr>
<td>16</td>
<td>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.</td>
</tr>
<tr>
<td>17</td>
<td>Extent has been overlaid. Reissue the request.</td>
</tr>
<tr>
<td>18</td>
<td>BUFL_SOURCE value is not valid for an entry in the Source buffer list (SRCLIST).</td>
</tr>
<tr>
<td>19</td>
<td>BUFL_SOURCE value is not valid for an entry in the Target buffer list (TRGLIST).</td>
</tr>
<tr>
<td>20</td>
<td>BUFTYPE value specified is not valid for this request</td>
</tr>
<tr>
<td>21</td>
<td>BUFSOURC value is not valid for this request.</td>
</tr>
<tr>
<td>22</td>
<td>Source and target buffers overlap, no data has been copied.</td>
</tr>
<tr>
<td>23</td>
<td>Unable to create the specified pool. Creation of the pool would cause the ECSA maximum limit to be exceeded.</td>
</tr>
<tr>
<td>24</td>
<td>ASID specified on the OWNERID parameter is not active.</td>
</tr>
<tr>
<td>25</td>
<td>CSM is waiting for the buffer.</td>
</tr>
<tr>
<td>26</td>
<td>Assign buffer request failed because CSM reached the maximum number of image buffers of the single CSM buffer.</td>
</tr>
<tr>
<td>Return code</td>
<td>Meaning</td>
</tr>
<tr>
<td>-------------</td>
<td>---------</td>
</tr>
<tr>
<td>8</td>
<td>System error while processing the request. See the following reason codes and meaning codes to identify the type of error encountered:</td>
</tr>
<tr>
<td>1</td>
<td>Unable to obtain storage for request.</td>
</tr>
<tr>
<td>2</td>
<td>Schedule SRB fail for PC routine.</td>
</tr>
<tr>
<td>3</td>
<td>Unable to create ALET for data space.</td>
</tr>
<tr>
<td>4</td>
<td>Error encountered while creating the data space.</td>
</tr>
<tr>
<td>5</td>
<td>Unable to create another data space. Number of data spaces exceeds the maximum.</td>
</tr>
<tr>
<td>6</td>
<td>An abend occurred while processing this request.</td>
</tr>
<tr>
<td>8</td>
<td>Page fix failed.</td>
</tr>
</tbody>
</table>
Appendix B. CSM DSECTs

This appendix contains the CSM DSECTs and contains the following sections:

- "CSM buffer list entry (IVTBUFL)"
- "CSM data space information (IVTDATSP)" on page 83
- "CSM resource status area (IVTSTATA)" on page 83

For general information on the use and purpose of DSECTs, see the z/OS Communications Server: SNA Programming.

Use the following information to interpret this appendix:

- 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. However, the object code, address columns, and statement number columns of the listing are not included.
- The source statements and comments are 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 that can be indicated by the BUFLIST, SRCLIST or TARGLIST parameters on the IVTCSM macroinstruction.

Requirements: The buffer list pointed to by the BUFLIST parameter is required on the following IVTCSM requests:

- "IVTCSM REQUEST=ASSIGN_BUFFER" on page 23
- "IVTCSM REQUEST=CHANGE_OWNER" on page 29
- "IVTCSM REQUEST=FIX_BUFFER" on page 52
- "IVTCSM REQUEST=FREE_BUFFER" on page 57
- "IVTCSM REQUEST=GET_BUFFER" on page 63
- "IVTCSM REQUEST=PAGE_BUFFER" on page 69

The buffer list pointed to by SRCLIST and TARGLIST is required on the "IVTCSM REQUEST=COPY_DATA" on page 34 macroinstruction.

Table 7. CSM buffer list format

<table>
<thead>
<tr>
<th>CSM buffer list format</th>
<th></th>
<th></th>
<th></th>
<th>BUFFER TOKEN</th>
<th>CSM DATA SPACE ALET</th>
<th>BUFFER ADDRESS</th>
<th>BUFFER LENGTH</th>
</tr>
</thead>
<tbody>
<tr>
<td>V E R S I O N</td>
<td>S O U R C E</td>
<td>T Y P E</td>
<td>0</td>
<td>BUFFER TOKEN</td>
<td>CSM DATA SPACE ALET</td>
<td>BUFFER ADDRESS</td>
<td>BUFFER LENGTH</td>
</tr>
</tbody>
</table>

Byte (hex) Contents

00 Version
<table>
<thead>
<tr>
<th>Buffer Source</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X'80'</td>
<td>ECSA</td>
</tr>
<tr>
<td>X'40'</td>
<td>Data space</td>
</tr>
<tr>
<td>X'20'</td>
<td>User data space</td>
</tr>
<tr>
<td>X'10'</td>
<td>User storage area other than a data space</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Buffer State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X'80'</td>
<td>Fixed</td>
</tr>
<tr>
<td>X'40'</td>
<td>Pageable</td>
</tr>
<tr>
<td>X'20'</td>
<td>Eligible to be page freed</td>
</tr>
</tbody>
</table>

04 - 0F
Buffer token

10 - 13
ALET for the data space

14 - 17
Address of buffer

18 - 1B
Length of buffer
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" on page 40 and "IVTCSM REQUEST=DUMP_INFO" on page 50 macroinstructions.

<table>
<thead>
<tr>
<th>Table 8. IVTDATSP DSECT layout</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATS</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Byte (hex)</th>
<th>Field name</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>00—03</td>
<td>DATSP_ACRO</td>
<td>DATS</td>
</tr>
<tr>
<td>04—07</td>
<td>DATSP_SNUM</td>
<td>Number of data spaces</td>
</tr>
<tr>
<td>08—0B</td>
<td>DATSP_SLEN</td>
<td>Length of a data space information entry</td>
</tr>
<tr>
<td>0C—</td>
<td>DATSP_SINF</td>
<td>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:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Byte (hex)</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>00—07</td>
<td>STOKEN for the data space</td>
</tr>
<tr>
<td>08—0F</td>
<td>Data space name</td>
</tr>
<tr>
<td>10</td>
<td>ALET for the data space</td>
</tr>
</tbody>
</table>

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" on page 40 and "IVTCSM REQUEST=RESOURCE_STATS" on page 74 macroinstructions.
For each resource type, the following bits are defined:

- One to indicate the usage of the resource is constrained
- One to indicate the usage is critical

If neither bit is set, the usage of the resource is considered normal. This allows CSM users 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 about other actions that an installation can consider to resolve resource shortages, see "Monitoring CSM storage" on page 2.

```
LOC SOURCE STATEMENT
000000 IVYSTATA 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 STATA_FLAG2 DS X Misc - Flags
  STATA_MONITOR EQU X'80' CSM Monitor active
```
Appendix C. Related protocol specifications

This appendix lists the related protocol specifications (RFCs) for TCP/IP. The Internet Protocol suite is still evolving through requests for comments (RFC). New protocols are being designed and implemented by researchers and are brought to the attention of the Internet community in the form of RFCs. Some of these protocols are so useful that they become recommended protocols. That is, all future implementations for TCP/IP are recommended to implement these particular functions or protocols. These become the de facto standards, on which the TCP/IP protocol suite is built.

You can request RFCs through electronic mail, from the automated Network Information Center (NIC) mail server, by sending a message to service@nic.ddn.mil with a subject line of RFC nnnn for text versions or a subject line of RFC nnnn.PS for PostScript versions. To request a copy of the RFC index, send a message with a subject line of RFC INDEX.

For more information, contact nic@nic.ddn.mil or at:

Government Systems, Inc.
Attn: Network Information Center
14200 Park Meadow Drive
Suite 200
Chantilly, VA 22021

Hard copies of all RFCs are available from the NIC, either individually or by subscription. Online copies are available at the following Web address:

http://www.rfc-editor.org/rfc.html

Draft RFCs that have been implemented in this and previous Communications Server releases are listed at the end of this topic.

Many features of TCP/IP Services are based on the following RFCs:

<table>
<thead>
<tr>
<th>RFC</th>
<th>Title and Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>RFC 652</td>
<td>Telnet output carriage-return disposition option D. Crocker</td>
</tr>
<tr>
<td>RFC 653</td>
<td>Telnet output horizontal tabstops option D. Crocker</td>
</tr>
<tr>
<td>RFC 654</td>
<td>Telnet output horizontal tab disposition option D. Crocker</td>
</tr>
<tr>
<td>RFC 655</td>
<td>Telnet output formfeed disposition option D. Crocker</td>
</tr>
<tr>
<td>RFC 657</td>
<td>Telnet output vertical tab disposition option D. Crocker</td>
</tr>
<tr>
<td>RFC 658</td>
<td>Telnet output linefeed disposition D. Crocker</td>
</tr>
<tr>
<td>RFC 698</td>
<td>Telnet extended ASCII option T. Mock</td>
</tr>
</tbody>
</table>
RFC 726
Remote Controlled Transmission and Echoing Telnet option J. Postel, D. Crocker

RFC 727
Telnet logout option M.R. Crispin

RFC 732
Telnet Data Entry Terminal option J.D. Day

RFC 733
Standard for the format of ARPA network text messages D. Crocker, J. Vittal, K.T. Pogran, D.A. Henderson

