z/OS

DFSMStvs Administration Guide

Version 2 Release 1

GC52-1388-00

z/OS

DFSMStvs Administration Guide

Version 2 Release 1

GC52-1388-00

Note
Before using this information and the product it supports, read the information in "Notices" on page 357.

This edition applies to Version 2 Release 1 of z/OS (5650-ZOS) and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Figures

Tables

About this document

This document is intended for system programmers, storage administrators, and operators who are responsible for customizing, administering, and operating z/OS DFSMStvs. Primarily, this document contains reference and guidance information for many of the major tasks that a user might need to perform when using DFSMS Transactional VSAM Services (DFSMStvs).

This document helps you in these ways:

- Understand certain functions of DFSMS, such as VSAM RLS, and the manner in which DFSMStvs relates to these functions and builds upon them
- Learn about the migration tasks that you must complete
- Be able to perform tasks such as customizing, administering, and operating DFSMStvs, as well as diagnosing problems that might be related to DFSMStvs

For information about accessibility features of z/OS, for users who have a physical disability, see "Accessibility," on page 353.

Required product knowledge

You should understand the following components and features to use this document effectively:

- Data Facility Storage Management Subsystem (DFSMS)
- Data Facility Storage Management Subsystem data facility product (DFSMSdfp)
- Data Facility Storage Management Subsystem data set services (DFSMSdss)
- Multiple Virtual Storage (MVS™)
- Virtual storage access method (VSAM)
- VSAM record-level sharing (VSAM RLS)

z/OS information

This information explains how z/OS references information in other documents and on the web.

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*.

To find the complete z/OS library, including the z/OS Information Center, see z/OS Internet Library (http://www.ibm.com/systems/z/os/zos/bkserv/).

Notational Conventions

This document describes various commands that you can use. A uniform notation describes the syntax of these commands. This notation is not part of the language; it is merely a way of describing the syntax of the commands. The command syntax definitions in this book use the following conventions:

[ ] Brackets enclose an optional entry. You can, but need not, include the entry. Examples follow:

- **[***length***]**
- **[MF=E]**

| An OR sign (a vertical bar) separates alternative entries. You must specify one, and only one, of the entries unless you allow an indicated default. Examples follow:

- **[REREAD|LEAVE]**
- **[***length***|'S']**

{ } Braces enclose alternative entries. You must use one, and only one, of the entries. Examples follow:

- **BFTEK={S|A}**
- **{K|D}**
- **{***address***|S|O}**

Sometimes alternative entries are shown in a vertical stack of braces. An example follows:

```
MACRF={{(R[C|P])   }
       {(W[C|P|L]) }
       {(R[C],W[C])}}
```

In the preceding example, you must choose only one entry from the vertical stack.

... An ellipsis indicates that the entry immediately preceding the ellipsis can be repeated. For example:

- **(***Dcbaddr***,[(***options***)],...)**

UPPERCASE BOLDFACE

Uppercase boldface type indicates entries that you must code exactly as shown. These entries consist of keywords and the following punctuation symbols: commas, parentheses, and equal signs. Examples follow:

- **CLOSE , , , ,TYPE=T**
- **MACRF=(PL,PTC)**

UNDERSCORED UPPERCASE BOLDFACE

Underscored uppercase boldface type indicates the default used if you do not specify any of the alternatives. Examples follow:

- **[EROPT={ACC|SKP|ABE}]**
- **[BFALN={F|D}]**

lowercase italic

Lowercase italic type indicates a value that you supply, according to the specifications and limits for the parameter. Examples follow:

- *number*
- *image-id*
- *count*

How to send your comments to IBM

We appreciate your input on this publication. Feel free to comment on the clarity, accuracy, and completeness of the information or provide any other feedback that you have.

Use one of the following methods to send your comments:

1. Send an email to mhvrcfs@us.ibm.com.
2. Send an email from the "Contact us" web page for z/OS (http://www.ibm.com/systems/z/os/zos/webqs.html).
3. Mail the comments to the following address:
   IBM Corporation
   Attention: MHVRCFS Reader Comments
   Department H6MA, Building 707
   2455 South Road
   Poughkeepsie, NY 12601-5400
   US
4. Fax the comments to us, as follows:
   From the United States and Canada: 1+845+432-9405
   From all other countries: Your international access code +1+845+432-9405

Include the following information:

- Your name and address.
- Your email address.
- Your telephone or fax number.
- The publication title and order number:
  z/OS V2R1.0 DFSMStvs Administration Guide
  GC52-1388-00
- The topic and page number that is related to your comment.
- The text of your comment.

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

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

If you have a technical problem

Do not use the feedback methods that are listed for sending comments. Instead, take one of the following actions:

- Contact your IBM service representative.
- Call IBM technical support.
- Visit the IBM Support Portal at z/OS support page (http://www.ibm.com/systems/z/support/).

z/OS Version 2 Release 1 summary of changes

See the following publications for all enhancements to z/OS Version 2 Release 1 (V2R1):

- *z/OS Migration*
- *z/OS Planning for Installation*
- *z/OS Summary of Message and Interface Changes*
- *z/OS Introduction and Release Guide*

Chapter 1. Evaluating, planning, and installing DFSMStvs

This topic provides information to help you evaluate, plan for, and install DFSMStvs.

Evaluating and planning for DFSMStvs

This topic provides information to help you evaluate your installation's readiness to use DFSMStvs and to help you plan for using it.

Software dependencies

The following software is required for DFSMStvs, which is a licensed feature of z/OS Version 1 Release 4 or above.

z/OS

DFSMStvs is available on z/OS Version 1 Release 4 or above.

Use of DFSMStvs requires services that the system logger and recoverable resource management services (RRMS) provide. The system logger and RRMS were shipped in earlier releases of z/OS.

DFSMS

DFSMStvs requires the DFSMS component of z/OS Version 1 Release 4 or above.

Changes to the lock information stored in the coupling facility for DFSMStvs require compatibility PTFs for any DFSMS 1.3 systems that might be running VSAM record-level sharing (VSAM RLS) in a sysplex where DFSMStvs is also used.

New IDCAMS SHCDS commands that are routed to all of the systems in the sysplex require a compatibility PTF to any sharing DFSMS 1.3 systems to handle these commands correctly.

CICS Transaction Server

To exploit the functions that VSAM RLS provides, you need CICS® Transaction Server (CICS TS) Version 1 Release 1 or later, which provides these features:

- VSAM RLS support, enabling CICS systems to share the VSAM data sets directly rather than function shipping to a file owning region
- Application programming interface (API) extensions for VSAM RLS
- The CICS TS log manager
- The CICS TS recovery manager

If you are running DFSMStvs and CICS TS, a compatibility PTF is required for CICS TS because DFSMStvs support requires some changes to the VSAM problem determination information used to report locking conflicts. The problem determination information is used by both DFSMStvs and CICS TS.

CICS VSAM Recovery

DFSMStvs supports the data set forward recovery option introduced into VSAM cluster definitions by VSAM RLS. DFSMStvs logs the redo records to system logger log streams. These log streams are shared across all DFSMStvs instances and CICS systems. This provides sysplex-wide, merged log streams, which you can use as

input to CICS VSAM Recovery (CICSVR) or other forward recovery products. To enable these products to process DFSMStvs forward recovery log records with minimal changes, the log records were designed to be as similar to those written by CICS as possible.

CICSVR version 2 release 3 or higher is needed.

Global Resource Serialization

Global Resource Serialization (GRS) or an equivalent product is required to ensure cross-system serialization of VSAM resources and other DFSMS control structures altered by DFSMStvs and VSAM RLS.

z/OS Security Server

You need Resource Access Control Facility (RACF®), a component of z/OS Security Server 2.1, or an equivalent product to provide authorization for and protection of data and system resources:

- An RACF FACILITY class profile, STGADMIN.IGWSHCDS.REPAIR, controls access to the IDCAMS SHCDS command functions, which you can use to list outstanding SMSVSAM recovery requirements and control that recovery. If you do not use RACF to secure your installation, ensure that your security product supports the new class name and the new FACILITY class profile.
- The SMSVSAM address space must be given authority to use the system logger log streams.
- To allow DFSMStvs to do automated BACKOUT and RETRY recovery for transactions, the region must be PRIVILEGED or TRUSTED, or the SMSVSAM address space must be part of a class that has UPDATE access to the data sets to be recovered.

If you use RACF as your external security manager, you need a minimum of RACF Version 2 Release 1 (5695-039) and PTFs to provide the required support for the LOGSTRM general resource class and to enable general resource class profile names for journals to be up to 17 characters long.

IMS

If a batch job uses both VSAM data sets and IMS™ databases, IMS version 6 or higher is needed so that IMS uses the MVS RRS services. This will allow two-phase commit to be done for both VSAM and IMS requests in the batch job.

DB2

If a batch job is to use both VSAM data sets and DB2® tables, DB2 version 5 or higher is needed so that DB2 will use MVS RRS services. This makes use of the DB2 Recoverable Resource Services Attachment Facility, RRSAF.

Language products

Runtime library support for COBOL, PL/I and C is provided by the Language Environment® (LE) component of z/OS. VSAM RLS support was originally provided in LE version 1 release 5. Because LE is part of the z/OS base elements, it now has the same version and release number as the level of z/OS with which it is shipped.

Language Environment: LE includes the following support for DFSMStvs:

- Conforming with DFSMStvs restrictions
- Handling new VSAM error codes, as necessary
- For languages that support dynamic allocation, including facilities in the language to specify the CRE read integrity option

- Externalizing the TIMEOUT value in the RPL to enable lock requests to time out if they cannot be satisfied within the time specified

The key item for support of DFSMStvs is the libraries used, not the compiler. So, while VS/COBOL II programs will work if they use the correct Language Environment libraries, they will not work with the VS COBOL II libraries.

Both COBOL and C have the ability to create self-contained load modules. At a minimum, relinking such programs is required for them to use the DFSMStvs support in LE.

Programming languages and environments: You can use these programming languages and environments with DFSMStvs:

- High-Level Assembler/MVS (5696-234)
- IBM COBOL for z/OS version 2 (5648-A25)
- IBM COBOL for MVS version 1 (5688-197)

  Restricted OS/VS COBOL language statements that result in a call to MVS GETMAIN services worked on earlier releases. Now, a COBOL verb that results in an MVS GETMAIN will cause an 0C4 abend. In these cases, it is not the application program itself that appears to cause the 0C4. The end-of-service date for this compiler was 12/31/2001.
- VS/COBOL II (5668-958 and 5688-023). No support is provided for NORES programs (CICS or batch). The Language Environment libraries must be used. VS/COBOL II went out of service in March 2001.
- C/370™ Version 1, Release 2 or later (5688-040)
- IBM C/C++ for MVS/ESA Version 3, Release 1 or later (5655-121)
- VisualAge® PL/I for z/OS (5655-B22)
- IBM PL/I for MVS version 1 (5688-235)
- OS PL/I Optimizing Compiler Version 2, Release 1 or later (5668-910)
- OS PL/I Optimizing Compiler Version 1, Release 5.1 or later (5734-PL1)

CICS also supports IBM SAA AD/Cycle Language Environment/370 Version 1, Release 1 and Release 2 runtime environment (5688-198) with the following COBOL, C/370, and PL/I SAA AD/Cycle compilers:

- SAA AD/Cycle COBOL/370 (5688-197)
- SAA AD/Cycle C/370 (5688-216)
- SAA AD/Cycle PL/I (5688-235)

Processing restrictions

The following restrictions apply to DFSMStvs processing:

- VSAM RLS and DFSMStvs do not support these features:
  - Linear data sets
  - Control interval access (CNV) for any data set organization.
  - Addressed access to a KSDS
  - Access to key range data sets
  - Access to clusters with the IMBED attribute
  - Access to temporary data sets
  - Access using the ISAM compatibility interface
  - Open of the individual components of a cluster
  - Direct open of an alternate index

  - Use of GETIX and PUTIX
  - Checkpoint/restart
  - Catalogs accessed in RLS or DFSMStvs mode
  - VVDSs access in RLS or DFSMStvs mode
  - The ACBSDS specification
  - Implicit positioning at OPEN; an explicit POINT or GET NSP is required
  - Hiperbatch
  - CFX (ignored; NFX is assumed)
  - DDN/DSN (ignored)
  - DFR (ignored; NDF is assumed for direct requests that do not specify NSP)
  - Improved control interval processing
  - Control blocks in common (CBIC)
  - UBF (user buffering)
  - SHRPOOL (ignored)
- *z/OS DFSMStvs Planning and Operating Guide* describes some additional restrictions.

Migrating to z/OS Version 1 Release 4

This topic provides information that you need to consider before migrating your z/OS system to use DFSMStvs and describes migration tasks. For more information on migrating DFSMS to z/OS Version 1 Release 4, see *z/OS Migration*.

DFSMStvs enables you to run batch VSAM processing concurrently with CICS online transactions. You can run multiple batch jobs and online transactions against the same VSAM data, in data sets defined as recoverable, with concurrent updates. DFSMStvs offers these features:

- Concurrent shared update of VSAM recoverable data sets across CICS transactions and batch applications
- Ability to run multiple batch jobs concurrently instead of serially
- Logging, commit, and backout functions
- 24 x 7 CICS Transaction Server (TS) applications
- Data sharing across CICS TS applications, batch applications, and local or distributed object-oriented (OO) applications

Building on the functionality of VSAM RLS, DFSMStvs provides transactional capability within the file system. If a batch job fails during concurrent shared updates of recoverable VSAM data sets, DFSMStvs provides the services to back out any changes that the batch job made automatically, restoring the data to the state it was in at the last synchronization point (commit or back out).

You can use DFSMStvs in two major areas:

- *Transactional processing* provides data sharing for recoverable resources. Transactional processing ensures that the data is kept in sync while multiple parties update the data and ensures data integrity in the event of a job or system failure. Products such as CICS, IMS, and DB2 provide a transactional environment.
- *Transactional recovery* isolates the changes made to recoverable resources into logical units of work that are recoverable. When a transaction makes a change,

only that transaction can update the changed data. After DFSMStvs commits the transaction, all data associated with that logical unit of work is available to other transactions for update.

DFSMStvs also supports forward recovery logging for data sets that are defined as forward recoverable (the LOG parameter value is ALL). If data is lost or damaged, you can restore it from a backup, and you can use a forward recovery utility such as CICS VSAM Recovery (CICSVR) to reapply changes that were made since the last backup.

CICSVR automates the recovery of lost or damaged VSAM data sets. It determines what CICS journals and VSAM backups are needed, and it constructs the recovery jobs. CICSVR provides automated complete recovery, forward recovery, and backout functions. CICSVR VSAM batch logging is available with CICS VSAM Recovery V3R1. For more information about CICSVR, see the CICS Transaction Server for z/OS Information Center at http://publib.boulder.ibm.com/infocenter/cicsts/v3r1/index.jsp .

Implementing DFSMStvs could affect the following areas of your processing environment.

Implementing DFSMStvs could affect the following areas of your processing environment.

Area	Considerations
System customization	Updated IGDSMS*xx* member of SYS1.PARMLIB DFSMStvs log stream definitions and initialization parameters
Storage administration	DFSMSdfp: New DFSMStvs parameters in the IGDSMS*xx* member of SYS1.PARMLIB and in the MVS SET SMS, VARY SMS, and SETSMS commands DFSMSdss: Data sets designated by DFSMStvs as eligible for backup-while-open processing
Operations	None
Auditing	None
Magnetic tapes	Block sizes of up to 262 144 bytes
Application development	Job control language (JCL) statements: You can specify the RLS parameter with the CRE (consistent read explicit) subparameter. Access method services: The ALLOCATE, ALTER, DEFINE ALTERNATEINDEX, DEFINE CLUSTER, and DEFINE PATH commands have been updated for DFSMStvs. Macros: The ACB, GENCB, GET UPD, IDALKADD, RPL, POINT, and PUT UPD macros have been updated for DFSMStvs.
Messages and codes	Many MVS system messages have been added or changed for DFSMStvs, including new routing and descriptor codes. For descriptions of these messages and codes, see *z/OS MVS System Messages, Vol 6 (GOS-IEA)*, *z/OS MVS System Messages, Vol 8 (IEF-IGD)*, and *z/OS MVS System Messages, Vol 9 (IGF-IWM)*.
General use	Before using DFSMStvs, evaluate which applications are good candidates and which applications cannot support the necessary changes.

SYS1.PARMLIB changes for DFSMStvs

Table 1 on page 6 lists the changes to the IGDSMS*xx* member of SYS1.PARMLIB for DFSMStvs.

Table 1. IGDSMSxx changes in SYS1.PARMLIB. This table lists the changes to the IGDSMSxx member of SYS1.PARMLIB for DFSMStvs.

Member name	Release	Description	Related support
IGDSMS*xx*	z/OS V1R4 or above.	**New parameter: AKP** Specifies the activity keypoint trigger value, which is the number of logging operations between taking keypoints, for one or more DFSMStvs instances.	*z/OS MVS Initialization and Tuning Reference* and *z/OS DFSMStvs Planning and Operating Guide*
		New parameter: LOG_OF_LOGS Specifies the log stream to use as the log of logs.	
		New parameter: MAXLOCKS Specifies the maximum number of unique lock requests that a single unit of recovery can make.	
		New parameter: QTIMEOUT Specifies the quiesce exit timeout value.	
		Changed parameter: RLSTMOUT Specifies a timeout value for DFSMStvs requests for required locks.	
		New parameter: SYSNAME Specifies the name or names of the systems on which DFSMStvs instances are to run.	
		New parameter: TVSNAME Specifies the identifier or identifiers of DFSMStvs instances that are to run in the sysplex.	
		New parameter: TV_START_TYPE Specifies the type of start that each instance of DFSMStvs is to perform.	

JCL changes for DFSMStvs

Table 2 lists changes to JCL for DFSMStvs support.

Table 2. Changed JCL statements. This table lists changes to JCL for DFSMStvs support.

JCL statement	Release	Description	Related support
EXEC	z/OS V1R4 or above	**Changed parameter: RLSTMOUT** Specifies a timeout value for DFSMStvs requests for required locks.	*z/OS MVS JCL Reference* and *z/OS DFSMStvs Planning and Operating Guide*
RLS	z/OS V1R4 or above	**New parameter: CRE** Obtains a shared lock for VSAM RLS.	*z/OS MVS JCL Reference* and *z/OS DFSMStvs Planning and Operating Guide*

System command changes for DFSMStvs

Table 3 lists new and changed system-level commands related to DFSMStvs support.

Table 3. New and changed system-level commands. This table lists new and changed system-level commands related to DFSMStvs support.

Command name	Release	Description	Related support
DISPLAY SMS	z/OS V1R4 or above	**New parameter: DSNAME** Displays all jobs that currently access the data set using DFSMStvs access on the systems within the sysplex.	*z/OS MVS System Commands* and *z/OS DFSMStvs Planning and Operating Guide*
		New parameter: JOB Displays information about a particular job that is using DFSMStvs services on one of the systems in the sysplex.	
		New parameter: LOG Displays information about a log stream that DFSMStvs is currently using on one of the systems in the sysplex.	
		New parameter: OPTIONS Displays all of the SMS parameters and their status at the time this command is issued. When DFSMStvs is running on the system, the output of this command includes DFSMStvs information.	
		New parameter: SHUNTED Displays the entries currently contained in the shunt logs of the systems in the sysplex.	
		New parameter: TRANVSAM Displays information about the instance of DFSMStvs on this system or on all systems in the sysplex.	
		New parameter: URID Displays information about an active unit of recovery within the sysplex.	

Table 3. New and changed system-level commands (continued). This table lists new and changed system-level commands related to DFSMStvs support.

Command name	Release	Description	Related support
SET SMS	z/OS V1R4 or above	**New parameter: AKP** Specifies the activity keypoint trigger value, which is the number of logging operations between taking keypoints, for one or more DFSMStvs instances.	*z/OS MVS System Commands* and *z/OS DFSMStvs Planning and Operating Guide*
		New parameter: LOG_OF_LOGS Specifies the log stream that is to be used as the log of logs.	
		New parameter: MAXLOCKS Specifies the maximum number of unique lock requests that a single unit of recovery can make.	
		New parameter: QTIMEOUT Specifies the quiesce exit timeout value.	
		Changed parameter: RLSTMOUT Specifies a timeout value for DFSMStvs requests for required locks.	
		New parameter: SYSNAME Specifies the name or names of the systems on which DFSMStvs instances are to run.	
		New parameter: TVSNAME Specifies the identifier or identifiers of DFSMStvs instances that are to run in the sysplex	
		New parameter: TV_START_TYPE Specifies the type of start that each instance of DFSMStvs is to perform.	
SETSMS	z/OS V1R4 or above	**New parameter: AKP** Changes the activity keypoint trigger value, which is the number of logging operations between taking keypoints, for one or more DFSMStvs instances.	*z/OS MVS System Commands* and *z/OS DFSMStvs Planning and Operating Guide*
		New parameter: MAXLOCKS Changes the maximum number of unique lock requests that a single unit of recovery can make.	
		New parameter: QTIMEOUT Changes the quiesce exit timeout value.	
		Changed parameter: RLSTMOUT Changes a timeout value for DFSMStvs requests for required locks.	

Table 3. New and changed system-level commands (continued). This table lists new and changed system-level commands related to DFSMStvs support.

Command name	Release	Description	Related support
VARY SMS	z/OS V1R4 or above	**New parameter: LOG** Enables, quiesces, or disables DFSMStvs access to a log stream.	*z/OS MVS System Commands* and *z/OS DFSMStvs Planning and Operating Guide*
		New parameter: SMSVSAM,SPHERE Quiesces or unquiesces a data set for DFSMStvs or RLS access.	
		New parameter: TRANVSAM Enables, quiesces, or disables a DFSMStvs instance or all DFSMStvs instances in the sysplex.	
		New parameter: TRANVSAM(*nnn*),PEERRECOVERY Starts or stops peer recovery processing for a failed instance of DFSMStvs.	

Access method services

Table 4 lists changes to access method services (IDCAMS) commands related to DFSMStvs support.

Table 4. Changed IDCAMS commands. This table lists changes to access method services (IDCAMS) commands related to DFSMStvs support.

Command name	Release	Description	Related support
ALLOCATE	z/OS V1R4 or above	**Changed parameter: BWO(TYPECICS)** The TYPECICS option of the BWO parameter specifies backup-while-open (BWO) in a DFSMStvs environment. For RLS processing, this parameter activates BWO processing for DFSMStvs.	*z/OS DFSMS Access Method Services Commands* and *z/OS DFSMStvs Planning and Operating Guide*
		Changed parameter: DATACLAS CISIZE For DFSMStvs, specification of *n**2 KB avoids wasting space in the coupling facility cache structure.	
		Changed parameter: SHAREOPTIONS When you use DFSMStvs access, DFSMS assumes that the value of SHAREOPTIONS is (3,3).	

Table 4. Changed IDCAMS commands (continued). This table lists changes to access method services (IDCAMS) commands related to DFSMStvs support.

Command name	Release	Description	Related support
ALTER	z/OS V1R4 or above	**Changed parameter: BWO(TYPECICS)** The TYPECICS option of the BWO parameter specifies backup-while-open (BWO) in a DFSMStvs environment. For RLS processing, this activates BWO processing for DFSMStvs.	*z/OS DFSMS Access Method Services Commands* and *z/OS DFSMStvs Planning and Operating Guide*
		Changed parameter: LOG Establishes whether the sphere to be accessed with DFSMStvs is recoverable or nonrecoverable. LOG(UNDO) specifies that changes to the sphere accessed in DFSMStvs mode can be backed out using an external log. DFSMStvs considers the sphere recoverable. LOG(ALL) specifies that changes to the sphere accessed in DFSMStvs mode can be backed out and forward recovered using an external log. DFSMStvs considers the sphere recoverable. LOGSTREAMID must also be defined.	
		Changed parameter: LOGSTREAMID Changes or adds the name of the DFSMStvs forward recovery log stream, for all components in the VSAM sphere.	
DEFINE ALTERNATEINDEX	z/OS V1R4 or above	**Changed parameter: BUFFERSPACE** When you use DFSMStvs access, DFSMS ignores the BUFFERSPACE parameter.	*z/OS DFSMS Access Method Services Commands* and *z/OS DFSMStvs Planning and Operating Guide*
		Changed parameter: CONTROLINTERVALSIZE For DFSMStvs, specification of *n**2K avoids wasting space in the coupling facility cache structure.	
		Changed parameter: KEYRANGES You cannot open key range data sets for DFSMStvs processing because DFSMS no longer supports this parameter.	
		Changed parameter: SHAREOPTIONS When you use DFSMStvs access, DFSMS assumes that the value of SHAREOPTIONS is (3,3).	
		Changed parameter: WRITECHECK When you use DFSMStvs access, DFSMS ignores the WRITECHECK parameter.	

Table 4. Changed IDCAMS commands (continued). This table lists changes to access method services (IDCAMS) commands related to DFSMStvs support.

Command name	Release	Description	Related support
DEFINE CLUSTER	z/OS V1R4 or above	**Changed parameter: BUFFERSPACE** When you use DFSMStvs access, DFSMS ignores the BUFFERSPACE parameter.	*z/OS DFSMS Access Method Services Commands* and *z/OS DFSMStvs Planning and Operating Guide*
		Changed parameter: BWO(TYPECICS) The TYPECICS option of the BWO parameter specifies backup-while-open (BWO) in a DFSMStvs environment. For RLS processing, this activates BWO processing for DFSMStvs.	
		Changed parameter: CONTROLINTERVALSIZE For DFSMStvs, specification of *n**2K avoids wasting space in the coupling-facility cache structure.	
		Changed parameter: KEYRANGES You cannot open key range data sets for DFSMStvs processing because DFSMS no longer supports this parameter.	
		Changed parameter: LOG Establishes whether the sphere to be accessed with DFSMStvs is recoverable or nonrecoverable. LOG(UNDO) specifies that changes to the sphere accessed in DFSMStvs mode can be backed out using an external log. DFSMStvs considers the sphere recoverable. LOG(ALL) specifies that changes to the sphere accessed in DFSMStvs mode can be backed out and forward recovered using an external log. DFSMStvs considers the sphere recoverable. LOGSTREAMID must also be defined. If you use LOG(NONE), DFSMStvs considers the sphere to be nonrecoverable.	
		Changed parameter: LOGSTREAMID Changes or adds the name of the DFSMStvs forward recovery log stream, for all components in the VSAM sphere.	

Table 4. Changed IDCAMS commands (continued). This table lists changes to access method services (IDCAMS) commands related to DFSMStvs support.

Command name	Release	Description	Related support
DEFINE CLUSTER (continued)		**Changed parameter: SHAREOPTIONS** When you use DFSMStvs access, DFSMS assumes that the value of SHAREOPTIONS is (3,3).	
		Changed parameter: WRITECHECK When you use DFSMStvs access, DFSMS ignores the WRITECHECK parameter.	
DEFINE PATH	z/OS V1R4 or above	**Changed parameter: NOUPDATE** Has the same meaning for DFSMStvs as it does for RLS.	*z/OS DFSMS Access Method Services Commands*
SHCDS	z/OS V1R4 or above	**Changed parameter: LISTDS** A new, optional JOBS keyword returns a list of the jobs that currently access the data set in DFSMStvs mode.	"Listing and controlling SMSVSAM recovery" on page 52 and *z/OS DFSMStvs Planning and Operating Guide*
		New parameter: LISTSHUNTED Lists information about work that was shunted due to an inability to complete a syncpoint (commit or backout) for a data set, a unit of recovery, or all shunted units of recovery.	
		New parameter: PURGE Discards log entries and releases the associated locks, for use when a data set is damaged and you cannot restore it to a state that is consistent with the log entries.	
		New parameter: RETRY Retries the syncpoint, for use when you can restore a data set to a state that is consistent with the log entries.	

Macros

Table 5 lists new and changed executable macros.

Table 5. DFSMSdfp: Summary of changed executable macros. This table lists new and changed executable macros.

Macro name	Release	Description	Related support
ACB	z/OS V1R4 or above	**Changed macro: CTRLACB=** Specifies whether the opened ACB is to be used as a control ACB. For applications using DFSMStvs, the value of this parameter should not be YES.	"Coding VSAM macros" on page 131 and *z/OS DFSMStvs Planning and Operating Guide*
		Changed macro: MACRF= The RLS subparameter enables DFSMStvs access to a VSAM data set, as well as RLS access. VSAM uses cross-system record-level locking as opposed to CI locking, uses the CF for buffer consistency, and manages a system-wide local cache. DFSMStvs access requires the NUB subparameter, which specifies that management of I/O buffers is left up to VSAM. DFSMStvs does not support ADR access to a KSDS. DFSMStvs ignores the CFX subparameter and assumes that the NFX subparameter is in effect. For DFSMStvs, the NRM subparameter does not allow the direct open of an alternate index. DFSMStvs does not affect key processing, which the KEY subparameter specifies. The AIX®, CNV, ICI, UBF subparameters are invalid for DFSMStvs. DFSMStvs ignores the DDN, DFR, DSN, LEW, NDF, NLW subparameters.	
		Changed macro: RLSREAD= For DFSMStvs access, the new CRE keyword specifies consistent read explicit so that an application can inhibit update or erase of a record by other transactions or applications until commit or backout. CRE can be specified only with MACRF=RLS.	
		Changed macro: RMODE31= If MACRF=RLS is specified, RMODE31=ALL is assumed. For RLS and DFSMStvs, VSAM control blocks and buffers are located in a data space owned by the SMSVSAM server address space and are not directly addressable.	
		DFSMStvs ignores the MAREA parameter.	

Table 5. DFSMSdfp: Summary of changed executable macros (continued). This table lists new and changed executable macros.

Macro name	Release	Description	Related support
GENCB	z/OS V1R4 or above	**Changed macro: CTRLACB=** Specifies whether the opened ACB is to be used as a control ACB. For applications using DFSMStvs, the value of this parameter should not be YES. **Changed macro: RLSREAD=** For DFSMStvs access, the new CRE keyword specifies consistent read explicit so that an application can inhibit update or erase of a record by other transactions or applications until commit or backout. CRE can be specified only with MACRF=RLS. **Changed macro: RMODE31=** For DFSMStvs, VSAM control blocks and buffers are located in a dataspace owned by the SMSVSAM server address space and are not directly addressable. RMODE31=ALL, which specifies that VSAM control blocks and I/O buffers are obtained above 16 megabytes, is assumed for DFSMStvs processing. **Changed macro: STRNO=** For DFSMStvs, STRNO is ignored and strings are dynamically acquired up to a limit of 1024. DFSMStvs ignores the BSTRNO, BUFND, BUFNI, BUFSP, JRNAD, MAREA, MLEN, SHRPOOL, SUBSYSNM, and UPAD parameters.	"Coding VSAM macros" on page 131

Messages and codes

For DFSMStvs descriptions of VSAM return and reason codes, see "Understanding VSAM macro return and reason codes" on page 190.

For descriptions of new and changed DFSMSdfp messages and return and reason codes for DFSMStvs, refer to *z/OS MVS System Messages, Vol 6 (GOS-IEA)*, *z/OS MVS System Messages, Vol 8 (IEF-IGD)*, and *z/OS MVS System Messages, Vol 9 (IGF-IWM)*.

For a listing of other new and changed DFSMSdfp messages and return and reason codes, see *z/OS Summary of Message and Interface Changes*.

Migration tasks

You need to consider each migration task in the following list to implement this change. **Required** tasks apply to any DFSMS installation enabling the function. **Optional** tasks apply to only specified operating environments or to situations where there is more than one way to set up or enable the function. For more details on the procedures associated with a given task, see the procedure reference (column 3).

This table lists migration tasks.

System-level tasks	Condition	Procedure reference
Update the IGDSMS*xx* member of SYS1.PARMLIB with DFSMStvs parameters	Required	*z/OS MVS Initialization and Tuning Reference*
Define resources for DFSMStvs: • Create the coupling facility structures for storing log stream data • Define the maximum number of log streams for each structure • Create the log stream definitions that DFSMStvs needs for its system logs • Define staging data sets • Authorize access to system log streams • Ensure that at least two active data sets and one spare, sharing control data set are always placed for maximum availability	Required	*z/OS DFSMStvs Planning and Operating Guide*

This table lists migration tasks.

Application-level tasks	Condition	Procedure reference
For DFSMStvs applications that use COBOL, PL/I, and C/370 runtime libraries, ensure that the following conditions apply: • The application can use DFSMStvs. Ensure that the application functions correctly in a multiple-update environment before you specify DFSMStvs access. • COBOL, PL/I, and C/370 support DD-based allocation, in which case you can use the JCL RLS parameter to specify the read integrity option. The recoverable data set is open for DFSMStvs access if its catalog entry specifies that logging is requested and if either CRE is requested or the data set is open for output. • If you use DD allocation, you might not need to recompile the application for DFSMStvs. **Recommendation:** Do some conversion of the application to make use of syncpoint processing. • COBOL has the ability to create self-contained load modules.	Optional	• *z/OS DFSMStvs Planning and Operating Guide* • *z/OS MVS JCL Reference* • *z/OS MVS Programming: Resource Recovery*

Additional information

For additional information about new and changed features that apply to DFSMStvs, see the rest of this administration guide and the following publications:

- *z/OS DFSMStvs Planning and Operating Guide*
- *z/OS MVS Initialization and Tuning Reference*
- *z/OS MVS JCL Reference*
- *z/OS MVS Programming: Resource Recovery*
- *z/OS MVS Setting Up a Sysplex*
- *z/OS MVS System Commands*

Installing DFSMStvs

Enabling DFSMStvs on your z/OS system

For information about enabling DFSMStvs on your z/OS system, see "Installation Information" in the z/OS V1R4 PSP upgrade, subsets ZOSGEN and DFSMS.

Coding IGDSMS*xx*

The IGDSMSxx member of SYS1.PARMLIB contains parameters for the initialization and tuning of DFSMStvs. For more information about coding IGDSMSxx, see *z/OS MVS Initialization and Tuning Reference*.

Chapter 2. Administering resources for DFSMStvs

This topic describes how to set up the resources that you need for using DFSMStvs.

Controlling access to VSAM data sets

You can specify the following options to control DFSMStvs access to VSAM data sets:

- VSAM record-level sharing (VSAM RLS)
- Read integrity options
- Timeout value for lock requests

If a VSAM data set is recoverable, DFSMStvs can open the data for input within a transaction. A recoverable VSAM data set is defined with the LOG(UNDO) or LOG(ALL) attribute. For more information about using recoverable VSAM data sets, see *z/OS DFSMStvs Planning and Operating Guide*.

Accessing data sets in DFSMStvs mode

This topic describes the use of VSAM data sets for DFSMStvs. For more information about VSAM data sets, see *z/OS DFSMS Using Data Sets*.

Using VSAM record-level sharing

VSAM record-level sharing (RLS) is an access option for VSAM data sets. This option provides multisystem sharing of VSAM data sets across a z/OS Parallel Sysplex®. VSAM RLS exploits the data sharing technology of the coupling facility (CF) including a CF-based lock manager and a CF cache manager. VSAM RLS uses the CF-based lock manager and the CF cache manager in its implementation of record-level sharing.

RLS is a mode of access to VSAM spheres. RLS is not an attribute of a sphere. It is an access option interpreted at open time. The option is selected either by specifying a new JCL parameter (RLS) or by specifying MACRF=RLS in the ACB. The RLS MACRF option is mutually exclusive with the MACRF NSR (nonshared resources), LSR (local shared resources), and GSR (global shared resources) options. This topic uses the term non-RLS access to distinguish between RLS access and NSR/LSR/GRS access.

Access method services do not use RLS when performing EXPORT, IMPORT, PRINT or REPRO commands. If the RLS keyword is specified in the DD statement of a data set to be opened by access method services, the keyword is ignored and the data set is opened and accessed in non-RLS mode. See “Non-RLS access to VSAM data sets” on page 23 for more information about non-RLS access.

RLS access is supported for KSDS, ESDS, RRDS, and VRRDS data sets. RLS access is supported for VSAM alternate indexes. RLS is not supported for control interval mode access (CNV or ICI) or for UNIX files.

The VSAM RLS functions are provided by the SMSVSAM server. This server resides in a system address space. The address space is created and the server is

started at MVS IPL time. VSAM internally performs cross-address space accesses and linkages between requestor address spaces and the SMSVSAM server address space.

The SMSVSAM server owns two data spaces. One data space is called the SMSVSAM data space. It contains some VSAM RLS control blocks and a system-wide buffer pool. VSAM RLS uses the other data space, called MMFSTUFF, to collect activity monitoring information that is used to produce SMF records. VSAM provides the cross-address space access and linkage between the requestor address spaces and the SMSVSAM address and data spaces. See Figure 1.

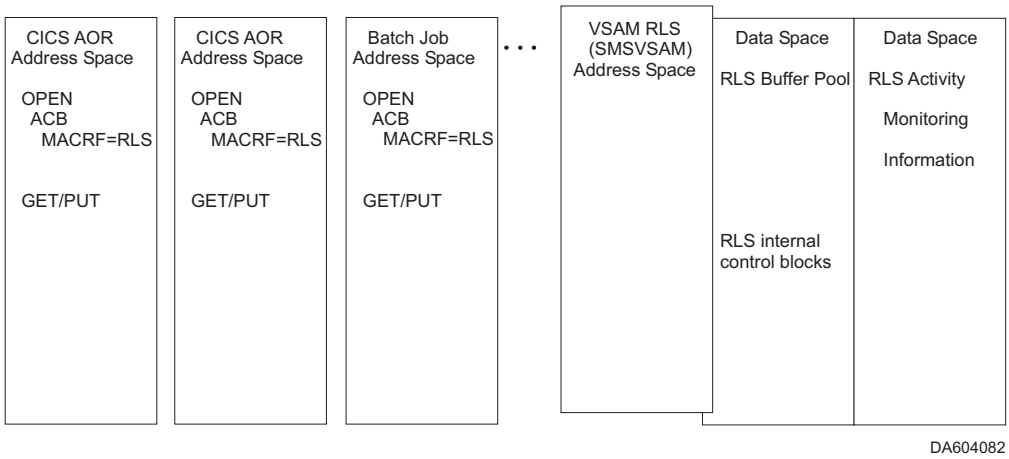


Figure 1. VSAM RLS address and data spaces and requestor address spaces

Record-level sharing CF caching: VSAM record-level sharing allows multiple levels of CF caching for DFSMS cache structures that are defined in the active storage management subsystem (SMS) configuration.