RFC 734
SUPDUP Protocol M.R. Crispin

RFC 735
Revised Telnet byte macro option D. Crocker, R.H. Gumpertz

RFC 736
Telnet SU P D U P option M.R. Crispin

RFC 749
Telnet SU P D U P—Output option B. Greenberg

RFC 765
File Transfer Protocol specification J. Postel

RFC 768
User Datagram Protocol J. Postel

RFC 779
Telnet send-location option E. Killian

RFC 783
TFTP Protocol (revision 2) K.R. Sollins

RFC 791
Internet Protocol J. Postel

RFC 792
Internet Control Message Protocol J. Postel

RFC 793
Transmission Control Protocol J. Postel

RFC 820
Assigned numbers J. Postel

RFC 821
Simple Mail Transfer Protocol J. Postel

RFC 822
Standard for the format of ARPA Internet text messages D. Crocker

RFC 823
DARPA Internet gateway R. Hinden, A. Sheltzer

RFC 826
Ethernet Address Resolution Protocol: Or converting network protocol addresses to 48-bit Ethernet address for transmission on Ethernet hardware D. Plummer

RFC 854
Telnet Protocol Specification J. Postel, J. Reynolds
Appendix C. Related protocol specifications
RFC 922
Broadcasting Internet datagrams in the presence of subnets J. Mogul

RFC 927
TACACS user identification Telnet option B.A. Anderson

RFC 933
Output marking Telnet option S. Silverman

RFC 946
Telnet terminal location number option R. Nedved

RFC 950
Internet Standard Subnetting Procedure J. Mogul, J. Postel

RFC 952
DoD Internet host table specification K. Harrenstien, M. Stahl, E. Feinler

RFC 959
File Transfer Protocol J. Postel, J.K. Reynolds

RFC 961
Official ARPA-Internet protocols J.K. Reynolds, J. Postel

RFC 974
Mail routing and the domain system C. Partridge

RFC 1001

RFC 1002

RFC 1006
ISO transport services on top of the TCP: Version 3 M.T. Rose, D.E. Cass

RFC 1009
Requirements for Internet gateways R. Braden, J. Postel

RFC 1011
Official Internet protocols J. Reynolds, J. Postel

RFC 1013
X Window System Protocol, version 11: Alpha update April 1987 R. Scheifler

RFC 1014
XDR: External Data Representation standard Sun Microsystems

RFC 1027
Using ARP to implement transparent subnet gateways S. Carl-Mitchell, J. Quarterman

RFC 1032
Domain administrators guide M. Stahl

RFC 1033
Domain administrators operations guide M. Lottor

RFC 1034
Domain names—concepts and facilities P.V. Mockapetris
RFC 1035
Domain names—implementation and specification P.V. Mockapetris

RFC 1038
Draft revised IP security option M. St. Johns

RFC 1041
Telnet 3270 regime option Y. Rekhter

RFC 1042
Standard for the transmission of IP datagrams over IEEE 802 networks J. Postel, J. Reynolds

RFC 1043
Telnet Data Entry Terminal option: DODIIS implementation A. Yasuda, T. Thompson

RFC 1044
Internet Protocol on Network System's HYPERchannel: Protocol specification K. Hardwick, J. Lekashman

RFC 1053
Telnet X.3 PAD option S. Levy, T. Jacobson

RFC 1055
Nonstandard for transmission of IP datagrams over serial lines: SLIP J. Romkey

RFC 1057

RFC 1058
Routing Information Protocol C. Hedrick

RFC 1060
Assigned numbers J. Reynolds, J. Postel

RFC 1067

RFC 1071
Computing the Internet checksum R.T. Braden, D.A. Borman, C. Partridge

RFC 1072
TCP extensions for long-delay paths V. Jacobson, R.T. Braden

RFC 1073
Telnet window size option D. Waitzman

RFC 1079
Telnet terminal speed option C. Hedrick

RFC 1085
ISO presentation services on top of TCP/IP based internets M.T. Rose

RFC 1091
Telnet terminal-type option J. VanBokkelen

RFC 1094
NFS: Network File System Protocol specification Sun Microsystems

RFC 1096
Telnet X display location option G. Marcy

RFC 1101
DNS encoding of network names and other types P. Mockapetris
RFC 1112
Host extensions for IP multicasting S.E. Deering

RFC 1113
Privacy enhancement for Internet electronic mail: Part I — message encipherment 
and authentication procedures J. Linn

RFC 1118
Hitchhiker's Guide to the Internet E. Krol

RFC 1122
Requirements for Internet Hosts—Communication Layers R. Braden, Ed.

RFC 1123
Requirements for Internet Hosts—Application and Support R. Braden, Ed.

RFC 1146
TCP alternate checksum options J. Zweig, C. Partridge

RFC 1155
Structure and identification of management information for TCP/IP-based 
internets M. Rose, K. McCloghrie

RFC 1156
Management Information Base for network management of TCP/IP-based internets 
K. McCloghrie, M. Rose

RFC 1157
Simple Network Management Protocol (SNMP) J. Case, M. Fedor, M. 
Schlofstall, J. Davin

RFC 1158
Management Information Base for network management of TCP/IP-based 
internets: MIB-II M. Rose

RFC 1166
Internet numbers S. Kirkpatrick, M.K. Stahl, M. Recker

RFC 1179
Line printer daemon protocol L. McLaughlin

RFC 1180
TCP/IP tutorial T. Socolofsky, C. Kale

RFC 1183
New DNS RR Definitions C.F. Everhart, L.A. Mamakos, R. Ullmann, P.V. 
Mockapetris

RFC 1184
Telnet Linemode Option D. Borman

RFC 1186
MD4 Message Digest Algorithm R.L. Rivest

RFC 1187
Bulk Table Retrieval with the SNMP M. Rose, K. McCloghrie, J. Davin

RFC 1188
Proposed Standard for the Transmission of IP Datagrams over FDDI Networks D. 
Katz

RFC 1190
Experimental Internet Stream Protocol: Version 2 (ST-II) C. Topolcic
RFC 1191
Path MTU discovery J. Mogul, S. Deering

RFC 1198
FYI on the X window system R. Scheifler

RFC 1207
FYI on Questions and Answers: Answers to commonly asked “experienced Internet user” questions G. Malkin, A. Marine, J. Reynolds

RFC 1208
Glossary of networking terms O. Jacobsen, D. Lynch

RFC 1213
Management Information Base for Network Management of TCP/IP-based internets: MIB-II K. McCloghrie, M.T. Rose

RFC 1215
Convention for defining traps for use with the SNMP M. Rose

RFC 1227
SNMP MUX protocol and MIB M.T. Rose

RFC 1228
SNMP-DPI: Simple Network Management Protocol Distributed Program Interface G. Carpenter, B. Wijnen

RFC 1229
Extensions to the generic-interface MIB K. McCloghrie

RFC 1230
IEEE 802.4 Token Bus MIB K. McCloghrie, R. Fox

RFC 1231
IEEE 802.5 Token Ring MIB K. McCloghrie, R. Fox, E. Decker

RFC 1236
IP to X.121 address mapping for DDN L. Morales, P. Hasse

RFC 1256
ICMP Router Discovery Messages S. Deering, Ed.

RFC 1267
Border Gateway Protocol 3 (BGP-3) K. Lougheed, Y. Rekhter

RFC 1268
Application of the Border Gateway Protocol in the Internet Y. Rekhter, P. Gross

RFC 1269
Definitions of Managed Objects for the Border Gateway Protocol: Version 3 S. Willis, J. Burruss

RFC 1270
SNMP Communications Services F. Kastenholz, ed.

RFC 1285
FDDI Management Information Base J. Case

RFC 1315
Management Information Base for Frame Relay DTEs C. Brown, F. Baker, C. Carvalho

RFC 1321
The MD5 Message-Digest Algorithm R. Rivest
RFC 1323
TCP Extensions for High Performance V. Jacobson, R. Braden, D. Borman

RFC 1325
FYI on Questions and Answers: Answers to Commonly Asked "New Internet User" Questions G. Malkin, A. Marine

RFC 1327
Mapping between X.400 (1988)/ISO 10021 and RFC 822 S. Hardcastle-Kille

RFC 1340
Assigned Numbers J. Reynolds, J. Postel

RFC 1344
Implications of MIME for Internet Mail Gateways N. Bornstein

RFC 1349
Type of Service in the Internet Protocol Suite P. Almquist

RFC 1350
The TFTP Protocol (Revision 2) K.R. Sollins

RFC 1351
SNMP Administrative Model J. Davin, J. Galvin, K. McCloghrie

RFC 1352
SNMP Security Protocols J. Galvin, K. McCloghrie, J. Davin

RFC 1353
Definitions of Managed Objects for Administration of SNMP Parties K. McCloghrie, J. Davin, J. Galvin

RFC 1354
IP Forwarding Table MIB F. Baker

RFC 1356
Multiprotocol Interconnect on X.25 and ISDN in the Packet Mode A. Malis, D. Robinson, R. Ullmann

RFC 1358
Charter of the Internet Architecture Board (IAB) L. Chapin

RFC 1363
A Proposed Flow Specification C. Partridge

RFC 1368
Definition of Managed Objects for IEEE 802.3 Repeater Devices D. McMaster, K. McCloghrie

RFC 1372
Telnet Remote Flow Control Option C. L. Hedrick, D. Borman

RFC 1374
IP and ARP on HIPPI J. Renwick, A. Nicholson

RFC 1381
SNMP MIB Extension for X.25 LAPB D. Throop, F. Baker

RFC 1382
SNMP MIB Extension for the X.25 Packet Layer D. Throop

RFC 1387
RIP Version 2 Protocol Analysis G. Malkin

RFC 1388
RIP Version 2 Carrying Additional Information G. Malkin
RFC 1389
RIP Version 2 MIB Extensions G. Malkin, F. Baker

RFC 1390
Transmission of IP and ARP over FDDI Networks D. Katz

RFC 1393
Traceroute Using an IP Option G. Malkin

RFC 1398
Definitions of Managed Objects for the Ethernet-Like Interface Types F.
  Kastenholz

RFC 1408
Telnet Environment Option D. Borman, Ed.