Before z/OS Version 1 Release 3, VSAM RLS placed the data of a VSAM sphere in the CF cache structure only when the sphere's CI size was less than 4K. For CIs greater than 4K, VSAM RLS placed the first 2K in the CF cache (to utilize the cross-invalidate features of the coupling facility). To retrieve a CI that was greater than 4K, VSAM RLS processing would always read the data from a direct access storage device (DASD) volume.

Since z/OS Version 1 Release 3, VSAM RLS has multiple levels of CF caching. The value of the SMS DATACLAS RLS CF Cache Value keyword determines the level of CF caching. The default value, ALL, indicates that RLS caches all data for a sphere in the coupling facility (for both index and data parts of the sphere). If you specify NONE, then RLS caches only the index part of the VSAM sphere. If you specify UPDATESONLY, then RLS caches data in the coupling facility only during write operations.

All active systems in a sysplex must have the greater than 4K CF caching code installed before the function is enabled.

To set up RLS CF caching, use the following values:

- ALL or UPDATESONLY or NONE for the SMS DATACLAS RLS CF Cache Value keyword

To allow greater than 4K caching of DFSMS VSAM data sets open for RLS processing, you need to make the following changes:

- You can change the value of the SMS DATACLAS RLS CF Cache Value keyword if you do not want caching of all VSAM RLS data:

  ALL Indicates that RLS is to cache VSAM index and data components. **ALL** is the default.

  NONE
  Indicates that RLS is to cache only the VSAM index data. The data components are not to be placed in the cache structure.

  UPDATESONLY
  Indicates that RLS is to place only WRITE requests in the cache structure.

- VSAM honors the RLS CF Cache Value keyword only when you specify RLS_MaxCfFeatureLevel(A) and all systems in the sysplex can run the greater than 4K caching code.

  To determine the code level on each system in the sysplex and whether the RLS CF Cache Value keyword is honored, use the 'D SMS,SMSVSAM', 'D SMS,SMSVSAM,ALL', D SMS,CFCACHE() operator commands. When DFSMS cache structures connect to the system, VSAM RLS issues an IGW500I message to indicate that greater than 4K caching is active. The cache structures connect to the system through the first instance of a sphere opened on each system.

- You can specify the following values for the RLS_MaxCfFeatureLevel keyword:
  - A—This value allows greater than 4K caching if all active VSAM RLS instances in the sysplex have the correct level of code.
  - Z—This is the default value if you do not specify RLS_MaxCfFeatureLevel in the active SMS configuration. Greater than 4K caching is not allowed.
- RLS_MaxCfFeatureLevel keyword in the SETSMS command
- RLS_MaxCfFeatureLevel keyword in the SET SMS*xx* command
- RLS_MaxCfCacheFeatureLevel in the D SMS,OPTIONS command

CICS use of VSAM RLS: The Customer Information Control System (CICS) file-control component is a transactional file system built on top of VSAM. Prior to VSAM RLS, CICS file control performs its own record-level locking. The VSAM data sets are accessed through a single CICS. Local data sets are accessed by the CICS application-owning region (AOR) submitting requests directly to VSAM. Remote (shared) data sets are accessed by the CICS AOR submitting (function shipping) a file-control request to a CICS file-owning region (FOR) and the FOR submitting a request to VSAM. CICS file control provides transactional function such as commit, rollback, and forward recovery logging functions for recoverable data sets. Figure 2 on page 20 shows the AOR, FOR, and VSAM request flow prior to VSAM RLS.

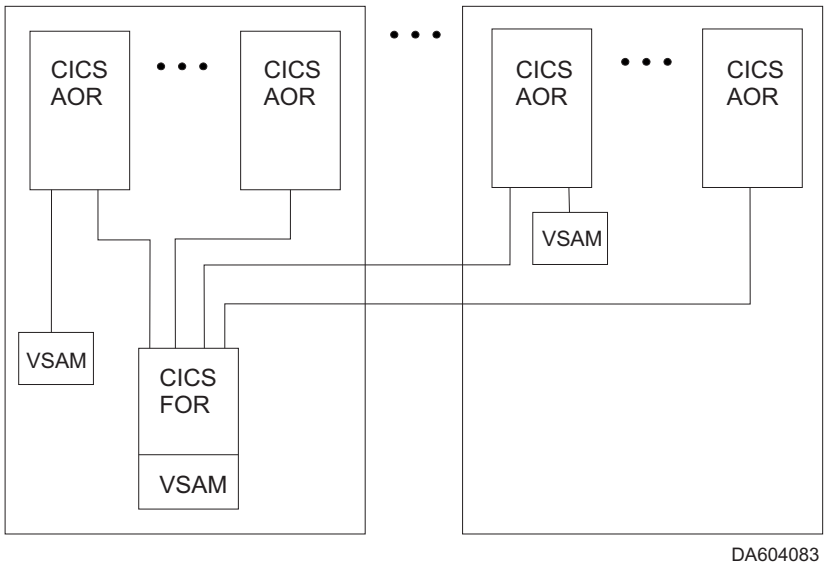


Figure 2. CICS VSAM non-RLS access

The CICS AOR's function ships VSAM requests to access a specific data set to the CICS FOR that owns the file that is associated with that data set. This distributed access form of data sharing has existed in CICS for some time.

With VSAM RLS, multiple CICS AORs can directly share access to a VSAM data set without CICS function shipping. With VSAM RLS, CICS continues to provide the transactional functions. The transactional functions are not provided by VSAM RLS itself. VSAM RLS provides CF-based record-level locking and CF data caching. Figure 3 shows a CICS configuration with VSAM RLS.

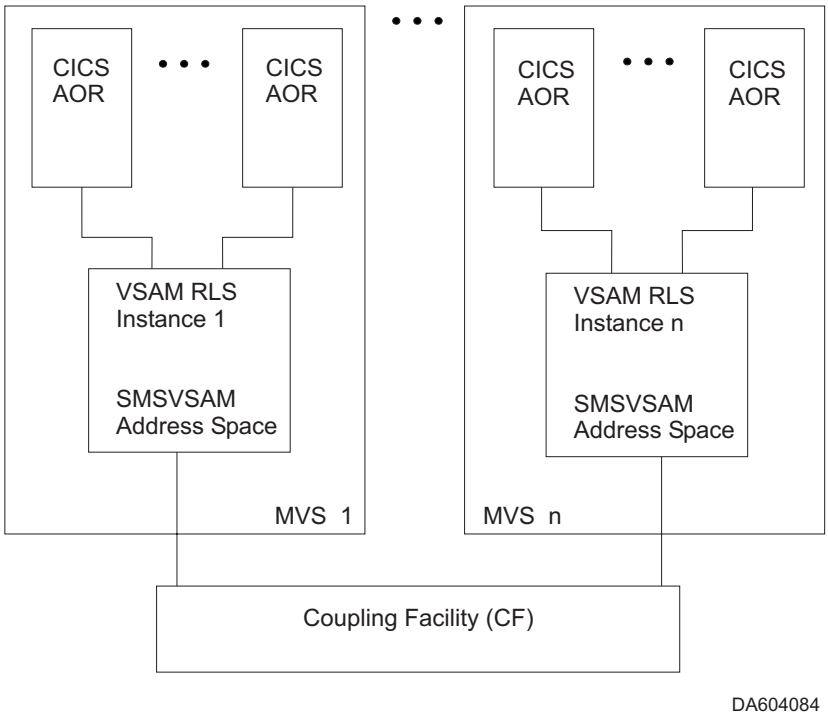


Figure 3. CICS VSAM RLS

VSAM RLS is a multisystem server. The CICS AORs access the shared data sets by submitting requests directly to the VSAM RLS server. The server uses the CF to serialize access at the record level.

Recoverable and nonrecoverable data sets: CICS file control supports a recoverable/nonrecoverable data set concept. The data set definition includes a recoverability attribute LOG. The attribute options are specified as follows:

- LOG(NONE)—nonrecoverable

  Specifies the data set as nonrecoverable. CICS does not perform any logging of changes for a data set that has this attribute. Neither rollback nor forward recovery is provided.
- LOG(UNDO)—recoverable

  Specifies the data set as commit/rollback recoverable. CICS logs the before (UNDO) images of changes to the data set and backs out the changes if the application requests rollback or if the transaction terminates abnormally.
- LOG(ALL)—recoverable

  Specifies the data set as both commit/rollback recoverable and forward recoverable. In addition to the logging and recovery functions provided for LOG(UNDO), CICS logs the after (REDO) image of changes to the data set. The redo log records are used by forward recovery programs/products such as CICSVR (CICS VSAM recovery) to reconstruct the data set in the event of hardware or software damage to the data set.

You can specify VSAM recoverable data set attributes in IDCAMS (access method services) DEFINE and ALTER commands. In the data class, you can specify LOG along with BWO and LOGSTREAMID. If you want to be able to back up a data set while it is open, you should define them using the IDCAMS BWO(TYPECICS) parameter. Only a CICS application or DFSMStvs can open a recoverable data set for output because VSAM RLS does not provide the logging and other transactional functions required for writing to a recoverable data set.

When a data set is opened in a non-RLS access mode (NSR, LSR, or GSR), the recoverable attributes of the data set do not apply and are ignored. The recoverable data set rules have no impact on existing programs that do not use RLS access.

CICS transactional recovery for VSAM recoverable data sets: The transactional services of CICS provide an ideal environment for data sharing. Exclusive locks held by VSAM RLS on the modified records cause read-with-integrity and write requests to these records by other transactions to wait. After the modifying transaction commits or rolls back, the locks are released and other transactions can access the records.

The CICS rollback (backout) function removes changes made to the recoverable data sets by a transaction. When a transaction terminates abnormally, CICS implicitly performs a rollback.

The commit and rollback functions protect an individual transaction from changes that other transactions make to a recoverable data set or other recoverable resource. This lets the transaction logic focus on the function it is providing and not have to be concerned with data recovery or cleanup in the event of problems or failures.

Non-CICS use of VSAM RLS: When VSAM RLS is used outside of CICS or DFSMStvs, the applications do not have the transactional recovery environment. In

most cases, this makes read/write data sharing not feasible. The following text describes the level of support that VSAM RLS provides for non-CICS applications, outside of DFSMStvs.

A non-CICS application outside of DFSMStvs is permitted to open a recoverable data set in RLS mode only for input. VSAM RLS provides the necessary record-level locking to provide read-with-integrity (if requested) for the non-CICS application. This support lets multiple CICS applications have the sphere open for read/write RLS access. CICS provides the necessary transactional recovery for the writes to the recoverable data set. Concurrently, non-CICS applications outside DFSMStvs can have the sphere open for read RLS access.

Read sharing of recoverable data sets: A non-CICS application outside DFSMStvs is permitted to open a recoverable data set in RLS mode only for input. VSAM RLS provides the necessary record-level locking to provide read-with-integrity (if requested) for the non-CICS application. This support lets multiple CICS applications have the sphere open for read/write RLS access. CICS provides the necessary transactional recovery for the writes to the recoverable data set. Concurrently, non-CICS applications outside DFSMStvs can have the sphere open for read RLS access. VSAM provides the necessary locking. Because the non-CICS application is not permitted to write to the sphere, transactional recovery is not required.

Read-sharing integrity across KSDS CI and CA splits: VSAM with non-RLS access does not ensure read integrity across splits for non-RLS access to a data set with cross-region share options 2, 3, and 4. If read integrity is required, the application must ensure it. When KSDS CI and CA splits move records from one CI to another CI, there is no way the writer can invalidate the data and index buffers for the reader. This can result in the reader not seeing some records that were moved.

VSAM RLS can ensure read integrity across splits. It uses the cross-invalidate function of the CF to invalidate copies of data and index CI in buffer pools other than the writer's buffer pool. This ensures that all RLS readers, DFSMStvs, CICS, and non-CICS outside DFSMStvs, are able to see any records moved by a concurrent CI or CA split. On each GET request, VSAM RLS tests validity of the buffers and when invalid, the buffers are refreshed from the CF or DASD.

Read and write sharing of nonrecoverable data sets: Nonrecoverable data sets do not participate in transactional recovery. Commit and rollback logging do not apply to these spheres. Because transactional recovery is not required, VSAM RLS permits read and write sharing of nonrecoverable data sets concurrently by DFSMStvs, CICS, and non-CICS applications. Any application can open the sphere for output in RLS mode.

VSAM RLS provides record locking and buffer coherency across the CICS and non-CICS read/write sharers of nonrecoverable data sets. However, the record lock on a new or changed record is released as soon as the buffer that contains the change has been written to the CF cache and DASD. This differs from the case in which a DFSMStvs or CICS transaction modifies VSAM RLS recoverable data sets and the corresponding locks on the added and changed records remain held until the end of the transaction.

For sequential and skip-sequential processing, VSAM RLS does not write a modified control interval (CI) until the processing moves to another CI or an

ENDREQ is issued by the application. In the event of an application abnormal termination or abnormal termination of the VSAM RLS server, these buffered changes are lost.

While VSAM RLS permits read and write sharing of nonrecoverable data sets across DFSMStvs and CICS and non-CICS applications, most applications are not designed to tolerate this sharing. The absence of transactional recovery requires very careful design of the data and the application.

Non-RLS access to VSAM data sets: RLS access does not change the format of the data in the VSAM data sets. The data sets are compatible for non-RLS access. If the data set has been defined with a cross-region share option of 2, a non-RLS open for input is permitted while the data set is open for RLS processing; but a non-RLS open for output fails. If the data set is already open for non-RLS output, an open for RLS fails. Therefore, at any time, a data set (sphere) can be open for non-RLS write access or open for RLS access.

CICS and VSAM RLS provide a quiesce function to assist in the process of switching a sphere from CICS RLS usage to non-RLS usage.

Differences between RLS access and non-RLS access: This topic describes the differences between RLS access and non-RLS access.

Share options: For non-RLS access, VSAM uses the share options settings to determine the type of sharing permitted. If you set the cross-region share option to 2, a non-RLS open for input is permitted while the data set is already open for RLS access. VSAM provides full read and write integrity for the RLS users, but does not provide read integrity for the non-RLS user. A non-RLS open for output is not permitted when already opened for RLS.

VSAM RLS provides full read and write sharing for multiple users; it does not use share options settings to determine levels of sharing. When an RLS open is requested and the data set is already open for non-RLS input, VSAM does check the cross-region setting. If it is 2, then the RLS open is permitted. The open fails for any other share option or if the data set has been opened for non-RLS output.

Locking: Non-RLS provides local locking (within the scope of a single buffer pool) of the VSAM control interval. Locking contention can result in an "exclusive control conflict" error response to a VSAM record management request.

VSAM RLS uses a DFSMS lock manager to provide a system-managed duplexing rebuild process. The locking granularity is at the VSAM record level. When contention occurs on a VSAM record, the request that encountered the contention waits for the contention to be removed. The DFSMS lock manager provides deadlock detection. When a lock request is in deadlock, VSAM rejects the request. This results in the VSAM record management request completing with a deadlock error response.

When you request a user-managed rebuild for a lock structure, the validity check function determines if there is enough space for the rebuild process to complete. If there is not enough space, the system rejects the request and displays an informational message.

When you request an alter operation for a lock structure, the validity check function determines if there is enough space for the alter process to complete. If there is not enough space, the system displays a warning message that includes the size recommendation.

VSAM RLS supports a timeout value that you can specify through the RPL, in the PARMLIB, or in the JCL. CICS uses this parameter to ensure that a transaction does not wait indefinitely for a lock to become available. VSAM RLS uses a timeout function of the DFSMS lock manager.

Retaining locks: VSAM RLS uses share and exclusive record locks to control access to the shared data. An exclusive lock is used to ensure that a single user is updating a specific record. The exclusive lock causes any read-with-integrity request for the record by another user (CICS transaction or non-CICS application) to wait until the update is finished and the lock released.

Failure conditions can delay completion of an update to a recoverable data set. This occurs when a CICS transaction enters in-doubt status. This means CICS can neither rollback nor commit the transaction. Therefore, the recoverable records modified by the transaction must remain locked. Failure of a CICS AOR also causes the current transaction's updates to recoverable data sets not to complete. They cannot complete until the AOR is restarted.

When a transaction enters in-doubt, sysplex failure, MVS failure, failure of an instance of the SMSVSAM Address Space, or a CICS AOR terminates, any exclusive locks on records of recoverable data sets held by the transaction must remain held. However, other users waiting for these locks should not continue to wait. The outage is likely to be longer than the user would want to wait. When these conditions occur, VSAM RLS converts these exclusive record locks into retained locks.

Both exclusive and retained locks are not available to other users. When another user encounters lock contention with an exclusive lock, the user's lock request waits. When another user encounters lock contention with a retained lock, the lock request is immediately rejected with “retained lock” error response. This results in the VSAM record management request that produced the lock request failing with “retained lock” error response.

If you close a data set in the middle of a transaction or unit of recovery and it is the last close for this data set on this system, then RLS converts the locks from active to retained.

Supporting non-RLS access while retained locks exist: Retained locks are created when a failure occurs. The locks need to remain until completion of the corresponding recovery. The retained locks only have meaning for RLS access. Lock requests issued by RLS access requests can encounter the retained locks. Non-RLS access does not perform record locking and therefore would not encounter the retained locks.

To ensure integrity of a recoverable sphere, VSAM does not permit non-RLS update access to the sphere while retained locks exist for that sphere. There can be situations where an installation must execute some non-CICS applications that require non-RLS update access to the sphere. VSAM RLS provides an IDCAMS command (SHCDS PERMITNONRLSUPDATE) that can be used to set the status of a sphere to enable non-RLS update access to a recoverable sphere while retained locks exist. This command does not release the retained locks. If this function is

used, VSAM remembers its usage and informs the CICSs that hold the retained locks when they later open the sphere with RLS.

If you use the SHCDS PERMITNONRLSUPDATE command, neither CICS nor DFSMStvs has any idea whether or not it is safe to proceed with pending backouts. Because of this, you must supply exits that DFSMStvs and CICS call, and each exit must tell the resource manager whether or not to go ahead with the backout. For more information, see the description of the batch override exit in "IGW8PNRU routine for batch override" on page 224.

VSAM options not supported by RLS: RLS does not support the following options and capabilities:

- Linear data sets
- Addressed access to a KSDS
- Control interval (CNV or ICI) to any VSAM data set type
- User buffering (UBF)
- Clusters that have been defined with the IMBED option
- Key Range data sets
- Temporary data sets
- GETIX and PUTIX requests
- MVS Checkpoint/Restart facility
- ACBSDS (system data set) specification
- Hiperbatch
- Catalogs, VVDS, the JRNAD exit, and any JCL AMP= parameters in JCL
- Data that is stored in z/OS UNIX System Services

In addition, VSAM RLS has the following restrictions:

- You cannot specify RLS access when accessing a VSAM data set using the ISAM compatibility interface.
- You cannot open individual components of a VSAM cluster for RLS access.
- You cannot specify a direct open of an alternate index for RLS access, but you can specify RLS open of an alternate index path.
- RLS open does not implicitly position to the beginning of the data set. For sequential or skip-sequential processing, specify a POINT or GET DIR, NSP request to establish a position in the data set.
- RLS does not support a request that is issued while the caller is executing in any of the following modes: cross-memory mode, SRB mode, or under an FRR. See "VSAM RLS request execution mode requirements" for a complete list of mode requirements.
- RLS does not support UNIX files.

VSAM RLS request execution mode requirements: When a program issues a VSAM RLS request (OPEN, CLOSE, or Record Management request), the program must be executing in the following execution mode with the listed constraints:

- Task mode (not SRB mode)
- Address Space Control=Primary
- Home address space=Primary address space=Secondary address space
- No functional recovery routine (FRR) can be in effect, but an ESTAE might be.

The VSAM RLS record management request task must be the same task that opened the ACB, or the task that opened the ACB must be in the task hierarchy. That is, the record management task was attached by the task that opened the ACB, or by a task that was attached by the task that opened the ACB.

VSAM RLS read integrity options: VSAM RLS provides three levels of read integrity as follows:

1. NRI—no read integrity

   This tells VSAM RLS not to obtain a record lock on the record accessed by a GET or POINT request. This avoids the overhead of record locking. This is sometimes referred to as dirty read because the reader might see an uncommitted change made by another transaction.

   Even with this option specified, VSAM RLS still performs buffer validity checking and buffer refresh when the buffer is invalid. Thus, a sequential reader of a KSDS does not miss records that are moved to new control intervals by control interval (CI) and control area (CA) splits.

   There are situations where VSAM RLS temporarily obtains a shared lock on the record even though NRI is specified. This happens when the read encounters an inconsistency within the VSAM sphere while attempting to access the record. An example of this is path access through an alternate index to a record for which a concurrent alternate index upgrade is being performed. The path access sees an inconsistency between the alternate index and base cluster. This would normally result in an error response return code 8 and reason code 144. Before giving this response to the NRI request, VSAM RLS obtains a shared lock on the base cluster record that was pointed to by the alternate index. This ensures that if the record was being modified, the change and corresponding alternate index upgrade completes. The record lock is released. VSAM retries the access. The retry should find the record correctly. This internal record locking may encounter locking errors such as deadlock or timeout. Your applications must be prepared to accept locking error return codes that may be returned on GET or POINT NRI requests. Normally such errors will not occur.
2. CR—consistent read

   This tells VSAM RLS to obtain a SHARE lock on the record accessed by a GET or POINT request. It ensures the reader does not see an uncommitted change made by another transaction. Instead, the GET/POINT waits for the change to be committed or backed out and the EXCLUSIVE lock on the record to be released.
3. CRE—consistent read explicit

   This is the same as CR, except VSAM RLS keeps the SHARE lock on the record until end-of-transaction. This option is only available to CICS or DFSMStvs transactions. VSAM does not understand end-of-transaction for non-CICS or non-DFSMStvs usage.

   This capability is often referred to as REPEATABLE READ.

   The record locks obtained by the VSAM RLS GET requests with CRE option inhibit update or erase of the records by other concurrently executing transactions. However, the CRE requests do not inhibit the insert of other records by other transactions. The following cases need to be considered when using this function.

   a. If a GET DIR (Direct) or SKP (Skip Sequential) request with CRE option receives a “record not found” response, VSAM RLS does not retain a lock on the nonexistent record. The record could be inserted by another transaction.

b. A sequence of GET SEQ (sequential) requests with CRE option results in a lock being held on each record that was returned. However, no additional locks are held that would inhibit the insert of new records in between the records locked by the GET CRE sequential processing. If the application were to re-execute the previously executed sequence of GET SEQ,CRE requests, it would see any newly inserted records. Within the transactional recovery community, these records are referred to as "phantom" records. The VSAM RLS CRE function does not inhibit phantom records.

Specifying read integrity

You can use one of these subparameters of the RLS parameter to specify a read integrity option for a VSAM data set.

NRI
Specifies no read integrity (NRI). The application can read all records.

CR Specifies consistent read (CR). This subparameter requests that VSAM obtain a SHARE lock on each record that the application reads.

CRE
Specifies consistent read explicit (CRE). This subparameter requests serialization of the record access with update or erase of the record by another unit of recovery.

CRE gives DFSMStvs access to VSAM data sets open for input or output. CR or NRI gives DFSMStvs access to VSAM recoverable data sets only for output. For information about how to use these read integrity options for DFSMStvs access, see *z/OS DFSMStvs Planning and Operating Guide*.

For complete descriptions of these subparameters, see the description of the RLS parameter in *z/OS MVS JCL Reference*.

Specifying a timeout value for lock requests

You can use the RLSTMOUT parameter of the JCL EXEC statement to specify a timeout value for lock requests. A VSAM RLS or DFSMStvs request waits the specified number of seconds for a required lock before the request times out and is assumed to be in deadlock.

For information about the RLSTMOUT parameter, see the description of the EXEC statement in *z/OS MVS JCL Reference*.

For information about avoiding deadlocks and additional information about specifying a timeout value, see *z/OS DFSMStvs Planning and Operating Guide* and *z/OS MVS Initialization and Tuning Guide*.

Defining data sets for DFSMStvs access

This topic describes DFSMS access method services that you can use for DFSMStvs. For more information on access method services, see *z/OS DFSMS Access Method Services Commands*.

Allocating data sets

Access method services identifies the verb name ALLOCATE and attaches the terminal monitor program (TMP), which runs Time Sharing Option Extended (TSO/E) commands in the background. You should use the ALLOCATE command only to allocate new data sets to the job step. If you use ALLOCATE through

access method services for anything else (such as the handling of SYSOUT data sets), you can get unpredictable results. For more information on using this command, see *z/OS TSO/E Programming Guide*. Table 6 on page 30 separates the parameters to be used under access method services from the parameters that cause unpredictable results.

When you use ALLOCATE, the data set is allocated to the job step. If your job contains multiple allocations, you might need to use the DYNAMNBR parameter in the job control language (JCL) EXEC statement. DYNAMNBR establishes a control limit that TMP uses when allocating a data set. The control limit is the sum of the number of data definition (DD) statements that are coded plus the value coded in DYNAMNBR. If you do not use DYNAMNBR, the system sets it to 0 (the default). If you code DYNAMNBR incorrectly, the system uses the default and issues a JCL warning message. For a description of how to code the DYNAMNBR parameter, see *z/OS MVS JCL User's Guide*. For an example that illustrates the use of DYNAMNBR, see "Allocate a data set using SMS class specifications: Example 1" on page 48.

When you use the ALLOCATE command within access method services, you must follow the data set naming conventions of TSO/E when you run TMP in batch mode:

- If the data set name is not in quotation marks and a USER parameter is given in the JCL, the value in the USER parameter is prefixed to all data set names given by ALLOCATE.
- If the USER parameter is not in the JCL, no prefix is added to any data set name given by ALLOCATE.

For information about the naming conventions of TSO/E and other considerations when you use access method services commands from a TSO/E background job, see *z/OS TSO/E User's Guide*. For information about the USER parameter and its Resource Access Control Facility (RACF) requirements, see *z/OS MVS JCL Reference*.

You can use the ALLOCATE command to define data set attributes in several ways:

- You can use the storage management subsystem (SMS) parameters STORCLAS, MGMTCLAS, and DATACLAS. You can either define these parameters explicitly or let them use the parameters assigned by the ACS routines that your storage administrator defines. For information about storage administration policies and about how the ACS routines might apply, contact your storage administrator.

  You cannot override attributes that the STORCLAS and MGMTCLAS parameters assign. You can override attributes that the DATACLAS parameter assigns. For example, if you use both the DATACLAS parameter and the SPACE parameter, SMS assigns all the attributes defined in the DATACLAS parameter but uses the values you defined in the SPACE parameter when allocating a data set.
- You can use the LIKE parameter to allocate a data set with the same attributes as an existing (model) data set. The model data set must be a cataloged data set. You can override any of the model data set attributes by stating them in the ALLOCATE command.
- You can identify a data set and explicitly describe its attributes.

Restrictions

- If the access method services job step contains either the SYSTSIN or SYSTSPRT DD statement, the ALLOCATE command is unsuccessful. Access method services allocates the SYSTSIN and SYSTSPRT DD statements to pass the command to TMP and to retrieve any error messages that are issued. This is

done for every ALLOCATE command. Any TMP error messages appear in the SYSPRINT data set, and access method services prints a summary message to show the final status of the command.

- The access method services ALLOCATE command is not supported if access method services is called in the foreground of TSO/E or if TSO/E Release 2 or later is not installed.
- You cannot use ALLOCATE if you have used the ATTACH macro to call IDCAMS from an application program. If you do, ALLOCATE fails with an ATTACH return code.

Allocation of SMS-managed data sets

If SMS is active, it can handle data set storage and management requirements for you. The storage administrator defines SMS classes with ACS routines, which assign classes to a new data set. When a storage administrator assigns a storage class to a new data set, the data set becomes an SMS-managed data set. Data class and management class are optional for SMS-managed data sets. For information on writing ACS routines, see *z/OS DFSMSdfp Storage Administration*.

Your storage administrator writes ACS routines that assign SMS classes to a data set. The SMS classes follow:

- *Storage class* Contains performance and availability attributes you can use to select a volume for a data set. You do not need to use the volume and unit parameters for a data set that is SMS-managed.
- *Data class* Contains the attributes related to the allocation of the data set, such as LRECL, RECFM, and SPACE. The data set attributes, if not specified in the ALLOCATE command, are derived from the model specified on LIKE, or from the data class. If the system cannot allocate the requested amount of space on the eligible volumes in the selected storage group, SMS retries allocation with a reduced space quantity. However, SMS will not do any retries, including reduced space quantity, unless Space Constraint Relief = Y is specified. If the data class assigned to the data set allows space constraint relief, other limits can be bypassed.

  For a list of the attributes for a data class, see the description of the DATACLAS parameter.
- *Management class* Contains the attributes related to the migration and backup of the data set by DFSMShsm™.

Allocation of non-SMS-managed data sets

You can define the DATACLAS parameter to allocate non-SMS-managed data sets. Do not specify the STORCLAS and MGMTCLAS parameters.

Return codes for the ALLOCATE command

Code	Description
0	Allocation successful.
12	Allocation unsuccessful. An error message has been issued. Refer to SYSPRINT for the error message.

Syntax for ALLOCATE parameters

In the following table, the access method services ALLOCATE parameters appear in the column “Acceptable parameters”. Parameters that might cause unpredictable results if used within access method services appear in the column “Parameters to use with caution”.

Table 6. ALLOCATE command parameters. This table lists the access method services ALLOCATE parameters.

Acceptable parameters	Parameters to use with caution
{**DATASET**(*dsname*)[**FILE**(*ddname*)]}	{* | *dsname-list*} | **DUMMY**
[**ACCODE**(*access code*)][1]	
[**ALTFILE**(*name*)]	
[**AVGREC**(**U** | **K** | **M**)]	
[**BFALN**(**F** | **D**)][2]	
[**BFTEK**(**S** | **E** | **A** | **R**)][2]	
[**BLKSIZE**(*value*)][2]	
[**BUFL**(*buffer-length*)][2]	
[**BUFNO**(*number-of-buffers*)]	
[**BUFOFF**({*block-prefix-length* | **L**)]}[2]	
	[**BURST** | **NOBURST**]
[**BWO**(**TYPECICS** | **TYPEIMS** | **NO**)]	
	[**CHARS**[*tablename-list*]]
	[**COPIES**((*number*),[*group-value-list*])]
[**DATACLAS**(*data-class-name*)]	
[**DEN**({**0** | **1** | **2** | **3** | **4**})][1]	
	[**DEST**(*destination* | *destination.userid*)]
[**DIAGNS**(*trace*)][2]	
[**DIR**(*integer*)]	
[**DSNTYPE(LIBRARY** | **PDS)**]	
[**DSORG**(**DA** | **DAU** | **PO** | **POU** | **PS** | **PSU**)][2]	
[**EROPT**(**ACC** | **SKP** | **ABE**)]	
[**EXPDT**(*year-day*) | **RETPD**(*number-of-days*)]	
	[**FCB**(*image-id*,**ALIGN,VERIFY**)]
	[**FLASH**(*overlay-name*,[*copies*])]
	[**FORMS**(*forms-name*)]
	[**HOLD** | **NOHOLD**]
	[**INPUT** | **OUTPUT**]
[**KEEP** | **CATALOG**]	[**DELETE** | **UNCATALOG**]
[**KEYLEN**(*bytes*)]	
[**KEYOFF**(*offset*)]	
[**LABEL**(*type*)][1]	
[**LIKE**(*model-dsname*)]	| [**USING**(*attr-list-name*)]
[**LIMCT**(*search-number*)]	
[**LRECL**({*logical-record-length* |	
(*nnnnnK* | *X*)})]	
[**MGMTCLAS**(*management-class-name*)]	
[**MAXVOL**(*count*)]	
	[**MODIFY**(*module-name*,[*trc*])]
[**NEW**]	[**OLD** | **SHR** | **MOD**]

Table 6. ALLOCATE command parameters (continued). This table lists the access method services ALLOCATE parameters.

Acceptable parameters	Parameters to use with caution
[**NCP**(*number-of-channel-programs*)][2]	
[**OPTCD**(**A,B,C,E,F,H,J,Q,R,T,W,Z**)][2]	
	[**OUTDES**(*output-descriptor-name*[,*output-descriptor-name*...])]
[**POSITION**(*sequence-number*)][1]	
[**PRIVATE**]	
[**PROTECT**]	
[**RECFM**(**A,B,D,F,M,S,T,U,V**)][2]	
[**RECORG**(**ES** | **KS** | **LS** | **RR**)]	
[**REFDD**(*dsname*)]	
[**RELEASE**][2]	
[**REUSE**]	
[**ROUND**][2]	
[**SECMODEL**(*profile-name*[,**GENERIC**])]	
[**SPACE**(*quantity*[,*increment*])	
{**BLOCK**(*value*) | **AVBLOCK**(*value*) |	
CYLINDERS | **TRACKS**}]	
[**STORCLAS**(*storage-class-name*)]	
	[**SYSOUT**(*class*)]
[**TRTCH**(**C** | **E** | **ET** | **T**)][1]	
[**UCOUNT**(*count*) | **PARALLEL**]	
	[**UCS**(*universal-character-set-name*)]
[**UNIT**(*type*)]	
[**VOLUME**(*serial-list*)]	
[**VSEQ**(*vol-seq-number*)]	
	[**WRITER**(*external-writer-name*)]

[1] Parameters applicable to tape data sets only.

[2] Parameters applicable to non-VSAM data sets only.

Abbreviation for the ALLOCATE command: ALLOC

Descriptions of the parameters within access method services follow. For information about ALLOCATE parameters not described in this topic, see *z/OS TSO/E Command Reference*.

Required parameters:

DATASET(*dsname***)**
Gives the name of the data set to be allocated. The data set name must be fully qualified.

- You must follow the data set naming conventions of TSO/E when you run TMP in batch mode.

- All temporary SMS-managed data sets must either have a name (DSNAME value) that starts with with & or && or have no name.

  Non-VSAM temporary data sets are the only uncataloged data sets that you can create.

  For more information about temporary data sets, see *z/OS MVS JCL Reference*. For more information about VSAM temporary data sets, see *z/OS DFSMS Using Data Sets*.
- You cannot concurrently allocate data sets that reside on the same physical tape volume.
- For allocation of a generation data group member, provide the fully qualified data set name, including the generation number.

Abbreviation: DA, DSN, DSNAME

FILE(*ddname***)**
Specifies the name of a data definition (DD) statement, which can be up to eight characters long. If you omit this parameter, the system assigns an available system file name. Do not use special DD names unless you want to use the facilities that those names represent to the system. For more information about AMSDUMP, see *z/OS DFSMS Access Method Services Commands*. For more information about the following special DD names, see *z/OS MVS JCL Reference*:

```
AMSDUMP         SYSABEND
JOBCAT          SYSCHK
JOBLIB          SYSCKEOV
STEPCAT         SYSMDUMP
STEPLIB         SYSUDUMP
```

For more information about these special DD names, see *z/OS TSO/E Command Reference*:

```
SYSTSIN         SYSTSPRT
```

You cannot use SYSTSIN and SYSTSPRT in a job step that runs the ALLOCATE command. See “Restrictions” on page 28 for further information.

Optional parameters:

ACCODE(*access code***)**
Gives or changes the accessibility code for an ISO/ANSI output tape data set, which protects it from unauthorized use. You can use up to eight characters in the access code, but ISO/ANSI validates only the first character. The ACCODE value can now be any of the following 57 ISO/ANSI a-type characters: blank, uppercase letters A-Z, numeric 0-9, or one of the special characters !*"%&'()+,-./:;<=>?_ Note that Version 3 still supports only uppercase characters A-Z. Password protection is supported for ANSI tape data sets under the PASSWORD/NOPWREAD options of the LABEL parameter. Password access overrides any ACCODE value if you use both options.

ALTFILE(*name***)**
Specifies the name of the SYSIN subsystem data set that is to be allocated. The name can be up to eight characters long. The system uses this parameter primarily in the background.

This gives the length in bytes of the average block.

AVGREC(*U*|*K*|*M***)**
Determines the size of the average record block. You can use these values:

U Use the primary and secondary quantities as given in the SPACE parameter.

K Multiply primary space quantity and secondary space quantity by 1024 (1 KB).

M Multiply primary space quantity and secondary space quantity by 1,048,576 (1 MB).

Use the AVGREC parameter to define a new data set when:

- The units of allocation that are requested for storage space are records.
- The primary and secondary space quantities used with the SPACE parameter represent units, thousands, or millions of records.

When you use AVGREC with the SPACE parameter, the first subparameter for the SPACE parameter must give the average record length of the records.

Use the AVGREC parameter when you want to show records as the units of allocation. You can also use this parameter to override the space allocation defined in the data class for the data set.

If SMS is not active, the system checks syntax and then ignores the AVGREC parameter.

BFALN(F|D)

Gives the boundary alignment of each buffer:

F Each buffer starts on a fullword boundary that might not be a doubleword boundary.

D Each buffer starts on a doubleword boundary.

If you do not use this parameter, the system defaults to a doubleword boundary.

BFTEK(S|E|A|R)

Is the type of buffering that you want the system to use:

S Simple buffering

E Exchange buffering

A Automatic record area buffering

R Record buffering

BFTEK(R) is not compatible with partitioned data sets extended (PDSE) and results in an error if used with the DSNTYPE(LIBRARY) parameter.

BLKSIZE(*value*)

Specifies the block size of the data control block (DCB) for the data set. The maximum allowable decimal value for the block size that is recorded in the DCB is 32,760. You can specify BLKSIZE for NEW or MOD data sets.