RFC 1413
Identification Protocol M. St. Johns

RFC 1416
Telnet Authentication Option D. Borman, ed.

RFC 1420
SNMP over IPX S. Bostock

RFC 1428
Transition of Internet Mail from Just-Send-8 to 8bit-SMTP/MIME G. Vaudreuil

RFC 1442
Structure of Management Information for version 2 of the Simple Network
  Management Protocol (SNMPv2) J. Case, K. McCloghrie, M. Rose, S.
  Waldbusser

RFC 1443
Textual Conventions for version 2 of the Simple Network Management Protocol
  (SNMPv2) J. Case, K. McCloghrie, M. Rose, S. Waldbusser

RFC 1445
Administrative Model for version 2 of the Simple Network Management Protocol
  (SNMPv2) J. Galvin, K. McCloghrie

RFC 1447
Party MIB for version 2 of the Simple Network Management Protocol
  (SNMPv2) K. McCloghrie, J. Galvin

RFC 1448
  (SNMPv2) J. Case, K. McCloghrie, M. Rose, S. Waldbusser

RFC 1464
Using the Domain Name System to Store Arbitrary String Attributes R.
  Rosenbaum

RFC 1469
IP Multicast over Token-Ring Local Area Networks T. Pusateri

RFC 1483
Multiprotocol Encapsulation over ATM Adaptation Layer 5 Juha Heinanen

RFC 1514
Host Resources MIB P. Grillo, S. Waldbusser

RFC 1516
Definitions of Managed Objects for IEEE 802.3 Repeater Devices D. McMaster,
  K. McCloghrie
RFC 1521
MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies N. Borenstein, N. Freed

RFC 1535
A Security Problem and Proposed Correction With Widely Deployed DNS Software E. Gavron

RFC 1536
Common DNS Implementation Errors and Suggested Fixes A. Kumar, J. Postel, C. Neuman, P. Danzig, S. Miller

RFC 1537
Common DNS Data File Configuration Errors P. Beertema

RFC 1540
Internet Official Protocol Standards J. Postel

RFC 1571
Telnet Environment Option Interoperability Issues D. Borman

RFC 1572
Telnet Environment Option S. Alexander

RFC 1573
Evolution of the Interfaces Group of MIB-II K. McCloghrie, F. Kastenholz

RFC 1577
Classical IP and ARP over ATM M. Laubach

RFC 1583
OSPF Version 2 J. Moy

RFC 1591
Domain Name System Structure and Delegation J. Postel

RFC 1592

RFC 1594
FYI on Questions and Answers—Answers to Commonly Asked “New Internet User” Questions A. Marine, J. Reynolds, G. Malkin

RFC 1644
T/TCP — TCP Extensions for Transactions Functional Specification R. Braden

RFC 1646
TN3270 Extensions for LUname and Printer Selection C. Graves, T. Butts, M. Angel

RFC 1647
TN3270 Enhancements B. Kelly

RFC 1652
SMTP Service Extension for 8bit-MIMEtransport J. Klensin, N. Freed, M. Rose, E. Stefferud, D. Crocker

RFC 1664
Using the Internet DNS to Distribute RFC1327 Mail Address Mapping Tables C. Allochio, A. Bonito, B. Cole, S. Giordano, R. Hagens

RFC 1693
An Extension to TCP: Partial Order Service T. Connolly, P. Amer, P. Conrad
RFC 1695
Definitions of Managed Objects for ATM Management Version 8.0 using SMIv2
M. Ahmed, K. Tesink

RFC 1701
Generic Routing Encapsulation (GRE) S. Hanks, T. Li, D. Farinacci, P. Traina

RFC 1702
Generic Routing Encapsulation over IPv4 networks S. Hanks, T. Li, D. Farinacci, P. Traina

RFC 1706
DNS NSAP Resource Records B. Manning, R. Colella

RFC 1712
DNS Encoding of Geographical Location C. Farrell, M. Schulze, S. Pleitner D. Baldoni

RFC 1713
Tools for DNS debugging A. Romao

RFC 1723
RIP Version 2—Carrying Additional Information G. Malkin

RFC 1752
The Recommendation for the IP Next Generation Protocol S. Bradner, A. Mankin

RFC 1766
Tags for the Identification of Languages H. Alvestrand

RFC 1771
A Border Gateway Protocol 4 (BGP-4) Y. Rekhter, T. Li

RFC 1794
DNS Support for Load Balancing T. Brisco

RFC 1819

RFC 1826
IP Authentication Header R. Atkinson

RFC 1828
IP Authentication using Keyed MD5 P. Metzger, W. Simpson

RFC 1829
The ESP DES-CBC Transform P. Karn, P. Metzger, W. Simpson

RFC 1830
SMTP Service Extensions for Transmission of Large and Binary MIME Messages G. Vaudreuil

RFC 1831

RFC 1832
XDR: External Data Representation Standard R. Srinivasan

RFC 1833
Binding Protocols for ONC RPC Version 2 R. Srinivasan

RFC 1850
OSPF Version 2 Management Information Base F. Baker, R. Coltun
RFC 1854
SMTP Service Extension for Command Pipelining N. Freed

RFC 1869
SMTP Service Extensions J. Klensin, N. Freed, M. Rose, E. Stefferud, D. Crocker

RFC 1870
SMTP Service Extension for Message Size Declaration J. Klensin, N. Freed, K. Moore

RFC 1876
A Means for Expressing Location Information in the Domain Name System C. Davis, P. Vixie, T. Goodwin, I. Dickinson

RFC 1883
Internet Protocol, Version 6 (IPv6) Specification S. Deering, R. Hinden

RFC 1884
IP Version 6 Addressing Architecture R. Hinden, S. Deering, Eds.

RFC 1886
DNS Extensions to support IP version 6 S. Thomson, C. Huitema

RFC 1888
OSI NSAPs and IPv6 J. Bound, B. Carpenter, D. Harrington, J. Houldsworth, A. Lloyd

RFC 1891
SMTP Service Extension for Delivery Status Notifications K. Moore

RFC 1892
The Multipart/Report Content Type for the Reporting of Mail System Administrative Messages G. Vaudreuil

RFC 1894
An Extensible Message Format for Delivery Status Notifications K. Moore, G. Vaudreuil

RFC 1901
Introduction to Community-based SNMPv2 J. Case, K. McCloghrie, M. Rose, S. Waldbusser

RFC 1902

RFC 1903

RFC 1904

RFC 1905

RFC 1906
RFC 1907

RFC 1908

RFC 1912
Common DNS Operational and Configuration Errors D. Barr

RFC 1918
Address Allocation for Private Internets Y. Rekhter, B. Moskowitz, D. Karrenberg, G.J. de Groot, E. Lear

RFC 1928

RFC 1930
Guidelines for creation, selection, and registration of an Autonomous System (AS) J. Hawkinson, T. Bates

RFC 1939
Post Office Protocol-Version 3 J. Myers, M. Rose

RFC 1981
Path MTU Discovery for IP version 6 J. McCann, S. Deering, J. Mogul

RFC 1982
Serial Number Arithmetic R. Elz, R. Bush

RFC 1985
SMTP Service Extension for Remote Message Queue Starting J. De Winter

RFC 1995
Incremental Zone Transfer in DNS M. Ohta

RFC 1996
A Mechanism for Prompt Notification of Zone Changes (DNS NOTIFY) P. Vixie

RFC 2010
Operational Criteria for Root Name Servers B. Manning, P. Vixie

RFC 2011
SNMPv2 Management Information Base for the Internet Protocol using SMIv2 K. McCloghrie, Ed.

RFC 2012

RFC 2013
SNMPv2 Management Information Base for the User Datagram Protocol using SMIv2 K. McCloghrie, Ed.