For direct access storage device (DASD) data sets: If you do not use BLKSIZE, the system determines an optimal DCB block size for a new data set. You can create the DCB block size in any of these ways:

- The system determines the block size if SMS is active and you do not assign the block size.
- You can assign the block size through the BLKSIZE parameter.
- You can use the LIKE parameter to obtain the block size from an existing model data set.

- If you do not assign BLKSIZE or LIKE, the system can determine the block size from the BLOCK parameter.

The block size that you assign for the DCB must be consistent with the requirements of the RECFM parameter. If you use:

- RECFM(F), the block size must be equal to, or greater than, the logical record length.
- RECFM(FB), the block size must be an integral multiple of the logical record length.
- RECFM(V), the block size must be equal to, or greater than, the largest block in the data set. (For unblocked variable-length records, the size of the largest block must allow space for the 4-byte block descriptor word, in addition to the largest logical record length. The logical record length must allow space for a 4-byte record descriptor word.)
- RECFM(VB), the block size must be equal to, or greater than, the largest block in the data set. For block variable-length records, the size of the largest block must allow space for the 4-byte block descriptor word, in addition to the sum of the logical record lengths that will go into the block. Each logical record length must allow space for a 4-byte record descriptor word.

  Because the number of logical records can vary, estimate the optimum block size and the average number of records for each block, based on your knowledge of the application that requires the I/O.
- RECFM(U) and BLKSIZE(80), one character is truncated from the line. That character (the last byte) is reserved for an attribute character.

For PDSEs:

- The system chooses the BLKSIZE value if you do not explicitly specify it. If BLKSIZE is given, the system treats the BLKSIZE value as the length of the simulated block. For create mode processing, the logical record length is equal to the block size if LRECL is not given. If you use LRECL, BLKSIZE must conform to the LRECL and RECFM definitions. If you use:

  RECFM(F)
  BLKSIZE must equal LRECL

  RECFM(FB) or RECFM(FBS)
  BLKSIZE must be a multiple of LRECL

  RECFM(V) or RECFM(VB)
  BLKSIZE must be at least four bytes larger than LRECL

  RECFM(VBS)
  BLKSIZE must be at least eight bytes.
- For input or update processing, the block size must conform to the currently defined record length. The BLKSIZE given when the data set was created is the default. However, you can use any BLKSIZE if it conforms to the record length definition.

BUFL(*buffer-length***)**
Specifies the length, in bytes, of each buffer in the buffer pool. Substitute a decimal number for *buffer-length*. The number must not exceed 32,760. If you omit this parameter and the system acquires buffers automatically, the BLKSIZE and KEYLEN parameters supply the information needed to establish buffer length.

BUFNO(*number-of-buffers***)**
Specifies the number of buffers that are assigned for data control blocks. Substitute a decimal number for number-of-buffers. The number must never

exceed 255. You can be limited to a smaller number of buffers depending on the limit established when the operating system was generated. The following list shows how to get a buffer pool and the action required:

Method
Action

BUILD macro instruction
You must use BUFNO

GETPOOL macro instruction
The system uses the number that you assign for GETPOOL

Automatically with BPAM, BSAM
You must use BUFNO

Automatically with QSAM
You can omit BUFNO and accept two buffers

BUFOFF({*block-prefix-length***|L})**
Defines the buffer offset. The *block-prefix-length* must not exceed 99. L specifies the block prefix field is 4 bytes long and contains the block length.

BWO(TYPECICS|TYPEIMS|NO)
Use this parameter if backup-while-open (BWO) is allowed for the VSAM sphere. BWO applies only to SMS data sets and cannot be used with TYPE(LINEAR). If BWO, LOG, or LOGSTREAMID is specified (or an RLS cell exists for the data set), access from DFSMS/MVS 1.2 or a lower-level system is denied.

If BWO is specified in the SMS data class, the specified BWO value is used as part of the data set definition, unless BWO was previously defined with an explicitly specified or modeled DEFINE attribute.

TYPECICS
Use TYPECICS to specify BWO in a CICS or DFSMStvs environment. For RLS processing, this activates BWO processing for CICS or DFSMStvs, or both. For non-RLS processing, CICS determines whether to use this specification or the specification in the CICS FCT. See the CICS Transaction Server for z/OS Information Center at http://publib.boulder.ibm.com/infocenter/cicsts/v3r1/index.jsp.

Exception: If CICS determines that it will use the specification in the CICS FCT, the specification might override the TYPECICS parameter for CICS processing.

Abbreviation: TYPEC

TYPEIMS
Enables BWO processing for IMS data sets. You can use this capability only with DFSMS 1.3 or higher-level DFSMS systems. If you attempt to open a cluster that has the TYPEIMS specification of a DFSMS 1.2 (or lower-level) system, the open will not be successful.

Abbreviation: TYPEI

NO Use this when BWO does not apply to the cluster.

Exception: If CICS determines that it will use the specification in the CICS FCT, the specification might override the NO parameter for CICS processing.

DATACLAS(*data-class-name***)**
This is the 1- to 8-character name of the data class for either SMS or

non-SMS-managed data sets. If you do not assign DATACLAS for a new data set and the storage administrator has provided an automatic class selection (ACS) routine, the ACS routine can select a data class for the data set. If you assign DATACLAS for an existing data set, SMS ignores it. If SMS is not active, the system checks the syntax and then ignores the DATACLAS parameter.

If you use the data class, you do not need to list all the attributes for a data set. For example, the storage administrator can provide RECFM, LRECL, RECORG, KEYLEN, and KEYOFF as part of the data class definition. However, you can override the DATACLAS parameter by explicitly defining the appropriate parameters in the ALLOCATE command.

The data class defines these data set allocation attributes:

- Data set organization:
  - Record organization (RECORG)
  - Record format (RECFM)
- Record length (LRECL)
- Key length (KEYLEN)
- Key offset (KEYOFF)
- Space allocation
  - AVGREC
  - SPACE
- Expiration date (EXPDT) or retention period (RETPD)
- Volume count (VOLUME)
- For VSAM data sets, the following:
  - Index options (IMBED, or REPLICATE, or both)
  - Control interval size (CISIZE)

    RLS and DFSMStvs support all CISIZE values, but a CISIZE value other than *n**2K consumes space in the coupling facility cache structure because a 2K data element does not have multiple coupling facilities. If CISIZE is not used and the storage class is a nonblank cache set, the system factors the default CISIZE.
  - Percent free space (FREESPACE)
  - Sharing options (SHAREOPTIONS)
    - A cluster defined with IMBED cannot be opened for RLS or DFSMStvs access.
    - SHAREOPTIONS is assumed to be (3,3) when you use RLS or DFSMStvs.

Table 7. Data class attributes for each data set organization. This table lists data class attributes for each data set organization.

Attributes	KS	ES	RR	LDS
CISIZE	X	X	X	X
FREESPACE	X			
IMBED	X			
KEYLEN	X			
KEYOFF	X			
LRECL	X	X	X	
REPLICATE	X			

Table 7. Data class attributes for each data set organization (continued). This table lists data class attributes for each data set organization.

Attributes	KS	ES	RR	LDS
SHAREOPTIONS	X	X	X	X
SPACE	X	X	X	X
Volume Count	X	X	X	X

DEN({0|1|2|3|4})
Gives the magnetic tape density as follows:

0 200 bpi/7 track

1 556 bpi/7 track

2 800 bpi/7 and 9 track

3 1600 bpi/9 track

4 6250 bpi/9 track (IBM 3420 Models 4, 6, and 8)

DIAGNS(*trace***)**
Specifies the OPEN/CLOSE/End-of-Volume trace option that gives a module-by-module trace of the OPEN/CLOSE/End-of-Volume work area and your DCB.

DIR(*integer***)**
Gives the number of 256 byte records for the directory of a new partitioned data set. Use this parameter to allocate a new partitioned data set.

DSNTYPE(LIBRARY|PDS)
Determines allocation of either a partitioned data set (PDS) or a partitioned data set extended (PDSE). A PDSE must be SMS-managed. If SMS is not active, the system checks the syntax and then ignores the DSNTYPE parameter.

LIBRARY
A PDSE in record format

For more information on PDSE, see *z/OS DFSMS Using Data Sets*.

DSORG(DA|DAU|PO|POU|PS|PSU)
the data set organization as:

DA Direct access

DAU Direct access unmovable

PO Partitioned organization

POU Partitioned organization unmovable

PS Physical sequential

PSU Physical sequential unmovable

When you allocate a new data set and do not use the DSORG parameter, these occur:

- If you assign a non-zero to the DIR parameter, DSORG defaults to the partitioned organization (PO) option.
- If you do not assign a value to the DIR parameter, DSORG defaults to the physical sequential (PS) option.
- The system does not store default DSORG information in the data set until a program opens and writes to the data set.

With PDSEs, the PSU and POU options are incompatible and result in an error if used with DSNTYPE(LIBRARY) while the data set is open for output. If the data set is open for input or update, PSU and POU are ignored.

To indicate the data set organization for VSAM data sets, see RECORG.

EROPT(ACC|SKP|ABE)
The option you want to run if an error occurs when the system reads or writes a record. The possible options are as follows:

ACC Accept the block of records in which the error was found

SKP Skip the block of records in which the error was found

ABE End the task abnormally

EXPDT(*year-day*)|RETPD(*number-of-days*)
Expiration date or the retention period. The MGMTCLAS maximum retention period, if given, limits the retention period in this parameter. The system ignores these parameters for temporary data sets.

EXPDT(*year-day*)
This shows the data set expiration date. Include the year and day as either:

- *yyddd*, where *yy* is the last two digits of the year and *ddd* is the three-digit number for the day of the year. The maximum for the year is 99 (for 1999), and for the day is 366.

  If you enter 99365 or 99366, the system retains your data sets permanently. Do not use those dates as expiration dates. Use them as no-scratch dates only.
- *yyyy/ddd*, where *yyyy* is the four-digit number for the year and *ddd* is the three-digit number for the day of the year. This syntax requires a slash. The maximum for the year is 2155. The maximum for the day is 366.

  If you use 1999/365 or 1999/366, the system retains your data sets permanently. Do not use those dates as an expiration date. Use them as no-scratch dates only.

EXPDT and RETPD are mutually exclusive.

RETPD(*number-of-days*)
Data set retention period, in days. It can be a one-digit to four-digit decimal number.

RETPD and EXPDT are mutually exclusive.

KEEP|CATALOG
A command processor can modify the final disposition with these parameters.

KEEP
This retains the data set by the system after step termination.

CATALOG
This retains the data set in a catalog after step termination.

KEYLEN(*bytes*)
This is the length, in bytes, of each of the keys used to locate blocks of records in the data set when the data set resides on a direct access device.

If an existing data set has standard labels, you can omit this parameter and let the system retrieve the key length from the standard label. If no source supplies a key length before you enter an OPEN macro instruction, the system uses zero (no keys). This parameter is mutually exclusive with TRTCH.

When you want to define the key length or to override the key length defined in the data class that the DATACLAS parameter specifies for the data set, use KEYLEN. The number of bytes follows:

- 1 to 255 for a record organization of key-sequenced (RECORG(KS))
- 0 to 255 for a data set organization of physical sequential (PS) or partitioned (PO)

For PDSEs, you can use 0 or 8. Use 8 only when opening the PDSE for input. Any other value results in an error.

KEYOFF(*offset*)
This shows the key position (offset) of the first byte of the key in each record. Use it to define key offset or override the key offset defined in the data class of the data set. It is only for a key-sequenced data set (RECORG=KS).

Use KEYOFF parameter to allocate both SMS-managed and non-SMS-managed data sets. If SMS is not active, however, the system checks syntax and then ignores the KEYOFF parameter.

LABEL(*type*)
This selects the label processing, one of: SL, SUL, AL, AUL, NSL, NL, LTM, or BLP, which correspond to the JCL label-types.

For VSAM data sets, the system always uses SL, whether you define SL or SUL or neither. NSL, NL, and BLP do not apply to VSAM data sets.

LIKE(*model-dsname*)
This names a model data set. The system uses these attributes as the attributes of the new data set that is being allocated. The model data set must be cataloged and must reside on a direct access device. The volume must be mounted when you enter the ALLOCATE command.

Note: TSO/E naming conventions apply when you assign *model-dsname*.

When the ALLOCATE command assigns attributes to a new data set, these attributes are copied from the model data set if SMS is active:

AVGREC
Size of average record block (kilobyte, megabyte)

BLOCK, AVBLOCK,

TRACKS, CYLINDERS
Space unit

DIR Directory space quantity

DSORG
Non-VSAM data set organization

KEYLEN
Key length

KEYOFF
Key offset

LRECL
Logical record length

RECFM
Record format

RECORG
VSAM data set organization

SPACE
Primary and secondary space quantities.

The system copies these attributes only if SMS is not active:

BLKSIZE
Block size

EXPDT
Data set expiration date

OPTCD
Optional services code (for ISAM data sets only)

VSEQ Volume sequence number.

You can still use the LIKE parameter even if you do not have an existing data set with the exact attributes you want to assign to a new data set. You can use ALLOCATE attributes to override any model data set attributes you do not want assigned to the new data set.

When you use the LIKE parameter, these rules apply:

- LIKE must be used with the NEW parameter; it cannot be used with OLD, SHR, or MOD.
- Use LIKE with the DATASET parameter; it cannot be used with FILE.
- Only one *dsname* can be given in the DATASET parameter.
- The system does not copy the block size from the model data set when SMS is active. If you do not show a block size in the ALLOCATE command, the system determines an optimal block size to assign to the data set.
- When SMS is active, attributes copied from the model data set override attributes from the data class.
- If you allocate the new data set with a member name (indicating a partitioned data set), the system prompts you for directory blocks unless that quantity is either shown in the ALLOCATE command or defaulted from the LIKE data set.
- If the new data set name is indicated with a member name, but the model data set is sequential and you have not given the quantity for directory blocks, you are prompted for directory blocks.

If you define the directory value as zero and the model data set is a PDS, the system allocates the new data set as a sequential data set.

The LIKE, REFDD, and USING operands are mutually exclusive. For more information on the USING operand, see *z/OS TSO/E Command Reference*.

LIMCT(*search-number***)**
This is the number of blocks or tracks that the system is to search for a block or available space. The number must not exceed 32760.

LRECL({*logical-record-length***|(***nnnnnK***|***X***)})**
This is the length, in bytes, of the largest logical record in the data set. You must define this parameter for data sets that consist of either fixed-length or variable-length records.

Use the DATACLAS parameter in place of LRECL to assign the logical record length. If SMS is active and you use LRECL, the system determines the block size.

If the data set contains undefined-length records, omit LRECL.

The logical record length must be consistent with the requirements of the RECFM parameter and must not exceed the block size (BLKSIZE parameter), except for variable-length spanned records. If you use:

- RECFM(V) or RECFM(V B), then the logical record length is the sum of the length of the actual data fields plus four bytes for a record descriptor word.
- RECFM(F) or RECFM(F B), then the logical record length is the length of the actual data fields.
- RECFM(U), omit the LRECL parameter.

LRECL(nnnnnK) allows users of ANSI extended logical records and users of QSAM "locate mode" to assign a K multiplier to the LRECL parameter. *nnnnn* can be a number within 1-16384. The K indicates that the value is multiplied by 1024.

For variable-length spanned records (VS or VBS) processed by QSAM (locate mode) or BSAM, use LRECL (X) when the logical record exceeds 32,756 bytes.

For PDSEs, the meaning of LRECL depends upon the data set record format:

- **Fixed-format records**. For PDSEs opened for output, the logical record length (LRECL) defines the record size for the newly created members. You cannot override the data set control block (DSCB) (LRECL); an attempt to do so will result in an error.
- **Variable-format records**. The LRECL is the maximum record length for logical records that are contained in members of the PDSE.
- **Undefined-format records**. The LRECL is the maximum record length for records that are contained in members of the PDSEs.

MGMTCLAS(*management-class-name***)**

For SMS-managed data sets: This is the 1-to-8 character name of the management class for a new data set. When possible, do not use MGMTCLAS. Allow it to default through the ACS routines.

After the system allocates the data set, attributes in the management class define the following items:

- The migration of the data set. This includes migration both from primary storage to migration storage, and from one migration level to another in a hierarchical migration scheme.
- The backup of the data set. This includes frequency of backup, number of versions, and retention criteria for backup versions.

If SMS is not active, the system checks the syntax and then ignores the MGMTCLAS parameter.

MAXVOL(*count***)**

This is the maximum number (1-255) of volumes upon which a data set can reside. This number corresponds to the count field on the VOLUME parameter in JCL. Use this to override the volume count attribute defined in the data class of the data set.

If VOLUME and PRIVATE parameters are not given, and MAXVOL exceeds UCOUNT, the system removes no volumes when all the mounted volumes have been used, causing abnormal termination of your job. If PRIVATE is given, the system removes one of the volumes and mounts another volume in its place to continue processing.

MAXVOL overrides any volume count in the data class (DATACLAS) of the data set.

Your user attribute data set (UADS) must contain the MOUNT attribute. Use of this parameter implies PRIVATE.

NEW
This creates a data set. For new partitioned data sets, you must use the DIR parameter. If you assign a data set name, the system keeps and catalogs a NEW data set. If you do not assign a data set name, the system deletes the data set at step termination.

NCP(*number-of-channel-programs***)**
This gives the maximum number of READ or WRITE macro instructions that are allowed before a CHECK macro instruction is entered. The number must not exceed 99 and must be less than 99 if a lower limit was established when the operating system was generated. If you are using chained scheduling, you must assign an NCP value greater than 1. If you omit the NCP parameter, the default value is 1.

OPTCD(A,B,C,E,F,H,J,Q,R,T,W,Z)
This lists optional services: (See also the OPTCD subparameter of the DCB parameter in *z/OS MVS JCL Reference* for details.)

A Requires that the actual device addresses be presented in READ and WRITE macro instructions.

B Requires that the end-of-file (EOF) recognition be disregarded for tapes.

C Uses chained scheduling.

E Asks for an extended search for block or available space.

F Returns device address feedback from a READ or WRITE macro instruction in the form it is presented to the control program.

H Requests the system to check for and bypass. For more information, see *z/OS MVS JCL Reference*.

J Makes the character after the carriage control character the table reference character for that line. The table reference character tells TSO/E which character arrangement table to select when printing the line.

Q Translates a magnetic tape from ASCII to EBCDIC or from EBCDIC to ASCII.

R Requires relative block addressing.

T Requests the user totaling facility.

W Tells the system to perform a validity check when it wrties data on a direct access device.

Z Asks the control program to shorten its normal error recovery procedure for input on magnetic tape.

You can use any or all the services by combining the characters in any sequence, separating them with blanks or commas.

For PDSEs, the system ignores OPTCD values other than OPTCD(J). OPTCD(J) requires that the first data byte in the output data line is a 3800 table reference character.

POSITION(*sequence-number*)
This is the relative position (1-9999) of the data set on a multiple data set tape. The sequence number corresponds to the data set sequence number field of the label parameter in JCL.

PRIVATE
This assigns the private-volume use attribute to a volume that is neither reserved nor permanently in resident. It corresponds to the PRIVATE keyword of the VOLUME parameter in JCL.

If you do not use VOLUME and PRIVATE parameters and MAXVOL exceeds UCOUNT, the system removes no volumes when all the mounted volumes have been used, causing abnormal termination of your job. If you use PRIVATE, the system removes one of the volumes and mounts another volume to continue processing.