RFC 2018
TCP Selective Acknowledgement Options M. Mathis, J. Mahdavi, S. Floyd, A. Romanow

RFC 2026
The Internet Standards Process — Revision 3 S. Bradner
RFC 2030
Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OSI D. Mills

RFC 2033
Local Mail Transfer Protocol J. Myers

RFC 2034
SMTP Service Extension for Returning Enhanced Error Codes N. Freed

RFC 2040

RFC 2045
Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies N. Freed, N. Borenstein

RFC 2052
A DNS RR for specifying the location of services (DNS SRV) A. Gulbrandsen, P. Vixie

RFC 2065
Domain Name System Security Extensions D. Eastlake 3rd, C. Kaufman

RFC 2066
TELNET CHARSET Option R. Gellens

RFC 2080
RIPng for IPv6 G. Malkin, R. Minnear

RFC 2096
IP Forwarding Table MIB F. Baker

RFC 2104
HMAC: Keyed-Hashing for Message Authentication H. Krawczyk, M. Bellare, R. Canetti

RFC 2119
Keywords for use in RFCs to Indicate Requirement Levels S. Bradner

RFC 2133
Basic Socket Interface Extensions for IPv6 R. Gilligan, S. Thomson, J. Bound, W. Stevens

RFC 2136
Dynamic Updates in the Domain Name System (DNS UPDATE) P. Vixie, Ed., S. Thomson, Y. Rekhter, J. Bound

RFC 2137
Secure Domain Name System Dynamic Update D. Eastlake 3rd

RFC 2163
Using the Internet DNS to Distribute MIXER Conformant Global Address Mapping (MCGAM) C. Allocchio

RFC 2168
Resolution of Uniform Resource Identifiers using the Domain Name System R. Daniel, M. Mealling

RFC 2178
OSPF Version 2 J. Moy

RFC 2181
Clarifications to the DNS Specification R. Elz, R. Bush
RFC 2205

RFC 2210
The Use of RSVP with IETF Integrated Services J. Wroclawski

RFC 2211
Specification of the Controlled-Load Network Element Service J. Wroclawski

RFC 2212
Specification of Guaranteed Quality of Service S. Shenker, C. Partridge, R. Guerin

RFC 2215
General Characterization Parameters for Integrated Service Network Elements S. Shenker, J. Wroclawski

RFC 2217
Telnet Com Port Control Option G. Clarke

RFC 2219
Use of DNS Aliases for Network Services M. Hamilton, R. Wright

RFC 2228
FTP Security Extensions M. Horowitz, S. Lunt

RFC 2230
Key Exchange Delegation Record for the DNS R. Atkinson

RFC 2233
The Interfaces Group MIB using SMIv2 K. McCloghrie, F. Kastenholz

RFC 2240
A Legal Basis for Domain Name Allocation O. Vaughn

RFC 2246
The TLS Protocol Version 1.0 T. Dierks, C. Allen

RFC 2251

RFC 2253

RFC 2254
The String Representation of LDAP Search Filters T. Howes

RFC 2261
An Architecture for Describing SNMP Management Frameworks D. Harrington, R. Presuhn, B. Wijnen

RFC 2262

RFC 2271
An Architecture for Describing SNMP Management Frameworks D. Harrington, R. Presuhn, B. Wijnen

RFC 2273
SNMPv3 Applications D. Levi, P. Meyer, B. Stewartz
**RFC 2274**  
*User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)*  
U. Blumenthal, B. Wijnen

**RFC 2275**  
*View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP)*  
B. Wijnen, R. Presuhn, K. McCloghrie

**RFC 2279**  
*UTF-8, a transformation format of ISO 10646*  
F. Yergeau

**RFC 2292**  
*Advanced Sockets API for IPv6*  
W. Stevens, M. Thomas

**RFC 2308**  
*Negative Caching of DNS Queries (DNS NCache)*  
M. Andrews

**RFC 2317**  
*Classless IN-ADDR.ARPA delegation*  
H. Eidnes, G. de Groot, P. Vixie

**RFC 2320**  
*Definitions of Managed Objects for Classical IP and ARP Over ATM Using SMIv2 (IPOA-MIB)*  
M. Greene, J. Luciani, K. White, T. Kuo

**RFC 2328**  
*OSPF Version 2*  
J. Moy

**RFC 2345**  
*Domain Names and Company Name Retrieval*  
J. Klensin, T. Wolf, G. Oglesby

**RFC 2352**  
*A Convention for Using Legal Names as Domain Names*  
O. Vaughn

**RFC 2355**  
*TN3270 Enhancements*  
B. Kelly

**RFC 2358**  
*Definitions of Managed Objects for the Ethernet-like Interface Types*  
J. Flick, J. Johnson

**RFC 2373**  
*IP Version 6 Addressing Architecture*  
R. Hinden, S. Deering

**RFC 2374**  
*An IPv6 Aggregatable Global Unicast Address Format*  
R. Hinden, M. O’Dell, S. Deering

**RFC 2375**  
*IPv6 Multicast Address Assignments*  
R. Hinden, S. Deering

**RFC 2385**  
*Protection of BGP Sessions via the TCP MD5 Signature Option*  
A. Hefferman

**RFC 2389**  
*Feature negotiation mechanism for the File Transfer Protocol*  
P. Hethmon, R. Elz

**RFC 2401**  
*Security Architecture for Internet Protocol*  
S. Kent, R. Atkinson

**RFC 2402**  
*IP Authentication Header*  
S. Kent, R. Atkinson

**RFC 2403**  
*The Use of HMAC-MD5-96 within ESP and AH*  
C. Madson, R. Glenn
RFC 2523
Photuris: Extended Schemes and Attributes P. Karn, W. Simpson

RFC 2535
Domain Name System Security Extensions D. Eastlake 3rd

RFC 2538
Storing Certificates in the Domain Name System (DNS) D. Eastlake 3rd, O. Gudmundsson

RFC 2539
Storage of Diffie-Hellman Keys in the Domain Name System (DNS) D. Eastlake 3rd

RFC 2540
Detached Domain Name System (DNS) Information D. Eastlake 3rd

RFC 2554
SMTP Service Extension for Authentication J. Myers

RFC 2570
Introduction to Version 3 of the Internet-standard Network Management Framework J. Case, R. Mundy, D. Partain, B. Stewart

RFC 2571
An Architecture for Describing SNMP Management Frameworks B. Wijnen, D. Harrington, R. Presuhn

RFC 2572

RFC 2573
SNMP Applications D. Levi, P. Meyer, B. Stewart

RFC 2574

RFC 2575
View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP) B. Wijnen, R. Presuhn, K. McCloghrie

RFC 2576

RFC 2578

RFC 2579
Textual Conventions for SMIv2 K. McCloghrie, D. Perkins, J. Schoenwaelder

RFC 2580
Conformance Statements for SMIv2 K. McCloghrie, D. Perkins, J. Schoenwaelder

RFC 2581
TCP Congestion Control M. Allman, V. Paxson, W. Stevens

RFC 2583
Guidelines for Next Hop Client (NHC) Developers R. Carlson, L. Winkler
RFC 2591
Definitions of Managed Objects for Scheduling Management Operations D. Levi, J. Schoenwaelder

RFC 2625
IP and ARP over Fibre Channel M. Rajagopal, R. Bhagwat, W. Rickard

RFC 2635
Don't SPEW A Set of Guidelines for Mass Unsolicited Mailings and Postings (spam*) S. Hambridge, A. Lunde

RFC 2637

RFC 2640
Internationalization of the File Transfer Protocol B. Curtin

RFC 2665
Definitions of Managed Objects for the Ethernet-like Interface Types J. Flick, J. Johnson

RFC 2671
Extension Mechanisms for DNS (EDNS0) P. Vixie

RFC 2672
Non-Terminal DNS Name Redirection M. Crawford

RFC 2675
IPv6 Jumbograms D. Borman, S. Deering, R. Hinden

RFC 2710
Multicast Listener Discovery (MLD) for IPv6 S. Deering, W. Fenner, B. Haberman

RFC 2711
IPv6 Router Alert Option C. Partridge, A. Jackson

RFC 2740
OSPF for IPv6 R. Coltun, D. Ferguson, J. Moy

RFC 2753
A Framework for Policy-based Admission Control R. Yavatkar, D. Pendarakis, R. Guerin

RFC 2782
A DNS RR for specifying the location of services (DNS SRV) A. Gubrandsen, P. Vixix, L. Esibov

RFC 2821
Simple Mail Transfer Protocol J. Klensin, Ed.

RFC 2822
Internet Message Format P. Resnick, Ed.

RFC 2840
TELNET KERMIT OPTION J. Altman, F. da Cruz

RFC 2845
Secret Key Transaction Authentication for DNS (TSIG) P. Vixie, O. Gudmundsson, D. Eastlake 3rd, B. Wellington

RFC 2851
Textual Conventions for Internet Network Addresses M. Daniele, B. Haberman, S. Routhier, J. Schoenwaelder
RFC 2852
Deliver By SMTP Service Extension D. Newman

RFC 2874
DNS Extensions to Support IPv6 Address Aggregation and Renumbering M. Crawford, C. Huitema

RFC 2915
The Naming Authority Pointer (NAPTR) DNS Resource Record M. Mealling, R. Daniel

RFC 2920
SMTP Service Extension for Command Pipelining N. Freed

RFC 2930
Secret Key Establishment for DNS (TKEY RR) D. Eastlake, 3rd

RFC 2941
Telnet Authentication Option T. Ts’o, ed., J. Altman

RFC 2942
Telnet Authentication: Kerberos Version 5 T. Ts’o

RFC 2946
Telnet Data Encryption Option T. Ts’o

RFC 2952
Telnet Encryption: DES 64 bit Cipher Feedback T. Ts’o

RFC 2953
Telnet Encryption: DES 64 bit Output Feedback T. Ts’o

RFC 2992
Analysis of an Equal-Cost Multi-Path Algorithm C. Hopps

RFC 3019
IP Version 6 Management Information Base for The Multicast Listener Discovery Protocol B. Haberman, R. Wurzel

RFC 3060
Policy Core Information Model—Version 1 Specification B. Moore, E. Ellesson, J. Strassner, A. Westerinen

RFC 3152
Delegation of IP6.ARPA R. Bush

RFC 3164
The BSD Syslog Protocol C. Lonvick

RFC 3207
SMTP Service Extension for Secure SMTP over Transport Layer Security P. Hoffman

RFC 3226
DNSSEC and IPv6 A6 aware server/resolver message size requirements O. Gudmundsson

RFC 3291
Textual Conventions for Internet Network Addresses M. Daniele, B. Haberman, S. Routhier, J. Schoenwaelder

RFC 3363
RFC 3376
Internet Group Management Protocol, Version 3 B. Cain, S. Deering, I. Kouvelas, B. Fenner, A. Thyagarajan

RFC 3390
Increasing TCP’s Initial Window M. Allman, S. Floyd, C. Partridge

RFC 3410
Introduction and Applicability Statements for Internet-Standard Management Framework J. Case, R. Mundy, D. Partain, B. Stewart

RFC 3411

RFC 3412

RFC 3413

RFC 3414

RFC 3415
View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP) B. Wijnen, R. Presuhn, K. McCloghrie

RFC 3416

RFC 3417

RFC 3418

RFC 3419
Textual Conventions for Transport Addresses M. Daniele, J. Schoenwaelder

RFC 3484
Default Address Selection for Internet Protocol version 6 (IPv6) R. Draves

RFC 3493

RFC 3513
Internet Protocol Version 6 (IPv6) Addressing Architecture R. Hinden, S. Deering

RFC 3526
More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE) T. Kivinen, M. Kojo
RFC 3542
Advanced Sockets Application Programming Interface (API) for IPv6 W. Richard Stevens, M. Thomas, E. Nordmark, T. Jinmei

RFC 3566
The AES-XCBC-MAC-96 Algorithm and Its Use With IPsec S. Frankel, H. Herbert

RFC 3569
An Overview of Source-Specific Multicast (SSM) S. Bhattacharyya, Ed.

RFC 3584

RFC 3602
The AES-CBC Cipher Algorithm and Its Use with IPsec S. Frankel, R. Glenn, S. Kelly

RFC 3629
UTF-8, a transformation format of ISO 10646 R. Kermode, C. Vicisano

RFC 3658
Delegation Signer (DS) Resource Record (RR) O. Gudmundsson

RFC 3678
Socket Interface Extensions for Multicast Source Filters D. Thaler, B. Fenner, B. Quinn

RFC 3715
IPsec-Network Address Translation (NAT) Compatibility Requirements B. Aboba, W. Dixon

RFC 3810

RFC 3826

RFC 3947
Negotiation of NAT-Traversal in the IKE T. Kivinen, B. Swander, A. Huttunen, V. Volpe

RFC 3948
UDP Encapsulation of IPsec ESP Packets A. Huttunen, B. Swander, V. Volpe, L. DiBurro, M. Stenberg

RFC 4001
Textual Conventions for Internet Network Addresses M. Daniele, B. Haberman, S. Routhier, J. Schoenwaelder

RFC 4007
IPv6 Scoped Address Architecture S. Deering, B. Haberman, T. Jinmei, E. Nordmark, B. Zill

RFC 4022
Management Information Base for the Transmission Control Protocol (TCP) R. Raghunarayan

RFC 4106
The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP) J. Viega, D. McGrew
RFC 4835
Cryptographic Algorithm Implementation Requirements for Encapsulating
Security Payload (ESP) and Authentication Header (AH) V. Manral

RFC 4862
IPv6 Stateless Address Autoconfiguration S. Thomson, T. Narten, T. Jinmei

RFC 4868
Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 with IPsec S.
Kelly, S. Frankel

RFC 4869
Suite B Cryptographic Suites for IPsec L. Law, J. Solinas

RFC 4941
Privacy Extensions for Stateless Address Autoconfiguration in IPv6 T. Narten, R.
Draves, S. Krishnan

RFC 4945
The Internet IP Security PKI Profile of IKEv1/ISAKMP, IKEv2, and PKIX B.
Korver

RFC 5014
IPv6 Socket API for Source Address Selection E. Nordmark, S. Chakrabarti, J.
Laganier

RFC 5095
Deprecation of Type 0 Routing Headers in IPv6 J. Abery, P. Savola, G.
Neville-Neil

RFC 5175
IPv6 Router Advertisement Flags Option B. Haberman, Ed., R. Hinden

RFC 5282
Using Authenticated Encryption Algorithms with the Encrypted Payload of the
Internet Key Exchange version 2 (IKEv2) Protocol D. Black, D. McGrew

RFC 5996
Internet Key Exchange Protocol Version 2 (IKEv2) C. Kaufman, P. Hoffman, Y.
Nir, P. Eronen

Internet drafts

Internet drafts are working documents of the Internet Engineering Task Force
(IETF), its areas, and its working groups. Other groups can also distribute working
documents as Internet drafts. You can see Internet drafts at http://www.ietf.org/
ID.html
Appendix D. Architectural specifications

This appendix lists documents that provide architectural specifications for the SNA Protocol.

The APPN Implementers' Workshop (AIW) architecture documentation includes the following architectural specifications for SNA APPN and HPR:
- APPN Architecture Reference (SG30-3422-04)
- APPN Branch Extender Architecture Reference Version 1.1
- APPN Dependent LU Requester Architecture Reference Version 1.5
- APPN Extended Border Node Architecture Reference Version 1.0
- APPN High Performance Routing Architecture Reference Version 4.0
- SNA Formats (GA27-3136-20)
- SNA Technical Overview (GC30-3073-04)

For more information, see the AIW documentation page at http://www.ibm.com/support/docview.wss?rs=852&uid=swg27017843.

The following RFC also contains SNA architectural specifications:
- RFC 2353 APPN/HPR in IP Networks APPN Implementers' Workshop Closed Pages Document

RFCs can be obtained from:

Government Systems, Inc.
Attn: Network Information Center
14200 Park Meadow Drive
Suite 200
Chantilly, VA 22021

Many RFCs are available online. Hardcopies of all RFCs are available from the NIC, either individually or by subscription. Online copies are available using FTP from the NIC at http://www.rfc-editor.org/rfc.html.

Use FTP to download the files, using the following format:
RFC:RFC-INDEX.TXT
RFC:RFCnnnn.TXT
RFC:RFCnnnn.PS

where:
- nnnn is the RFC number.
- TXT is the text format.
- PS is the postscript format.

You can also request RFCs through electronic mail, from the automated NIC mail server, by sending a message to service@nic.ddn.mil with a subject line of RFC nnnn for text versions or a subject line of RFC nnnn.PS for PostScript versions. To request a copy of the RFC index, send a message with a subject line of RFC INDEX.
For more information, contact nic@nic.ddn.mil.
Appendix E. Accessibility

Publications for this product are offered in Adobe Portable Document Format (PDF) and should be compliant with accessibility standards. If you experience difficulties when using PDF files, you can view the information through the z/OS Internet Library website or IBM Knowledge Center. If you continue to experience problems, send an email to mhvrcfs@us.ibm.com or write to:

IBM Corporation
Attention: MHVRCFS Reader Comments
Department H6MA, Building 707
2455 South Road
Poughkeepsie, NY 12601-5400
USA

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

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

Using assistive technologies

Assistive technology products, such as screen readers, function with the user interfaces found in z/OS. Consult the assistive technology documentation for specific information when using such products to access z/OS interfaces.

Keyboard navigation of the user interface

Users can access z/OS user interfaces using TSO/E or ISPF. See z/OS TSO/E Primer, z/OS TSO/E User’s Guide and z/OS ISPF User’s Guide Vol I for information about accessing TSO/E and ISPF interfaces. These guides describe how to use TSO/E and ISPF, including the use of keyboard shortcuts or function keys (PF keys). Each guide includes the default settings for the PF keys and explains how to modify their functions.

z/OS information

z/OS information is accessible using screen readers with the BookServer or Library Server versions of z/OS books in the Internet library at www.ibm.com/systems/z/os/zos/bkserv/

One exception is command syntax that is published in railroad track format, which is accessible using screen readers with IBM Knowledge Center, as described in “Dotted decimal syntax diagrams.”

Dotted decimal syntax diagrams

Syntax diagrams are provided in dotted decimal format for users accessing IBM Knowledge Center using a screen reader. In dotted decimal format, each syntax element is written on a separate line. If two or more syntax elements are always
present together (or always absent together), they can appear on the same line, because they can be considered as a single compound syntax element.

Each line starts with a dotted decimal number; for example, 3 or 3.1 or 3.1.1. To hear these numbers correctly, make sure that your screen reader is set to read out punctuation. All the syntax elements that have the same dotted decimal number (for example, all the syntax elements that have the number 3.1) are mutually exclusive alternatives. If you hear the lines 3.1 USERID and 3.1 SYSTEMID, you know that your syntax can include either USERID or SYSTEMID, but not both.

The dotted decimal numbering level denotes the level of nesting. For example, if a syntax element with dotted decimal number 3 is followed by a series of syntax elements with dotted decimal number 3.1, all the syntax elements numbered 3.1 are subordinate to the syntax element numbered 3.

Certain words and symbols are used next to the dotted decimal numbers to add information about the syntax elements. Occasionally, these words and symbols might occur at the beginning of the element itself. For ease of identification, if the word or symbol is a part of the syntax element, it is preceded by the backslash (\) character. The * symbol can be used next to a dotted decimal number to indicate that the syntax element repeats. For example, syntax element *FILE with dotted decimal number 3 is given the format 3 \* FILE. Format 3* FILE indicates that syntax element FILE repeats. Format 3* \* FILE indicates that syntax element * FILE repeats.