PROTECT
This RACF-protects the DASD data set or the first data set on a tape volume.

- For a new permanent DASD data set, the status must be NEW or MOD, treated as NEW, and the disposition must be either KEEP, CATALOG, or UNCATALOG. With SMS, SECMODEL overrides PROTECT.
- For a tape volume, the tape must have an SL, SUL, AL, AUL, or NSL label. The file sequence number and volume sequence number must be one (except for NSL). You must assign PRIVATE as the tape-volume use attribute.

The PROTECT parameter is not valid if a data set name is not given, or if the FCB parameter or status other than NEW or MOD is used.

RECFM(A,B,D,F,M,S,T,U,V)
This sets the format and characteristics of the records in the data set. They must be completely described by one source only. If they are not available from any source, the default is an undefined-length record. See also the RECFM subparameter of the DCB parameter in *z/OS MVS JCL Reference* for a detailed discussion.

Use these with the RECFM parameter:

A To show the record contains ASCII printer control characters

B To indicate the records are blocked

D For variable length ASCII records

F For fixed length records.

M For records with machine code control characters.

S For fixed-length records, the system writes the records as standard blocks (there must be no truncated blocks or unfilled tracks except for the last block or track). For variable-length records, a record can span more than one block. Exchange buffering, BFTEK(E), must not be used.

T The records can be written onto overflow tracks, if required. Exchange buffering, BFTEK(E), or chained scheduling, OPTCD(C), cannot be used.

U The records are of undefined length.

V Shows variable length records.

You must provide one or more values for this parameter.

For PDSEs, these statements apply:

- RECFM can be partially modified from the value that is saved in the DSCB when creating members.
- In a PDSE that is created as fixed or fixed blocked, members must always be created with fixed-length logical records. However, the attribute of blocked might change between member creates. The first record format assigned to the PDSE is the default for member creates. The characteristic of blocked might not change during an open.
- Attempts to overwrite the record format characteristic of F, U, or V with another value from that set causes a system error.
- RECFM(A) and RECFM(M) are compatible with PDSEs.

RECFM and RECORG are mutually exclusive.

RECORG(*ES*|*KS*|*LS*|*RR***)**
Determines the organization of the records in a new VSAM data set. To override the record organization defined in the data class (DATACLAS) of the data set, use RECORG.

You can assign:

ES For a VSAM entry-sequenced data set

KS For a VSAM key-sequenced data set

LS For a VSAM linear space data set. You cannot access linear data sets with VSAM record-level sharing (RLS) or DFSMStvs.

RR For a VSAM relative record data set

If you do not use RECORG, SMS assumes a non-VSAM data set.

RECORG and RECFM are mutually exclusive. To define the data set organization for a non-VSAM data set, see DSORG.

Exception: You can use the RECORG parameter to allocate both SMS-managed and non-SMS-managed data sets. If SMS is not active, however, the system checks the syntax and ignores the RECORG parameter.

REFDD(*dsname***)**
Specifies the name of an existing data set whose attributes are copied to a new data set. The system copies these attributes to the new data set:

- Data set organization:
  - Record organization (RECORG)
  - Record format (RECFM)
- Record length (LRECL)
- Key length (KEYLEN)
- Key offset (KEYOFF)
- Space allocation
  - AVGREC
  - SPACE

The system does not copy the retention period (RETPD) or expiration date (EXPDT) to the new data set.

LIKE and REFDD are mutually exclusive.

Exception: You can use the REFDD parameter to allocate both SMS-managed and non-SMS-managed data sets. If SMS is not active, however, the system checks the syntax and then ignores the REFDD parameter.

RELEASE
To delete unused space when the data set is closed.

If you use RELEASE for a new data set with the BLOCK or BLKSIZE parameter, then you must also use the SPACE parameter.

REUSE
Frees and reallocates a data set if it is currently in use.

You cannot use the REUSE parameter to reallocate a data set from a disposition of OLD to a disposition of SHR. However, you can first free the data set with OLD and then reallocate it with SHR.

ROUND
Allocates space equal to one or more cylinders. Use this parameter only when you request space in units of blocks. This parameter corresponds to the ROUND parameter of the SPACE parameter in JCL.

SECMODEL(*profile-name***[,GENERIC])**
Names an existing RACF profile to copy to the discrete profile. Use SECMODEL when you want a different RACF data set profile from the default profile selected by RACF, or when there is no default profile. The model profile can be any of these profiles:

- RACF model profile
- RACF discrete data set profile
- RACF generic data set profile

Use GENERIC to state the profile name as a generic data set profile.

The system copies this information from the RACF data set profile to the discrete data set profile of the new data set:

- OWNER indicates the user or group assigned as the owner of the data set profile.
- ID is the access list of users or groups that are authorized to access the data set.
- UACC gives universal access authority that is associated with the data set.
- AUDIT|GLOBALAUDIT selects which access attempts are logged.
- ERASE indicates that the data set when it is deleted (scratched).
- LEVEL is the installation-defined level indicator.
- DATA is installation-defined information.
- WARNING indicates that an unauthorized access causes RACF to issue a warning message, but allows access to the data set.
- SECLEVEL is the name of an installation-defined security level.

Exception: You can use the SECMODEL parameter to allocate both SMS-managed and non-SMS managed data sets. If SMS is not active, however, the system checks the syntax and then ignores the SECMODEL parameter.

For more information about RACF, see *z/OS Security Server RACF Command Language Reference*.

SPACE(*quantity***[,***increment***])**
Allocates the amount of space for a new data set. If you omit this parameter and:

- You are running MVS/ESA Version 3. The system uses the IBM-supplied default value of SPACE(10,50) AVBLOCK (1000).
- You are running MVS/ESA SP Version 4. The system uses the IBM-supplied default value of SPACE(4,24) AVBLOCK (8192).

However, your installation might have changed the default. For more information about default space, see *z/OS MVS Programming: Authorized Assembler Services Guide*.

To have the system determine the amount of space, include the AVGREC parameter in place of BLOCK, AVBLOCK, CYLINDERS, and TRACKS. To supply your own space value, define one of the following: BLOCK(*value*), BLKSIZE(*value*), AVBLOCK(*value*), CYLINDERS, or TRACKS. The amount of space requested is determined as follows:

- BLOCK(*value*) or BLKSIZE(*value*): The BLOCK or BLKSIZE parameter's *value* is multiplied by the SPACE parameter's *quantity*.
- AVBLOCK(*value*): The AVBLOCK parameter's *value* is multiplied by the SPACE parameter's *quantity*.
- CYLINDERS: The SPACE parameter's *quantity* is given in cylinders.
- TRACKS: The SPACE parameter's *quantity* is given in tracks.

Use SPACE for NEW and MOD data sets.

quantity
: Allocates the initial number of units of space for a data set. For a partitioned data set, a directory quantity is not necessary.

increment
: This is the number of units of space to be added to the data set each time the previously allocated space has been filled. You must provide the primary quantity along with the increment value.

BLOCK(*value*)
: Shows the average length (in bytes) of the records written to the data set. The maximum block value used to determine space to be allocated is 65,535. The block value is the unit of space that is used by the SPACE parameter. A track or a cylinder on one device can represent a different amount of storage (number of bytes) from a track or a cylinder on another device. Determine the unit of space value from the:
  - Default value of (10 50) AVBLOCK(1000) if no space parameters (SPACE, AVBLOCK, BLOCK, CYLINDERS, or TRACKS) are given.
  - The BLOCK parameter.
  - The model data set, if the LIKE parameter is used and BLOCK, AVBLOCK, CYLINDERS, or TRACKS is not given.
  - The BLKSIZE parameter if BLOCK is not used.

AVBLOCK(*value*)
: This shows only the average length (in bytes) of the records that are written to the data set.

CYLINDERS
: Requests allocation in cylinders as the unit of space.

TRACKS
: Requests allocation in tracks as the unit of space.

  Exception: If you specify tracks for a VSAM data set, the space allocated will be contiguous. For more information, see *z/OS DFSMS Using Data Sets*.

STORCLAS(*storage-class-name***)**
For SMS-managed data sets: Gives the 1-to-8-character name of the storage class. When possible, allow STORCLAS to default through the ACS routines established by your storage administrator. Attributes assigned through storage class and the ACS routines replace storage attributes such as UNIT and VOLUME. If SMS is not active, the system checks the syntax and then ignores the STORCLAS parameter.

TRTCH(C|E|ET|T)
Selects the recording technique for 7-track tape as follows:

C Data conversion with odd parity and no translation.

E Even parity with no translation and no conversion.

ET Even parity and no conversion. BCD to EBCDIC translation when reading, and EBCDIC to BCD translation when writing.

T Odd parity and no conversion. BCD to EBCDIC translation when reading, and EBCDIC to BCD translation when writing.

The TRTCH and KEYLEN parameters are mutually exclusive.

UCOUNT(*count***)|PARALLEL**
Shows device allocation.

UCOUNT(*count***)**
This allocates the maximum number of devices, where count is a value from 1-59.

If you do not use VOLUME and PRIVATE parameters and MAXVOL exceeds UCOUNT, the system removes no volumes when the mounted volumes have been used, causing abnormal termination of your job. If you use PRIVATE, the system removes one of the volumes and mounts another volume in its place to continue processing.

PARALLEL
Mounts one device for each volume given in the VOLUME parameter or in the catalog.

UNIT(*type***)**
Defines the unit type to which a file or data set is to be allocated. You can list an installation-defined group name, a generic device type, or a specific device address. If you do not supply volume information (the system retrieves volume and unit information from a catalog), the unit type that is coded overrides the unit type from the catalog. This condition exists only if the coded type and class are the same as the cataloged type and class.

For VSAM data sets, use the AFF subparameter carefully. If the cluster components and the data and its index reside on unlike devices, the results of UNIT=AFF are unpredictable.

When you allocate a new SMS-managed data set, the system ignores the UNIT parameter. The system determines the UNIT and VOLUME from the storage class associated with the data set. Use UNIT only if you want to allocate a non-SMS-managed data set to a specific unit type.

If the storage administrator has set up a default unit under SMS regardless of whether the data set is SMS-managed, you do not have to use UNIT. If you do not, the system determines the default UNIT for both SMS-managed and non-SMS-managed data sets.

VOLUME(*serial-list***)**
This is the serial number of an eligible direct access volume on which a new data set is to reside or on which an old data set is located. If you use VOLUME for an old data set, the data set must be on the specified volume for allocation to take place. If you do not include VOLUME, the system allocates new data sets to any eligible direct access volume. The UNIT information in your procedure entry in the user attribute data set (UADS) determines eligibility. You can use up to 255 volume serial numbers.

For VSAM data sets, you need to use this subparameter carefully. For more information about DD parameters to avoid when processing VSAM data sets, see *z/OS MVS JCL User's Guide* before you use the VOLUME subparameter REF, *volume-sequence-number*, or *volume-count*.

When you allocate new SMS-managed data sets, you can let the ACS routines select the volume for you. The ACS routines assign your data set to a storage class that contains attributes such as VOLUME and UNIT. You can allocate your data set to a *specific* volume only if your storage administrator has stated GUARANTEED SPACE=YES in the storage class assigned to your data set. The volume serial numbers you provide might then override the volume serial numbers used by SMS. If space is not available on the given volume, however, your request is not successful.

Abbreviation: VOL

VSEQ(*vol-seq-number***)**
This locates which volume (1-255) of a multivolume begins data set processing. This parameter corresponds to the volume sequence number on the VOLUME parameter in JCL. Use VSEQ only when the data set is a cataloged data set.

ALLOCATE examples

The following scenarios use the ALLOCATE command to perform various functions:

Allocate a data set using SMS class specifications: Example 1: In this example, the ALLOCATE command is used to allocate a new data set. By providing the SMS data class, management class, and storage class, you can take advantage of the attributes assigned by your storage administrator through the ACS routines.

Although this example includes DYNAMNBR, it is not required in this example. Because this example contains two DD statements, you can do up to two allocations. DYNAMNBR is required only when the number of allocations exceeds the number of DD statements. This example sets DYNAMNBR to 1. This allows up to three allocations for each DD statement (2) plus DYNAMNBR (1).

```
//ALLOC    JOB  ...
           EC PGM=IDCAMS,DYNAMNBR=1
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *
     ALLOC -
           DSNAME(ALX.ALLOCATE.EXAMP1) -
           NEW CATALOG -
           DATACLAS(STANDARD) -
           STORCLAS(FAST) -
           MGMTCLAS(VSAM)
/*
```

Because the system syntax checks and ignores SMS classes when SMS is inactive, and because no overriding attributes are given, this example works only if SMS is active. The parameters are:

- DSNAME states that the name of the data set being allocated is ALX.ALLOCATE.EXAMP1.
- NEW creates a data set.
- CATALOG retains the data set by the system in the catalog after step termination. This is mandatory for SMS-managed data sets.
- DATACLAS gives an installation-defined name of a data class to be assigned to this new data set. The data set assumes the RECORG or RECFM, LRECL, KEYLEN, KEYOFF, AVGREC, SPACE, EXPDT or RETPD, VOLUME, CISIZE, FREESPACE, SHAREOPTIONS, and IMBED or REPLICATE parameters assigned to this data class by the ACS routines. This parameter is optional. If it is not used, the data set assumes the default data class assigned by the ACS routines.
- STORCLAS gives an installation-defined name of an SMS storage class to be assigned to this new data set. This storage class and the ACS routines are used to determine the volume. This parameter is optional and, if not given, the data set assumes the default storage class assigned by the ACS routines.
- MGMTCLAS is the installation-defined name of an SMS management class to be assigned to this new data set. The data set assumes the migration and backup criteria assigned to this management class by the ACS routines. This parameter is optional and, if not given, the data set assumes the default management class assigned by the ACS routines.

Allocate a VSAM data set using SMS class specifications: Example 2: This example uses the ALLOCATE command to allocate a new data set. Data class is not assigned, and attributes assigned through the default data class are overridden by explicitly specified parameters. By providing the SMS management class and storage class, you can take advantage of attributes already assigned through the ACS routines.

```
//ALLOC    JOB  ...
//STEP1    EXEC PGM=IDCAMS,DYNAMNBR=1
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *
     ALLOC -
           DSNAME(M166575.ALLOC.EXAMPLE) -
           NEW CATALOG -
           SPACE(10,2) -
           AVBLOCK(80) -
           AVGREC(K) -
           LRECL(80) -
           RECORG(ES) -
           STORCLAS(FAST) -
           MGMTCLAS(VSAM)
/*
```

The parameters are:

- DSNAME states that the name of the data set being allocated is M166575.ALLOC.EXAMPLE.
- NEW creates the data set.
- CATALOG retains the data set by the system in the catalog after step termination. This is mandatory for SMS-managed data sets.
- The SPACE parameter determines the amount of space to be allocated to the new data set.
  - The first amount (10) is the primary allocation. The second amount (2) is the secondary allocation.

  - Using AVGREC(K) determines that the amounts defined in the SPACE parameter represent kilobytes (K) of records. In this example, the primary allocation is 10K or 10240 records and the secondary allocation is 2K or 2048 records.
  - To determine the space allocation in bytes, multiply the number of records by 80, the record length in LRECL(80). The primary allocation is 819200 bytes. The secondary allocation is 163840 bytes.
- AVBLOCK is the average block length. This example uses an average block length of 80 bytes.
- AVGREC determines whether the quantity in the SPACE parameter represents units, thousands, or millions of records. "K" indicates that the primary and secondary space quantities are to be multiplied by 1024 (1 KB).
- LRECL says the logical record length in the data set is 80 bytes.
- RECORG shows entry-sequenced records in the new VSAM data set.
- STORCLAS gives an installation-defined name of an SMS storage class to be assigned to this new data set. This storage class and the ACS routines are used to determine the volume. This parameter is optional. If it is not used, the data set assumes the default storage class assigned by the ACS routines.
- MGMTCLAS shows an installation-defined name of an SMS management class to be assigned to this new data set. The data set assumes the migration and backup criteria assigned to this management class by the ACS routines. This parameter is optional and, if not given, the data set assumes the default management class assigned by the ACS routines.

Allocate a new data set: Example 3: This example shows the ALLOCATE command being used to allocate a new data set XMP.ALLOCATE.EXAMP3.

```
//ALLOC    JOB  ...
//STEP1    EXEC PGM=IDCAMS,DYNAMNBR=1
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *
     ALLOC -
           DSNAME(XMP.ALLOCATE.EXAMP3) -
           NEW CATALOG -
           SPACE(10,5) TRACKS -
           BLKSIZE(1000) -
           LRECL(100) -
           DSORG(PS) -
           UNIT(3380) -
           VOL(338002) -
           RECFM(F,B)
/*
```

The parameters are:

- DSNAME states that the name of the data set to be allocated is XMP.ALLOCATE.EXAMP3.
- NEW creates the data set.
- CATALOG retains the data set in the catalog after step termination.
- SPACE allocates the amount of space to the new data set. In this example, TRACKS is also used so the primary space is 10 tracks with an increment of 5 tracks.
- BLKSIZE requires that the data set control block (DCB) block size is 1000.
- LRECL sets the length of a logical record in the data set to 100.
- DSORG makes the data set physical sequential (PS).
- UNIT and VOL indicate that the data set is to reside on 3380 volume 338002.

- RECFM shows fixed block records in the data set.

Allocate a non-VSAM data set: Example 4: This example shows the ALLOCATE command being used to allocate a non-VSAM data set. ALLOCATE, unlike DEFINE NONVSAM, lets you give the SMS classes for a non-VSAM data set.

```
//ALLOC    JOB  ...
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=A
//SYSABEND DD   SYSOUT=A
 /SYSIN    DD   *
     ALLOC -
           DSNAME(NONVSAM.EXAMPLE) -
           NEW -
           DATACLAS(PS000000) -
           MGMTCLAS(S1P01M01) -
           STORCLAS(S1P01S01)
/*
```

The parameters are:

- DSNAME specifies that the name of the data set to be allocated is NONVSAM.EXAMPLE.
- NEW creates the data set does.is
- DATACLAS assigns an installation-defined name (PS000000) of a data class to this new data set. This parameter is optional and, if not used, the data set assumes the default data class assigned by the ACS routines.
- MGMTCLAS assigns an installation-defined name (S1P01M01) of a management class to this new data set. The data set assumes the migration and backup criteria assigned to this management class by the ACS routines. This parameter is optional and, if not used, the data set assumes the default management class assigned by the ACS routines.
- STORCLAS assigns an installation-defined name (S1P01S01) of a storage class to this new data set. This storage class and the ACS routines determine the volume. This parameter is optional and, if not used, the data set assumes the default storage class assigned by the ACS routines.

Allocate a partitioned data set extended: Example 5: This example shows the ALLOCATE command being used with the DSNTYPE keyword to allocate a PDSE.

```
//ALLOC    EXEC  PGM=IDCAMS,DYNAMNBR=1
//SYSPRINT DD    SYSOUT=A
//SYSIN    DD    *
           ALLOC -
              DSNAME(XMP.ALLOCATE.EXAMPLE1) -
              NEW -
              STORCLAS(SC06) -
              MGMTCLAS(MC06) -
              DSNTYPE(LIBRARY)
/*
```

The parameters follow:

- DSNAME specifies that the name of the data set to be allocated is XMP.ALLOCATE.EXAMPLE1.
- NEW creates the data set.
- STORCLAS uses the SC06 storage class definition for this data set.
- MGMTCLAS uses the SC06 management class definition for this data set.
- DSNTYPE(LIBRARY) indicates that the object being allocated is an SMS-managed PDSE.