Characters such as commas, which are used to separate a string of syntax elements, are shown in the syntax just before the items they separate. These characters can appear on the same line as each item, or on a separate line with the same dotted decimal number as the relevant items. The line can also show another symbol giving information about the syntax elements. For example, the lines 5.1*, 5.1 LASTRUN, and 5.1 DELETE mean that if you use more than one of the LASTRUN and DELETE syntax elements, the elements must be separated by a comma. If no separator is given, assume that you use a blank to separate each syntax element.

If a syntax element is preceded by the % symbol, this indicates a reference that is defined elsewhere. The string following the % symbol is the name of a syntax fragment rather than a literal. For example, the line 2.1 %OP1 means that you should see separate syntax fragment OP1.

The following words and symbols are used next to the dotted decimal numbers:

- A question mark (?) means an optional syntax element. A dotted decimal number followed by the ? symbol indicates that all the syntax elements with a corresponding dotted decimal number, and any subordinate syntax elements, are optional. If there is only one syntax element with a dotted decimal number, the ? symbol is displayed on the same line as the syntax element, (for example 5? NOTIFY). If there is more than one syntax element with a dotted decimal number, the ? symbol is displayed on a line by itself, followed by the syntax elements that are optional. For example, if you hear the lines 5 ?, 5 NOTIFY, and 5 UPDATE, you know that syntax elements NOTIFY and UPDATE are optional; that is, you can choose one or none of them. The ? symbol is equivalent to a bypass line in a railroad diagram.

- An exclamation mark (!) means a default syntax element. A dotted decimal number followed by the ! symbol and a syntax element indicate that the syntax element is the default option for all syntax elements that share the same dotted
decimal number. Only one of the syntax elements that share the same dotted
decimal number can specify a ! symbol. For example, if you hear the lines 2?
FILE, 2.1! (KEEP), and 2.1 (DELETE), you know that (KEEP) is the default option
for the FILE keyword. In this example, if you include the FILE keyword but do
not specify an option, default option KEEP will be applied. A default option also
applies to the next higher dotted decimal number. In this example, if the FILE
keyword is omitted, default FILE(KEEP) is used. However, if you hear the lines
2? FILE, 2.1, 2.1.1! (KEEP), and 2.1.1 (DELETE), the default option KEEP applies
only to the next higher dotted decimal number, 2.1 (which does not have an
associated keyword), and does not apply to 2? FILE. Nothing is used if the
keyword FILE is omitted.

• An asterisk (*) means a syntax element that can be repeated 0 or more times. A
dotted decimal number followed by the * symbol indicates that this syntax
element can be used zero or more times; that is, it is optional and can be
repeated. For example, if you hear the line 5.1* data area, you know that you
can include one data area, more than one data area, or no data area. If you hear
the lines 3*, 3 HOST, and 3 STATE, you know that you can include HOST,
STATE, both together, or nothing.

Notes:
1. If a dotted decimal number has an asterisk (*) next to it and there is only one
   item with that dotted decimal number, you can repeat that same item more
   than once.
2. If a dotted decimal number has an asterisk next to it and several items have
   that dotted decimal number, you can use more than one item from the list,
   but you cannot use the items more than once each. In the previous example,
   you could write HOST STATE, but you could not write HOST HOST.
3. The * symbol is equivalent to a loop-back line in a railroad syntax diagram.

• + means a syntax element that must be included one or more times. A dotted
decimal number followed by the + symbol indicates that this syntax element
must be included one or more times; that is, it must be included at least once
and can be repeated. For example, if you hear the line 6.1+ data area, you must
include at least one data area. If you hear the lines 2+, 2 HOST, and 2 STATE,
you know that you must include HOST, STATE, or both. Similar to the * symbol,
the + symbol can only repeat a particular item if it is the only item with that
dotted decimal number. The + symbol, like the * symbol, is equivalent to a
loop-back line in a railroad syntax diagram.
Notices

This information was developed for products and services offered in the USA.

IBM may not offer all of the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.
IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who want to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

Site Counsel
IBM Corporation
P.O. Box 12195
3039 Cornwallis Road
Research Triangle Park, North Carolina 27709-2195
U.S.A

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

If you are viewing this information softcopy, the photographs and color illustrations might not appear.

COPYRIGHT LICENSE:

This information might contain sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing
application programs conforming to the application programming interface for the
operating platform for which the sample programs are written. These examples
have not been thoroughly tested under all conditions. IBM, therefore, cannot
guarantee or imply reliability, serviceability, or function of these programs. The
sample programs are provided "AS IS", without warranty of any kind. IBM shall
not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work must
include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp.
Sample Programs. © Copyright IBM Corp. _enter the year or years_.

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,
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,

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.

Some portions of this publication relating to X Window System are Copyright ©

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.


Copyright © 1988, 1993 The Regents of the University of California.

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:
1. Redistributions of source code must retain the above copyright notice, this list
   of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this
   list of conditions and the following disclaimer in the documentation and/or
   other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software must
   display the following acknowledgment:
This product includes software developed by the University of California, Berkeley and its contributors.

4. Neither the name of the University nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This software program contains code, and/or derivatives or modifications of code originating from the software program "Popper." Popper is Copyright ©1989-1991 The Regents of the University of California. Popper was created by Austin Shelton, Information Systems and Technology, University of California, Berkeley.

Permission from the Regents of the University of California to use, copy, modify, and distribute the "Popper" software contained herein for any purpose, without fee, and without a written agreement is hereby granted, provided that the above copyright notice and this paragraph and the following two paragraphs appear in all copies. HOWEVER, ADDITIONAL PERMISSIONS MAY BE NECESSARY FROM OTHER PERSONS OR ENTITIES, TO USE DERIVATIVES OR MODIFICATIONS OF POPPER.

IN NO EVENT SHALL THE UNIVERSITY OF CALIFORNIA BE LIABLE TO ANY PARTY FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, INCLUDING LOST PROFITS, ARISING OUT OF THE USE OF THE POPPER SOFTWARE, OR ITS DERIVATIVES OR MODIFICATIONS, AND ITS DOCUMENTATION, EVEN IF THE UNIVERSITY OF CALIFORNIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

THE UNIVERSITY OF CALIFORNIA SPECIFICALLY DISCLAIMS ANY WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE POPPER SOFTWARE PROVIDED HEREUNDER IS ON AN "AS IS" BASIS, AND THE UNIVERSITY OF CALIFORNIA HAS NO OBLIGATIONS TO PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.

Copyright © 1983 The Regents of the University of California.

Redistribution and use in source and binary forms are permitted provided that the above copyright notice and this paragraph are duplicated in all such forms and that any documentation, advertising materials, and other materials related to such distribution and use acknowledge that the software was developed by the University of California, Berkeley. The name of the University may not be used to endorse or promote products derived from this software without specific prior
written permission. THIS SOFTWARE IS PROVIDED "AS IS" AND WITHOUT 
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, WITHOUT 
LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND 
FITNESS FOR A PARTICULAR PURPOSE.

Copyright © 1991, 1993 The Regents of the University of California.

Redistribution and use in source and binary forms, with or without modification, 
are permitted provided that the following conditions are met:
1. Redistributions of source code must retain the above copyright notice, this list 
of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this 
list of conditions and the following disclaimer in the documentation and/or 
other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software must 
display the following acknowledgment:
   This product includes software developed by the University of 
   California, Berkeley and its contributors.
4. Neither the name of the University nor the names of its contributors may be 
used to endorse or promote products derived from this software without 
specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS "AS 
IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT 
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND 
FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT 
SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, 
INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL 
 DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF 
SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR 
BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF 
LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT 
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF 
The use of this software, even if advised of the possibility of 
such damage.

Copyright © 1990 by the Massachusetts Institute of Technology

Export of this software from the United States of America may require a specific 
license from the United States Government. It is the responsibility of any person or 
organization contemplating export to obtain such a license before exporting.

WITHIN THAT CONSTRAINT, permission to use, copy, modify, and distribute this 
software and its documentation for any purpose and without fee is hereby granted, 
provided that the above copyright notice appear in all copies and that both that 
copyright notice and this permission notice appear in supporting documentation, 
and that the name of M.I.T. not be used in advertising or publicity pertaining to 
distribution of the software without specific, written prior permission. Furthermore 
if you modify this software you must label your software as modified software and 
not distribute it in such a fashion that it might be confused with the original M.I.T. 
software. M.I.T. makes no representations about the suitability of this software for 
any purpose. It is provided "as is" without express or implied warranty.

Copyright © 1998 by the FundsXpress, INC.
Export of this software from the United States of America may require a specific license from the United States Government. It is the responsibility of any person or organization contemplating export to obtain such a license before exporting.

WITHIN THAT CONSTRAINT, permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of FundsXpress not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission. FundsXpress makes no representations about the suitability of this software for any purpose. It is provided "as is" without express or implied warranty.

THIS SOFTWARE IS PROVIDED "AS IS" AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.


Permission to use, copy, modify, and distribute this software for any purpose with or without fee is hereby granted, provided that the above copyright notice and this permission notice appear in all copies.