Listing and controlling SMSVSAM recovery

Use the SHCDS command to list SMSVSAM recovery associated with subsystems and spheres and to control that recovery. This command works both in batch and in the TSO/E foreground. The functions include the following subcommands:

- List subcommands
- Subcommands that enable you to take action on work that was shunted
- Subcommands to control a manual forward recovery in the absence of a forward recovery utility that supports SMSVSAM protocols
- Subcommands that enable you to run critical non-RLS batch window work when it is not possible to first close out all outstanding SMSVSAM recovery
- A subcommand that allows for a subsystem cold start

  Recommendation: After a cold start, if recovery was not completed for any data sets, they are most likely left in a damaged state and must be recovered manually. If the data sets are forward recoverable, their forward recovery logs might also be damaged. Manually recover the data sets (without using forward recovery), take backups of them and of any other data sets that use the forward recovery log, and then delete and redefine the forward recovery log.

Use this command cautiously. The CICS Transaction Server for z/OS Information Center at http://publib.boulder.ibm.com/infocenter/cicsts/v3r1/index.jsp. describes many of the situations that require the use of the SHCDS command. For details about administering VSAM RLS, see *z/OS DFSMSdfp Storage Administration*.

The syntax of the access method services SHCDS command follows.

This table shows the syntax of the access method services SHCDS command.

Command	Parameters
SHCDS	{[**LISTDS(***base-cluster***)**[**JOBS**]] | [**LISTSHUNTED{SPHERE(***base-cluster***)** | **URID(**{*urid* | **ALL}**)}] | [**LISTSUBSYS(***subsystem* | **ALL)**] | [**LISTSUBSYSDS(***subsystem* | **ALL)**] | [**LISTRECOVERY(***base-cluster***)** | [**LISTALL**] | [**FRSETRR(***base-cluster***)**] | [**FRUNBIND(***base-cluster***)**] | [**FRBIND(***base-cluster***)**] | [**FRRESETRR(***base-cluster***)**] | [**FRDELETEUNBOUNDLOCKS(***base-cluster***)**] | [**PERMITNONRLSUPDATE(***base-cluster***)**] | [**DENYNONRLSUPDATE(***base-cluster***)**] | [**REMOVESUBSYS(***subsystem***)**] | [**CFREPAIR(**{**INFILE(***ddname***)** | **INDATASET(***dsname***)**} [({**LIST** | **NOLIST**})]] | [**CFRESET(**{**INFILE(***ddname***)** | **INDATASET(***dsname***)**}] [({**LIST** | **NOLIST**})]] | [**CFRESETDS(***base-cluster***)**]} [**PURGE{SPHERE(***base-cluster***)** | **URID(***urid***)**}] | [**RETRY{SPHERE(***base-cluster***)** | **URID(***urid***)**}] | [**OUTFILE(***ddname***)**]

Base-cluster is a fully or partially qualified VSAM data set name. The high-level qualifier must be specified. You can use an asterisk (*) for a subsequent qualifier, but then no lower-level qualifiers are allowed. For example, this is allowed:

```
A.*
```

This is not allowed:

```
A.*.B
```

Subsystem is the name of an online system, such as CICS, as registered to the SMSVSAM server.

Requirements:

1. Various levels of authority are required to use the SHCDS parameters. For more information, see *z/OS DFSMS Access Method Services Commands*.
2. A program that calls the SHCDS command must be APF-authorized. For more information, see *z/OS DFSMS Access Method Services Commands*.
3. To use the SHCDS command in the TSO foreground, you must add SHCDS to the authorized command list (AUTHCMD) in the IKJTSO*xx* member of SYS1.PARMLIB or to the CSECT IKJEGSCU. For more information, see *z/OS DFSMS Access Method Services Commands* and *z/OS TSO/E Customization*.
4. For examples and explanations of the output from the list parameters, see *z/OS DFSMS Access Method Services Commands*.

SHCDS parameters

The SHCDS parameters provide for these tasks:

- Listing information kept by the SMSVSAM server and the catalog as related to VSAM RLS or DFSMStvs.
  - LISTDS
  - LISTSUBSYS
  - LISTSUBSYSDS
  - LISTRECOVERY
  - LISTALL
  - LISTSHUNTED
- Controlling forward recovery, preserving retained locks when a data set is moved or copied, and, in rare cases when forward recovery fails, deleting the locks.
  - FRSETRR
  - FRUNBIND
  - FRBIND
  - FRRESETRR
  - FRDELETEUNBOUNDLOCKS
- Allowing non-RLS updates when forward recovery is required.
  - PERMITNONRLSUPDATE
  - DENYNONRLSUPDATE
- Removing the SMSVSAM server's knowledge of an inactive subsystem, thus forcing a cold start of the online application. Use REMOVESUBSYS only when procedures provided by the application have failed or you have no intention of ever using the subsystem again.
- REMOVESUBSYS

- Resetting VSAM RLS indicators in the catalog, allowing reconstruction of RLS information or fallback from VSAM RLS. (For the fallback procedure, see *z/OS DFSMSdfp Storage Administration*.)
  - CFREPAIR
  - CFRESET
  - CFRESETDS
- Taking action on work that DFSMStvs has shunted. Units of recovery are shunted when DFSMStvs is unable to finish processing them, for example, due to an I/O error. For each shunted log entry that exists, the locks associated with that entry are retained.
  - RETRY
  - PURGE

Required parameters: SHCDS has no required parameters, but you must specify one of the optional parameters. OUTFILE is a second optional parameter you can specify.

Optional parameters:

LISTDS(*base-cluster***) [JOBS]**
Lists the following information:
- The assigned coupling facility cache structure name
- The subsystem type and status:
  - Active for batch
  - Active or failed for online
- Whether the VSAM sphere is recoverable or nonrecoverable
- The state of the data set:
  - Forward recovery required
  - Retained locks
  - Lost locks
  - Locks unbound
  - Non-RLS update permitted
  - Permit-first-time switch

JOBS
When this keyword is specified, LISTDS returns a list of the jobs currently accessing the data set in DFSMStvs mode.

Abbreviation: LDS

LISTSHUNTED {SPHERE(*base-cluster***) | URID}(***urid***|ALL) }}**
Lists information about work that was shunted due to an inability to complete a syncpoint (commit or backout) for a given data set or unit of recovery, or for all shunted units of recovery when the ALL keyword is specified. The output includes the following information:
- The unit of recovery identifier
- The data set name
- The job with which the unit of recovery was associated
- The step within the job with which the unit of recovery was associated
- Whether the unit of recovery will be committed or backed out if it is retried

One of the following errors can cause shunting:

- C-FAILED: A commit failed.
- B-FAILED: A backout failed.
- IO-ERROR: An I/O error occurred on the data set.
- DS-FULL: The data set was full; no space on DASD to add records.
- IX-FULL: A larger alternate index is required.
- LOCK: A failure occurred during an attempt to obtain a lock during backout.
- LOG: A log stream became or was made unavailable.
- CACHE: A cache structure or connection to it failed.

This parameter requires that you have UPDATE authority to the data set specified.

Abbreviation: LSH

LISTSUBSYS(*subsystem***|ALL)**
lists information about a specific subsystem or all subsystems known to the SMSVSAM server:

- Subsystem status
  - Active for batch
  - Active or failed for online
- A summary showing whether the subsystem's shared data sets have:
  - Lost locks
  - Retained locks
  - Non-RLS update permitted

For an active subsystem, LISTSUBSYS gives the number of held locks, waiting lock requests, and retained locks. For a failed subsystem, LISTSUBSYS shows the number of retained locks.

Abbreviation: LSS

LISTSUBSYSDS(*subsystem***|ALL)**
Lists information about a specific subsystem or all subsystems known to the SMSVSAM server, including data sets that it is sharing. For each subsystem, this parameter lists the following information:

- Sharing protocol (online or batch)
- The status (active or failed)
- Recovery information for each shared data set:
  - Whether it has retained locks owned by this subsystem
  - Whether it has lost locks owned by this subsystem
  - Whether there are locks not bound to the data set
  - If forward recovery is required
  - If non-RLS update is permitted
  - The permit-first-time switch setting

Abbreviation: LSSDSL

LISTRECOVERY(*base-cluster***)**
lists data sets requiring recovery and the subsystems that share those data sets. Recovery indicators listed are:

- Lost locks
- Retained locks
- Non-RLS update permitted

- Forward recovery required

Abbreviation: LRCVY

LISTALL
Lists all information related to recovery for subsystems and VSAM spheres accessed in RLS mode. The output from this parameter can be quite large.

Abbreviation: LALL

FRSETRR(*base-cluster***)**
This parameter sets the forward-recovery-required indicator. Until reset with the FRRESETRR parameter, access is prevented until forward recovery is complete.

If you use a forward recovery utility that supports RLS, such as CICSVR, do not use this parameter.

Abbreviation: SETRR

FRUNBIND(*base-cluster***)**
This parameter unbinds the retained locks prior to restoring or moving the data set. These locks protect uncommitted changes and are needed for eventual backout. They must be rebound by using the FRBIND parameter.

If you use a forward recovery utility that supports RLS, such as CICSVR, do not use this parameter.

Abbreviation: UNB

FRBIND(*base-cluster***)**
Use this parameter after BLDINDEX to rebind the associated locks to the restored data set.

Attention: Between the unbind and the bind, do not delete any clusters in the sphere or change their names.

If you use a forward recovery utility that supports RLS, such as CICSVR, do not use this parameter.

Abbreviation: BIND

FRRESETRR(*base-cluster***)**
Use this parameter after forward recovery is complete and after locks have been bound to the new location of the data set using FRBIND. This allows access to the newly recovered data set by applications other than the forward recovery application.

If you use a forward recovery utility that supports RLS, such as CICSVR, do not use this parameter.

Abbreviation: RESET

FRDELETEUNBOUNDLOCKS(*base-cluster***)**
The FRDELETEUNBOUNDLOCKS parameter lets you delete locks in the rare case when a successful forward recovery is not possible. Every attempt should be made to complete forward recovery, whether using a product such as CICSVR that supports VSAM RLS or using another forward recovery procedure.

If forward recovery does not successfully complete, locks cannot be reassociated (bound) to the new version of the data set, because these locks do not provide the protection that online backout requires.

Before using this parameter, check the documentation for your online application. For CICS, the procedure is documented in the CICS Transaction Server for z/OS Information Center at http://publib.boulder.ibm.com/infocenter/cicsts/v3r1/index.jsp.

Abbreviation: DUNBL

PERMITNONRLSUPDATE(*base-cluster***)**
Allows a data set with pending RLS recovery to be opened for output in non-RLS mode. This command is used when it is necessary to run critical batch updates and RLS recovery cannot first be completed. This is reset the next time the data set is accessed for RLS. If after using PERMITNONRLSUPDATE, you do not run a non-RLS batch job, you must use DENYNONRLSUPDATE to prevent non-RLS updates.

Abbreviation: PERMT

DENYNONRLSUPDATE(*base-cluster***)**
If you inadvertently issue PERMITNONRLSUPDATE, use this parameter to reset the effect of PERMITNONRLSUPDATE.

If recovery was pending, but you did not run a non-RLS batch job, you must use this parameter. If not reset, CICS takes action assuming the data set has been opened for update in non-RLS mode.

Do not use DENYNONRLSUPDATE if you do indeed run non-RLS work after specifying PERMITNONRLSUPDATE. The permit status is reset the next time the data set is opened in RLS mode.

Abbreviation: DENY

REMOVESUBSYS(*subsystem***)**
Use this parameter to remove any knowledge of recovery owed to SMSVSAM by the named subsystem, including locks protecting uncommitted updates.

Normally, a failed online application would be restarted so that it can do the required backouts and release locks that protect uncommitted updates. However, sometimes it might be necessary to cold start the online application. For more information about cold starts, see the CICS Transaction Server for z/OS Information Center at http://publib.boulder.ibm.com/infocenter/cicsts/v3r1/index.jsp.

Use of this parameter is equivalent to cold starting the named subsystem with respect to the SMSVSAM server. Use REMOVESUBSYS for the rare cases where either there is no intention of ever running the subsystem again or the application's cold start procedures cannot be used. An example of an appropriate use of REMOVESUBSYS would be removing a test system that is no longer needed.

If the removed subsystem is ever run again, every effort should be made to cold start the subsystem.

Attention: Use of REMOVESUBSYS can result in loss of data integrity.

Abbreviation: RSS

CFREPAIR({INFILE(*ddname***) | INDATASET(***dsname***)})**
Use this command to reconstruct the RLS indicators for all applicable data sets in a restored catalog. CFREPAIR uses information known to the VSAM RLS server at the time the SHCDS command is used. The catalog must be import-connected on all systems to the master catalog before the CFREPAIR parameter can be used.

INFILE*(ddname)*
Indicates which DD statement defines the catalog to be processed.

INDATASET*(dsname)*
Use this to specify the name of the catalog to be processed.

({LIST|NOLIST})
Optional subparameters, which control the information returned by the CFREPAIR parameter.

LIST
Requests a list of data sets for which CFREPAIR successfully restored the RLS information. If you do not specify this subparameter, CFREPAIR lists only those data sets whose RLS information could not be restored.

NOLIST
Only data sets whose information could not be restored are listed. Using this subparameter is the same as not specifying LIST or NOLIST.

Abbreviation: CFREP

CFRESET({INFILE*(ddname)***|INDATASET(***dsname***)}**
Use this parameter if you've decided to fall back from using VSAM RLS. It clears VSAM RLS indicators in the catalog for all applicable data sets. *z/OS DFSMSdfp Storage Administration* includes a detailed fallback procedure. Also, for information specific to CICS, see the CICS Transaction Server for z/OS Information Center at http://publib.boulder.ibm.com/infocenter/cicsts/v3r1/index.jsp.

If the catalog is later restored, use CFREPAIR to reconstruct critical information required by the SMSVSAM server.

INFILE*(ddname)*
Specifies the data definition (DD) name of the catalog to be processed.

INDATASET*(dsname)*
Specifies the data set name of the catalog to be processed.

({LIST|NOLIST})
Optional subparameters, which control the information returned by the CFRESET parameter.

LIST
Requests a list of data sets for which CFRESET successfully processed the RLS indicators. If you do not specify this subparameter, CFRESET lists only those data sets whose indicators were not cleared.

NOLIST
Only data sets that were not successfully processed are listed. Using this subparameter is the same as not specifying LIST or NOLIST.

Abbreviation: CFRES

CFRESETDS(*base-cluster***)**
Use this parameter if you've decided to fall back from using VSAM RLS. It clears VSAM RLS indicators in the catalog for all applicable data sets. CFRESETDS This parameter differs from CFRESET in that it lets you select one or more data sets for fallback. CFRESETDS lists all data sets processed, not just those with errors.

A detailed fallback procedure is included in *z/OS DFSMSdfp Storage Administration*. Also, for information specific to CICS, see the CICS Transaction Server for z/OS Information Center at http://publib.boulder.ibm.com/infocenter/cicsts/v3r1/index.jsp.

Abbreviation: CFRDS

PURGE {SPHERE(*base-cluster*)|URID(*urid*)}
Discards the log entries and releases the associated locks. Use this command when the data set is damaged and cannot be restored to a state where it is consistent with the log entries. For example, it might have been necessary to restore the data set from a backup copy that predates the updates that were made to the data set prior to the failure.

Recommendation: If any data sets are in a lost locks status, do not issue this command while a DFSMStvs restart is in progress. If any lost locks recovery was not completed for a data set that is being processed by this command, the command does not complete until the DFSMStvs restart completes.

This parameter requires that you have update authority for the specified data set.

Abbreviation: none

RETRY {SPHERE(*base-cluster*)|URID(*urid*)}
Retries the syncpoint. Use this command when the data set can be restored to a state where it is consistent with the log entries. By *consistent*, we mean that the data set reflects the state that existed before the time of the particular unit of recovery for which DFSMStvs was unable to complete processing. This is possible for data sets that are forward recoverable or for failures that do not damage the data set (such as a dropped path). When the command completes successfully, locks associated with the log entries are released.

Recommendation: If any data sets are in a lost locks status, do not issue this command while a DFSMStvs restart is in progress. If any lost locks recovery was not completed for a data set that is being processed by this command, the command does not complete until the DFSMStvs restart completes.

This parameter requires that you have update authority for the specified data set.

Abbreviation: none

OUTFILE(*ddname*)
Specifies a data set, other than the SYSPRINT data set, to receive the output produced by the SHCDS command.

The value of *ddname* identifies the DD statement of the alternate target data set.

Abbreviation: OUTDD

SCHDS examples

The following examples show functions that the SCHDS command can perform.

Using PERMITNONRLSUPDATE with a generic data set name specification:
Example 1: The following example shows using the SHCDS subparameter PERMITNONRLSUPDATE with a generic data set name specification.

```
  /* SET NONRLS UPDATE ON                                  */
  SHCDS PERMITNONRLSUPDATE(SYSPLEX.PERMIT.*)
IDC2917I NO RACF PROFILE ON STGADMIN.IGWSHCDS.REPAIR
IDC01885I NON-RLS UPDATE PERMITTED FOR SYSPLEX.PERMIT.CLUS2
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0
```

Listing data sets with the high-level qualifier SYSPLEX: Example 2: The following example lists the data sets with the high-level qualifier of SYSPLEX.