THE SOFTWARE IS PROVIDED "AS IS" AND INTERNET SOFTWARE CONSORTIUM DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL INTERNET SOFTWARE CONSORTIUM BE LIABLE FOR ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Copyright © 1995-1998 Eric Young (eay@cryptsoft.com)

This package is an SSL implementation written by Eric Young (eay@cryptsoft.com). The implementation was written so as to conform with Netscape’s SSL.

This library is free for commercial and non-commercial use as long as the following conditions are adhered to. The following conditions apply to all code found in this distribution, be it the RC4, RSA, lhash, DES, etc., code; not just the SSL code. The SSL documentation included with this distribution is covered by the same copyright terms except that the holder is Tim Hudson (tjh@cryptsoft.com).

Copyright remains Eric Young’s, and as such any Copyright notices in the code are not to be removed. If this package is used in a product, Eric Young should be given attribution as the author of the parts of the library used. This can be in the form of a textual message at program startup or in documentation (online or textual) provided with the package.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this
list of conditions and the following disclaimer in the documentation and/or
other materials provided with the distribution.

3. All advertising materials mentioning features or use of this software must
display the following acknowledgment: "This product includes cryptographic
software written by Eric Young (eay@cryptsoft.com)". The word 'cryptographic'
 can be left out if the routines from the library being used are not cryptographic
related.

4. If you include any Windows specific code (or a derivative thereof) from the
apps directory (application code) you must include acknowledgment:
"This product includes software written by Tim Hudson (tjh@cryptsoft.com)"

This product includes cryptographic software written by Eric Young.


Permission to use, copy, modify, and distribute this software for any purpose with
or without fee is hereby granted, provided that the above copyright notice and this
permission notice appear in all copies.

THE SOFTWARE IS PROVIDED "AS IS" AND INTERNET SOFTWARE
CONSORTIUM DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS
SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY
AND FITNESS. IN NO EVENT SHALL INTERNET SOFTWARE CONSORTIUM
BE LIABLE FOR ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL
DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS
OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT,
NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN
CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Copyright © 2004 IBM Corporation and its licensors, including Sendmail, Inc., and
the Regents of the University of California.

Copyright © 1999,2000,2001 Compaq Computer Corporation

Copyright © 1999,2000,2001 Hewlett-Packard Company
Copyright © 1999,2000,2001 IBM Corporation
Copyright © 1999,2000,2001 Hummingbird Communications Ltd.
Copyright © 1999,2000,2001 Sun Microsystems, Inc.
Copyright © 1999,2000,2001 The Open Group

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, provided that the above copyright notice(s) and this permission notice appear in all copies of the Software and that both the above copyright notice(s) and this permission notice appear in supporting documentation.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE BE LIABLE FOR ANY CLAIM, OR ANY SPECIAL INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Except as contained in this notice, the name of a copyright holder shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization of the copyright holder.

X Window System is a trademark of The Open Group.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

You can obtain softcopy from the z/OS Collection (SK3T-4269), which contains BookManager and PDF formats.

Minimum supported hardware

The minimum supported hardware for z/OS releases identified in z/OS announcements can subsequently change when service for particular servers or devices is withdrawn. Likewise, the levels of other software products supported on a particular release of z/OS are subject to the service support lifecycle of those products. Therefore, z/OS and its product publications (for example, panels, samples, messages, and product documentation) can include references to hardware and software that is no longer supported.

- For information about software support lifecycle, see [http://www-01.ibm.com/software/support/systemsz/lifecycle/](http://www-01.ibm.com/software/support/systemsz/lifecycle/)
- For information about currently-supported IBM hardware, contact your IBM representative.
Programming interface information

This publication documents intended Programming Interfaces that allow the customer to write programs to obtain the services of z/OS Communications Server.

Policy for unsupported hardware

Various z/OS elements, such as DFSMS, HCD, JES2, JES3, and MVS, contain code that supports specific hardware servers or devices. In some cases, this device-related element support remains in the product even after the hardware devices pass their announced End of Service date. z/OS may continue to service element code; however, it will not provide service related to unsupported hardware devices. Software problems related to these devices will not be accepted for service, and current service activity will cease if a problem is determined to be associated with out-of-support devices. In such cases, fixes will not be issued.

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at Copyright and trademark information at www.ibm.com/legal/copytrade.shtml.

Intel is a registered trademark of Intel Corporation or its subsidiaries in the United States and other countries.

Java™ and all Java-based trademarks are trademarks or registered trademarks of Oracle and/or its affiliates.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, and Windows NT are trademarks of Microsoft Corporation in the United States, other countries, or both.

Adobe and PostScript are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other product and service names might be trademarks of IBM or other companies.
Bibliography

This bibliography contains descriptions of the documents in the z/OS Communications Server library.

z/OS Communications Server documentation is available in the following forms:
- Online at the z/OS Internet Library web page at www.ibm.com/systems/z/os/zos/bkserv/
- In softcopy on CD-ROM collections. See “Softcopy information” on page xv.

z/OS Communications Server library updates

An index to z/OS Communications Server book updates is at http://www.ibm.com/support/docview.wss?uid=swg21178966. Updates to documents are also available on RETAIN and in information APARs. Go to http://www.ibm.com/software/network/commserver/zos/support to view information APARs.

z/OS Communications Server information

z/OS Communications Server product information is grouped by task in the following tables.

Planning

<table>
<thead>
<tr>
<th>Title</th>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>z/OS Communications Server: New Function Summary</td>
<td>GC27-3664</td>
<td>This document is intended to help you plan for new IP or SNA function, whether you are migrating from a previous version or installing z/OS for the first time. It summarizes what is new in the release and identifies the suggested and required modifications needed to use the enhanced functions.</td>
</tr>
<tr>
<td>z/OS Communications Server: IPv6 Network and Application Design Guide</td>
<td>SC27-3663</td>
<td>This document is a high-level introduction to IPv6. It describes concepts of z/OS Communications Server's support of IPv6, coexistence with IPv4, and migration issues.</td>
</tr>
</tbody>
</table>

Resource definition, configuration, and tuning

<table>
<thead>
<tr>
<th>Title</th>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>z/OS Communications Server: IP Configuration Guide</td>
<td>SC27-3650</td>
<td>This document describes the major concepts involved in understanding and configuring an IP network. Familiarity with the z/OS operating system, IP protocols, z/OS UNIX System Services, and IBM Time Sharing Option (TSO) is recommended. Use this document with the z/OS Communications Server: IP Configuration Reference</td>
</tr>
<tr>
<td>Title</td>
<td>Number</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| z/OS Communications Server: IP Configuration Reference | SC27-3651 | This document presents information for people who want to administer and maintain IP. Use this document with the z/OS Communications Server: IP Configuration Guide. The information in this document includes:  
  - TCP/IP configuration data sets  
  - Configuration statements  
  - Translation tables  
  - Protocol number and port assignments |
| z/OS Communications Server: SNA Network Implementation Guide | SC27-3672 | This document presents the major concepts involved in implementing an SNA network. Use this document with the z/OS Communications Server: SNA Resource Definition Reference. |
| z/OS Communications Server: SNA Resource Definition Reference | SC27-3675 | This document describes each SNA definition statement, start option, and macroinstruction for user tables. It also describes NCP definition statements that affect SNA. Use this document with the z/OS Communications Server: SNA Network Implementation Guide. |
| z/OS Communications Server: SNA Resource Definition Samples | SC27-3676 | This document contains sample definitions to help you implement SNA functions in your networks, and includes sample major node definitions. |
| z/OS Communications Server: IP Network Print Facility | SC27-3658 | This document is for systems programmers and network administrators who need to prepare their network to route SNA, JES2, or JES3 printer output to remote printers using TCP/IP Services. |

### Operation

<table>
<thead>
<tr>
<th>Title</th>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>z/OS Communications Server: IP User’s Guide and Commands</td>
<td>SC27-3662</td>
<td>This document describes how to use TCP/IP applications. It contains requests with which a user can log on to a remote host using Telnet, transfer data sets using FTP, send and receive electronic mail, print on remote printers, and authenticate network users.</td>
</tr>
<tr>
<td>z/OS Communications Server: IP System Administrator’s Commands</td>
<td>SC27-3661</td>
<td>This document describes the functions and commands helpful in configuring or monitoring your system. It contains system administrator’s commands, such as TSO NETSTAT, PING, TRACERTE and their UNIX counterparts. It also includes TSO and MVS commands commonly used during the IP configuration process.</td>
</tr>
<tr>
<td>z/OS Communications Server: SNA Operation</td>
<td>SC27-3673</td>
<td>This document serves as a reference for programmers and operators requiring detailed information about specific operator commands.</td>
</tr>
<tr>
<td>z/OS Communications Server: Quick Reference</td>
<td>SC27-3665</td>
<td>This document contains essential information about SNA and IP commands.</td>
</tr>
</tbody>
</table>
## Customization

<table>
<thead>
<tr>
<th>Title</th>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>z/OS Communications Server:</td>
<td>SC27-3666</td>
<td>This document enables you to customize SNA, and includes the following information:</td>
</tr>
<tr>
<td>SNA Customization</td>
<td></td>
<td>• Communication network management (CNM) routing table</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Logon-interpret routine requirements</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Logon manager installation-wide exit routine for the CLU search exit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• TSO/SNA installation-wide exit routines</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• SNA installation-wide exit routines</td>
</tr>
</tbody>
</table>

## Writing application programs

<table>
<thead>
<tr>
<th>Title</th>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>z/OS Communications Server:</td>
<td>SC27-3660</td>
<td>This document 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 document to adapt your existing applications to communicate with each other using sockets over TCP/IP.</td>
</tr>
<tr>
<td>IP Sockets Application Programming Interface Guide and Reference</td>
<td></td>
<td></td>
</tr>
<tr>
<td>z/OS Communications Server:</td>
<td>SC27-3649</td>
<td>This document is for programmers who want to set up, write application programs for, and diagnose problems with the socket interface for CICS® using z/OS TCP/IP.</td>
</tr>
<tr>
<td>IP CICS Sockets Guide</td>
<td></td>
<td></td>
</tr>
<tr>
<td>z/OS Communications Server:</td>
<td>SC27-3653</td>
<td>This document is for programmers who want application programs that use the IMS™ TCP/IP application development services provided by the TCP/IP Services of IBM.</td>
</tr>
<tr>
<td>IP IMS Sockets Guide</td>
<td></td>
<td></td>
</tr>
<tr>
<td>z/OS Communications Server:</td>
<td>SC27-3659</td>
<td>This document 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 z/OS operating system, TCP/IP protocols, and IBM Time Sharing Option (TSO) is recommended.</td>
</tr>
<tr>
<td>IP Programmer’s Guide and Reference</td>
<td></td>
<td></td>
</tr>
<tr>
<td>z/OS Communications Server:</td>
<td>SC27-3674</td>
<td>This document describes how to use SNA 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.</td>
</tr>
<tr>
<td>SNA Programming</td>
<td></td>
<td></td>
</tr>
<tr>
<td>z/OS Communications Server:</td>
<td>SC27-3669</td>
<td>This document describes how to use the SNA LU 6.2 application programming interface for host application programs. This document 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 document.)</td>
</tr>
<tr>
<td>SNA Programmer’s LU 6.2 Guide</td>
<td></td>
<td></td>
</tr>
<tr>
<td>z/OS Communications Server:</td>
<td>SC27-3670</td>
<td>This document provides reference material for the SNA LU 6.2 programming interface for host application programs.</td>
</tr>
<tr>
<td>SNA Programmer’s LU 6.2 Reference</td>
<td></td>
<td></td>
</tr>
<tr>
<td>z/OS Communications Server:</td>
<td>SC27-3647</td>
<td>This document describes how applications use the communications storage manager.</td>
</tr>
<tr>
<td>CSM Guide</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
This document describes the Common Management Information Protocol (CMIP) programming interface for application programmers to use in coding CMIP application programs. The document provides guide and reference information about CMIP services and the SNA topology agent.

This document 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.

This document explains how to gather the trace data that is collected and stored in the host processor. It also explains how to use the Advanced Communications Function/Trace Analysis Program (ACF/TAP) service aid to produce reports for analyzing the trace data information.

These documents help you identify an SNA 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.

These documents describe SNA data areas and can be used to read an SNA dump. They are intended for IBM programming service representatives and customer personnel who are diagnosing problems with SNA.

This document describes the ELM, IKT, IST, IUT, IVT, and USS messages. Other information in this document includes:
• Command and RU types in SNA messages
• Node and ID types in SNA messages
• Supplemental message-related information

This volume contains TCP/IP messages beginning with EZA.

This volume contains TCP/IP messages beginning with EZB or EZD.

This volume contains TCP/IP messages beginning with EZY.

This volume contains TCP/IP messages beginning with EZZ and SNM.

This document describes codes and other information that appear in z/OS Communications Server messages.
Index

A
abnormal termination 19
accessibility 111
accessing another user's data 5
address space identifier (ASID) 3, 10
API
  CSM 1
application design considerations 5
application programming interface
  CSM 1
  ASID (address space identifier) 3, 10
ASSIGN_BUFFER macroinstruction
description 14, 23
effect on DISPLAY CSM command 3
authorization 5

B
borrowers 5
buffer list
  contents 9
  purpose 9
buffer ownership 2
buffer pool tokens 8
buffer pools
  sizes 1
  types 1
buffer return exit routine 9, 18
buffer states 9
buffer types 9
buffers, sharing 3
BUFLIST parameter 9
BUFTYPE parameter 9

C
CHANGE_OWNER macroinstruction 11, 29
changing buffer ownership 29
CLEAR parameter 12
clearing data from buffers 12
codes
  summary of reason codes 79
  summary of return codes 79
commands
  DISPLAY CSM 2
  DISPLAY TRL 4
  MODIFY CSM 3
Communications Server for z/OS, online information xvii
communications storage manager
defining 2
initializing 2
installing 2
starting 2
system definition 2
communications storage manager, see CSM 1
contraction, CSM buffer pools 18
COPY_DATA macroinstruction
description 13, 34
storage key differences 5
copying data from a CSM buffer 34
CREATE_POOL 40
CREATE_POOL macroinstruction 8, 40
creating a buffer pool 8
CSM 83
  API 1
    application programming interface 1
data space 2
defining 2
definition of 1
initializing 2
installing 2
overview 1
starting 2
  system definition 2
CSM buffer list
  contents 9
  purpose 9
CSM dump information 5
CSM dumping information 15
CSM parmib member 2, 17
CSM resource status area 83
CSM storage usage 4

D
data handling 5
data space 6
data space in CSM 2
defining storage limits 17
DELETE_POOL macroinstruction 12, 46
design considerations, application 5
diagnosing CSM problems 4
disability 111
DISPLAY CSM command 2
DISPLAY TRL command 2, 4
DNS, online information xviii
documentation 6
DSECTS
  IVTBUFL (CSM Buffer List Entry) 81
  IVDATSP (CSM Data Space Information) 83
  IVTSTAFA (CSM Resource Status Area) 83
DUMP_INFO macroinstruction 50
dumps 5, 15

E
ECSA (extended common service area) 6
EXPBUF parameter 17
extended common service area (ECSA) 6

F
FIX_BUFFER macroinstruction 52
fixed buffers, guaranteed 10
formatting 5
FREE_BUFFER macroinstruction 12, 57
FREERTN parameter 18
FREEETO parameter 12

© Copyright IBM Corp. 2000, 2015
GET_BUFFER macroinstruction
  description  63
  original requester  5
  usage  9
getting dump information  50
getting storage  8, 63
GTF trace facility  4

high performance data transfer (HPDT)
  changing buffer ownership  11
  description  1
  design considerations  5
HPDT (high performance data transfer)
  changing buffer ownership  11
  description  1
  design considerations  5

IBM Software Support Center, contacting  x
Information APARs  xv
INITBUF parameter  17

key, storage  1, 5
keyboard  111

license, patent, and copyright information  115

macroinstructions
  ASSIGN_BUFFER request  14, 23
  CHANGE_OWNER request  11, 29
  COPY_DATA request  13, 34
  CREATE_POOL request  8, 40
  DELETE_POOL request  12, 46
  DUMP_INFO request  16, 50
  FIX_BUFFER request  52

normal termination  19

Obtaining CSM dumping information  csmdump.dita  15
obtaining storage  8, 63
original requester  5
OWNERID  3, 10
ownership of buffers, changing  29
ownership, buffer  2

paged buffers, eligible  10
paged buffers, guaranteed  10
parmlib member  2, 17
performance monitor interface  4
PMI  4
pool registration  40
pools, buffer
  sizes  1
  types  1
prerequisite information  xv

reason codes, summary  79
registering a buffer pool  8
registration  40
registration to use a CSM buffer pool  8
requester, original  5
requesting buffer pools  8, 63
RESOURCE_STATS  74
RESOURCE_STATS macroinstruction
  description  6, 74
  usage  16
  responsibilities of ownership  5, 11
  RETPTOKN parameter  8
  return and reason codes  79
  return codes, summary  79
returning buffers to application  9, 18
returning buffers to CSM  12, 57
RFC (request for comments)  85
  accessing online  xvii

sharing buffers  3, 14, 23

z/OS V2R2.0 Communications Server: CSM Guide
shortcut keys 111
SNA protocol specifications 109
softcopy information xv
SRCLIST parameter 9
storage
  constraint 3
  DISPLAY CSM command 2
  DISPLAY TRL command 4
  MODIFY CSM command 4
  monitoring 2
storage constraint 16
storage key 1, 5
storage limits 2
storage limits, defining 17
storage usage, CSM 4
storage use, critical 16
summary of changes xxi
syntax diagram, how to read xi

T
TARGLIST parameter 9
task control block 10
TASKID 10
TCP/IP
  online information xvii
  protocol specifications 85
Technotes xv
termination 19
tokens
  for buffer pools 8
  for individual buffers 9
  tracing 4
trademark information 123
tuning buffer pools 8

U
usage 11, 12, 13, 14, 16

V
VIT 4
VTAM internal trace 4
VTAM, online information xvii

Z
z/OS Basic Skills Information Center xv
z/OS, documentation library listing 125
Communicating your comments to IBM

If you especially like or dislike anything about this document, you can send us comments electronically by using one of the following methods:

**Internet email:**
comsvrcf@us.ibm.com

**World Wide Web:**

If you would like a reply, be sure to include your name, address, and telephone number. Make sure to include the following information in your comment or note:

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

Feel free to comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this document. 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.