In general, when a base cluster name can be specified for the SHCDS command, a generic can be used.

```
   SHCDS LISTDS(SYSPLEX.*)
 IDC2917I NO RACF PROFILE ON STGADMIN.IGWSHCDS.REPAIR
 ----- LISTING FROM SHCDS ----- IDCSH02
 ------------------------------------------------------------------------------
 DATA SET NAME----SYSPLEX.PERMIT.CLUS2
   CACHE STRUCTURE----CACHE01
   RETAINED LOCKS---------YES   NON-RLS UPDATE PERMITTED--------YES
   LOST LOCKS-------------NO   PERMIT FIRST TIME---------------YES
   LOCKS NOT BOUND---------NO   FORWARD RECOVERY REQUIRED--------NO
   RECOVERABLE------------YES

                           SHARING SUBSYSTEM STATUS
   SUBSYSTEM    SUBSYSTEM         RETAINED    LOST     NON-RLS UPDATE
   NAME         STATUS            LOCKS       LOCKS    PERMITTED
   ---------    --------------    --------    -----    --------------
   RETLK05A     ONLINE--FAILED    YES         NO       YES
 DATA SET NAME----SYSPLEX.RETAINED.CLUS1
   CACHE STRUCTURE----CACHE01
   RETAINED LOCKS---------YES   NON-RLS UPDATE PERMITTED---------NO
   LOST LOCKS-------------NO   PERMIT FIRST TIME----------------NO
   LOCKS NOT BOUND---------NO   FORWARD RECOVERY REQUIRED--------NO
   RECOVERABLE------------YES

                           SHARING SUBSYSTEM STATUS
   SUBSYSTEM    SUBSYSTEM         RETAINED    LOST     NON-RLS UPDATE
   NAME         STATUS            LOCKS       LOCKS    PERMITTED
   ---------    --------------    --------    -----    --------------
   RETLK05A     ONLINE--FAILED    YES         NO       NO
 DATA SET NAME----SYSPLEX.SHARED.CLUS4
   CACHE STRUCTURE----CACHE01
   RETAINED LOCKS---------YES   NON-RLS UPDATE PERMITTED---------NO
   LOST LOCKS-------------NO   PERMIT FIRST TIME----------------NO
   LOCKS NOT BOUND---------NO   FORWARD RECOVERY REQUIRED--------NO
   RECOVERABLE------------YES

                           SHARING SUBSYSTEM STATUS
   SUBSYSTEM    SUBSYSTEM         RETAINED    LOST     NON-RLS UPDATE
   NAME         STATUS            LOCKS       LOCKS    PERMITTED
   ---------    --------------    --------    -----    --------------
   RETLK05A     ONLINE--FAILED    YES         NO       NO
 IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0
```

Listing data sets with JOBS: Example 3: The following example shows an SHCDS LISTDS command for a data set with no retained locks. The data set is currently in use by 10 jobs accessing it in DFSMStvs mode.

```
 SHCDS LISTDS(SYSPLEX.KSDS.RETAINED.CLUS1) JOBS
 ----- LISTING FROM SHCDS ----- IDCSH02
 ------------------------------------------------------------------------
 DATA SET NAME----SYSPLEX.KSDS.RETAINED.CLUS1
   CACHE STRUCTURE----CACHE01
   RETAINED LOCKS----------NO   NON-RLS UPDATE PERMITTED---------NO
   LOST LOCKS-------------NO   PERMIT FIRST TIME----------------NO
   LOCKS NOT BOUND---------NO   FORWARD RECOVERY REQUIRED--------NO
   RECOVERABLE------------YES
                         SHARING SUBSYSTEM STATUS
   SUBSYSTEM    SUBSYSTEM        RETAINED    LOST    NON-RLS UPDATE
   NAME        STATUS           LOCKS      LOCKS    PERMITTED
   ---------    --------------    --------    -----    --------------
   RETLK05A    ONLINE--ACTIVE    YES        NO      NO
```

```
  JOB NAMES:

     TRANV001    TRANV002    TRANV003    TRANV004    TRANV005
     TRANJOB1    TRANJOB2    TRANJOB3    TRANJOB4    TRANJOB5
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0
```

Listing shunted entries: Example 4: The following example lists information for each shunted entry.

```
SHCDS LISTSHUNTED SPHERE(SYSPLEX.KSDS.CLUSTER.NAME)
 ------------------------------------------------------------------------
 CLUSTER NAME----SYSPLEX.KSDS.CLUSTER.NAME
 URID             DISPOSITION    JOB NAME    STEP NAME    CAUSE
 ----------------    -----------    --------    ---------    -----
 ABCDEFGH00000001    BACKOUT       TRANJOB1    TRANSTP3    B-FAILED
 XYZ@#$0000000000    BACKOUT       TRANJOB2    STPTRAN1    IO-ERROR
 0101BF$$22222222    COMMIT        TRANV001    TRANSTP1    C-FAILED
 IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0
```

Altering data set attributes

The ALTER command modifies the attributes of defined data sets and catalogs.

The syntax of the ALTER command follows.

This table shows the syntax of the ALTER command.

Command	Parameters
ALTER	*entryname* **[ACCOUNT(***account–info***)]** **[ADDVOLUMES(***volser***[*volser*...])]** **[BUFFERSPACE(***size***)]** **[BUFND(***number***)]** **[BUFNI(***number***)]** **[BWO(TYPECICS | TYPEIMS | NO)]** **[CCSID(***value***)]** **[CODE(***code***)]** **[ECSHARING | NOECSHARING]** **[EMPTY | NOEMPTY]** **[ERASE | NOERASE]** **[EXCEPTIONEXIT(***entrypoint***)]** **[FILE(***ddname***)]** **[FILEDATA(TEXT | BINARY)]** **[FREESPACE(***CI-percent***[*CA-percent*])]** **[FRLOG(NONE[REDO])]** **[INHIBIT | UNINHIBIT]** **[KEYS(***length offset***)]** **[LIMIT(***limit***)]** **[LOCK | UNLOCK]** **[LOG(NONE | UNDO | ALL)]** **[LOGSTREAMID(***logstream***)]** **[MANAGEMENTCLASS(***class***)]** **[NEWNAME(***newname***)]** **[NULLIFY(** **[AUTHORIZATION(MODULE | STRING)]** **[BWO]** **[CODE]** **[EXCEPTIONEXIT]** **[LOG]** **[LOGSTREAMID]**

This table shows the syntax of the ALTER command.

Command	Parameters
	[OWNER] **[RETENTION]** **[OWNER(***ownerid***)]** **[RECORDSIZE(***average maximum***)]** **[REMOVEVOLUMES(***volser***[***volser***...])]** **[REUSE | NOREUSE]** **[ROLLIN]** **[SCRATCH | NOSCRATCH]** **[SHAREOPTIONS(***crossregion***[***crosssystem***])]** **[STORAGECLASS(***class***)]** **[STRNO(***number***)]** **[TO(***date***) | FOR(***days***)]** **[TYPE(LINEAR)]** **[UNIQUEKEY | NONUNIQUEKEY]** **[UPDATE | NOUPDATE]** **[UPGRADE | NOUPGRADE]** **[WRITECHECK | NOWRITECHECK]** **[CATALOG(***catname***)]**

Entry types that can be altered

An "X" in Figure 4 on page 63 indicates that you can alter the value or attribute for the type of catalog entry that is shown. Some attributes apply only to either the data component or the index component of a cluster or alternate index entry. You can use some attributes only for the data or index component of a cluster or alternate index entry; you must then identify the entryname of the component. Use the LISTCAT command to determine the names generated for the object's components.

You can identify a group of entries with a generic name. Entrynames that match the supplied qualifiers are altered if they have the information that is used with the ALTER command.

You cannot alter alias entries or a master catalog's self-describing entries, nor can you change a fixed-length relative record data set to a variable-length relative record data set, or the reverse. You cannot change a linear data set (LDS) to any other VSAM data set format. Any attempt to alter a data set defined with a device type named by the user (for example, SYSDA) is unsuccessful.

When the data set characteristics being altered are for a compressed data set, the maximum record length of the control interval size is less than if compression is not done.

```
*******************************************************
                 J E S 2  J O B  L O G  --  S Y S T E M  3 0 8 1  --  N O D E  N 1
07.39.08 JOB   29  $HASP373 S2RAS031 STARTED - INIT  1 - CLASS A - SYS 3081
07.40.13 JOB   29       S2RAS031    STEP 0      IKJEFT01    0000
07.40.19 JOB   29  DFPWTX30 ISSUING COMMAND.
07.40.20 JOB   29 *41 S2RAS031--REPLY GO STEP1
07.40.46 JOB   29  R 41,U
07.40.52 JOB   29       S2RAS031    STEP1       WTORPGM     0000
07.42.31 JOB   29       S2RAS031    STEP2       AMBLIST     0000
07.42.58 JOB   29       S2RAS031    STEP2       AMBLIST     0000
                  IGD300I AN ABEND OCCURRED DURING SMS PROCESSING
                    ABEND SYSTEM CODE=06F ASID=0010
                    COMPONENT NAME=SMS  COMPONENT ID=28462
                    ACTIVE LOAD MODULE  NAME=IGDZILLA ADDRESS=01BE1000
                    CSECT IN ERROR      DESCRIPTION=BUILD MSG RTN 2
                                        NAME=IGDMCSC2 ADDRESS=01BF5758 OFFSET=00000026
                                        ASSEMBLY DATE=032487   PTF LEVEL=HDP3310
                    PSW AT TIME OF ERROR 071C0000 81BF577E
                    DATA AT PSW 01BF5778 - C5404040 400090EC D00C18CF
                    GPR  0-3  008C7444  7F70FC78  00000010  00000000
                    GPR  4-7  00000018  7F70FD10  7F70F434  7F70FD14
                    GPR  8-11 7F70FDD8  01BF58B2  01BF48B3  7F70FAB0
                    GPR 12-15 81BF38B4  7F70FAB0  81BF41B6  81BF5758
07.43.03 JOB   29  IGD306I UNEXPECTED ERROR DURING IGDMCSCN PROCESSING
                   RETURN CODE IS 8, REASON CODE IS 12008
                   THE MODULE THAT DETECTED THE ERROR IS IGDMCSCN
                   SMS MODULE TRACE BACK - MCSCM DSP00 SSIRT
                   SYMPTOM RECORD CREATED, PROBLEM ID IS IGD00025
07.43.04 JOB   29       S2RAS031    STEP3       IGDRAS00    0000
07.43.04 JOB   29  $HASP395 S2RAS031 ENDED
SAVE AREA TRACE

DGTFMD01 WAS ENTERED VIA LINK         AT EP DGTFMD01..90.349
SA   0002EFD8  WD1 000000D0   HSA 0002E818   LSA 05301C50   RET 80FD2C38   EPA 8538C630   R0  03178AEC
               R1  0002F084   R2  0002EB84   R3  FFFFFFFF   R4  0002EB84   R5  0002A4D0   R6  00000000
               R7  00000001   R8  0002EB80   R9  00029740   R10 00000000   R11 00000000   R12 83178790
DGTFMD01 WAS ENTERED VIA CALL         AT EP DGTFMD05..90.349
SA   05301C50  WD1 00000000   HSA 0002EFD8   LSA 053018E0   RET 8538C832   EPA 8538DA38   R0  03178AEC
               R1  05301D70   R2  0000000C   R3  00000048   R4  05301F54   R5  05301E74   R6  00000000
               R7  00000000   R8  8002EB9A   R9  00029 740  R10 0539213C   R11 05301C50   R12 8538C630
UNKNOWN  WAS ENTERED VIA CALL         AT EP ISPDIR.912 17
SA   00030AC0  WD1 000004C0   HSA 00030300   LSA 0003C010   RET 831708A6   EPA 8316ABC8   R0  00000000
               R1  00030B10   R2  00030E80   R3  000301B8   R4  000000FE   R5  00017C14   R6  00015218
               R7  00030668   R8  00030D60   R9  00029740   R10 00030DD8   R11 00000000   R12 83170018
UNKNOWN  WAS ENTERED VIA CALL         AT EP ISPDIL.92014.OY51175.3.3
SA   0003C010  WD1 000007C0   HSA 00030AC0   LSA 0003C7D0   RET 8316B89A   EPA 83178790   R0  05300FCC
               R1  0003C0E0   R2  00000000   R3  00000000   R4  00017000   R5  00017000   R6  00015218
               R7  0003C378   R8  0316DBC8   R9  00029740   R10 0316CBC8   R11 0316BBC8   R12 8316ABC8
DGTFVA00 WAS ENTERED VIA LINK
SA   0003C7D0  WD1 000000D0   HSA 0003C010   LSA 05301638   RET 80FD2C38   EPA 853AA7A0   R0  03178AEC
               R1  0003C87C   R2  0003C37C   R3  FFFFFFFF   R4  0003C37C   R5  0002A4D0   R6  00000000
               R7  00000001   R8  0003C378   R9  00029740   R10 00000000   R11 00000000   R12 83178790
DGTFVA00 WAS ENTERED VIA CALL         AT EP DGTFVA11..91.221
SA   053966F4  WD1 00000000   HSA 05301638   LSA 053B90BC   RET 853AB422   EPA 8539FCB0   R0  00000000
               R1  05396A20   R2  00000001   R3  00000000   R4  0003C37C   R5  0002A4D0   R6  00000000
               R7  00000001   R8  05396DB8   R9  053AC03D   R10 0539213C   R11 053966F4   R12 853AB03E
DGTFMD01 WAS ENTERED VIA CALL         AT EP DGTFCTPR..91.227
SA   053E80A4  WD1 00000000   HSA 053EA4E4   LSA 053E305C   RET 853FFB5A   EPA 0533F540   R0  053E83F4
               R1  053E827C   R2  053E8404   R3  0005BCD9   R4  053F3E80   R5  053EA874   R6  0005B832
               R7  0005A833   R8  053010CC   R9  053F3145   R10 0539213C   R11 053E80A4   R12 853FF476
```

Figure 4. ALTER attributes that can be altered and types of catalog entries

ALTER parameters

Required parameters: The ALTER command takes the following required and optional parameters.

entryname
This names the entry to be altered.

When attributes of a catalog are altered, entryname must include either the data or index components. Giving the catalog name alters attributes defined at the cluster level only. The catalog name is also the data component name.

The restricted prefix SYS1.VVDS.V or its generic form SYS1.VVDS.* or SYS1.*.V is not allowed as an entryname for the ALTER command.

If you are renaming a member of a non-VSAM partitioned data set, the entryname must given as: pdsname(membername).

See the NEWNAME parameter for information on renaming SMS-managed data sets.

You identify a general data stream (GDS) with its GDG name followed by the generation and version numbers of the data set (*GDGname*.G*xxxx*V*yy*). You cannot use relative generation numbers (that is, *GDGname*(+1)) with the entryname

Optional parameters:

ACCOUNT(*account–info***)**
Account is supported only for SMS-managed VSAM data sets or non-VSAM data sets.

account–info
Use this to change accounting information and user data for the data set. It *must* be between 1 and 32 bytes, otherwise you will receive an error message.

Abbreviation: ACCT

ADDVOLUMES(*volser* **[** *volser***])**
This provides the volumes to be added to the list of candidate volumes. You can use ALTER ADDVOLUMES to add candidate volumes to non-managed VSAM data sets and SMS-managed VSAM, non-VSAM, and general data stream (GDS) data sets. Only nonspecific volumes can be added to SMS-managed, non-VSAM data sets and GDS data sets. If an ALTER ADDVOLUMES is done to a data set already opened and allocated, the data set must be closed, unallocated, reallocated, and reopened before VSAM can extend onto the newly added candidate volume. Adding a nonexistent volume to the list can result in an error when the data set is extended. Ensure that the volume exists and is on-line before attempting to extend the data set.

Restriction: This does not work with NONSMS NONVSAM

SMS might not use candidate volumes for which you request specific volsers with the ADDVOLUMES parameter. Sometimes a user-specified volser for an SMS-managed data set results in an error. To avoid candidate-volume problems with SMS, you can have SMS choose the volser used for a candidate volume. To do this, you can code an * for each volser that you request with the ADDVOLUMES parameter. If, however, you request both specified and unspecified volsers in the same command, you must enter the specified volsers

first in the command syntax. The system does not allocate space on candidate volumes until VSAM extends to the candidate volume. This includes SMS-managed data sets with guaranteed space.

Abbreviation: AVOL

BUFFERSPACE(*size*)
Provides the amount of space for buffers. The size you specify for the buffer space helps VSAM determine the size. IBM recommends that the size you give is equal to or greater than the amount specified in the original definition. If the amount is less, VSAM attempts to get enough space to contain two data component control intervals and, if the data is key-sequenced, one index component control interval. You can specify BUFFERSPACE only for a catalog or for the data component of a cluster or alternate index. If you use BUFFERSPACE for a catalog, then you must specify the CATALOG parameter.

The BUFFERSPACE parameter is ignored for VSAM RLS and DFSMStvs access.

size
is the amount of space for buffers. This helps VSAM determine the size of the data component's and index component's control interval.

Size can be entered in decimal (n), hexadecimal (X'n'), or binary (B'n') form, but must not exceed 16,776,704. The specified size should not be less than the space needed to contain two data component control intervals and, if the data is key-sequenced, to contain one index control interval. If the given size is less than what VSAM requires, it gets it when the data set is opened.

Abbreviations BUFSP or BUFSPC

BUFND(*number*)
Gives the number of I/O buffers VSAM is to use for transmitting data between virtual and auxiliary storage. The size of the buffer area is the size of the data component control interval. Use this parameter only to alter the data component of an integrated catalog facility catalog.

The BUFND parameter is ignored for VSAM RLS and DFSMStvs access.

number
is the number of data buffers you can use. The minimum number is 3, and the maximum is 255.

Abbreviation: BFND

BUFNI(*number*)
Is the number of I/O buffers VSAM uses for transmitting the contents of index entries between virtual and auxiliary storage for keyed access. The size of the buffer area is the size of the index control intervals. Use this parameter only to alter the index component of the integrated catalog facility catalog.

The BUFNI parameter is ignored for VSAM RLS and DFSMStvs access .

number
Is the number of index buffers you can use. The minimum number is 2 and the maximum is 255.

Abbreviation: BFNI

BWO(TYPECICS|TYPEIMS|NO)
Use this parameter if backup-while-open (BWO) is allowed for the VSAM sphere. BWO applies only to SMS data sets and cannot be used with

TYPE(LINEAR). If BWO, LOG, or LOGSTREAMID is specified (or an RLS cell exists for the data set), access from DFSMS/MVS 1.2 or a lower-level system is denied.

If BWO is specified in the SMS data class, the specified BWO value is used as part of the data set definition, unless BWO was previously defined with an explicitly specified or modeled DEFINE attribute.

TYPECICS
Use TYPECICS to specify BWO in a CICS or DFSMStvs environment. For RLS processing, this activates BWO processing for CICS or DFSMStvs, or both. For non-RLS processing, CICS determines whether to use this specification or the specification in the CICS FCT. For more information about the use of TYPECICS, see the CICS Transaction Server for z/OS Information Center at http://publib.boulder.ibm.com/infocenter/cicsts/v3r1/index.jsp.

Exception: If CICS determines that it will use the specification in the CICS FCT, the specification might override the TYPECICS parameter for CICS processing.

Abbreviation: TYPEC

TYPEIMS
Enables BWO processing for IMS data sets. You can use this capability only with DFSMS 1.3 or higher-level DFSMS systems. If you attempt to open a cluster that has the TYPEIMS specification of a DFSMS 1.2 (or lower-level) system, the open will not be successful.

Abbreviation: TYPEI

NO Use this when BWO does not apply to the cluster.

Exception: If CICS determines that it will use the specification in the CICS FCT, the specification might override the NO parameter for CICS processing.

CATALOG(*catname*)
Specifies the catalog containing the entry to be altered.

To assign catalog names for SMS-managed data sets, you must have access to the RACF STGADMIN.IGG.DIRCAT facility class. For more information, see*z/OS DFSMS Access Method Services Commands*.

catname
Is the name of the catalog that contains the entry.

Abbreviation: CAT

CCSID(*value*)
Is the Coded Character Set Identifier attribute; it identifies:

- Encoding scheme identifier
- Character set identifier or identifiers
- Code page identifier or identifiers
- Additional coding required to uniquely identify the coded graphic used

You can use Coded Character Set Identifier (CCSID) only for system-managed data sets. If the CCSID parameter is not in the catalog at the time ALTER is called, it is created.

The *value* for CCSID can be specified in decimal (n), hexadecimal (X'n'), or binary (B'n'). The acceptable range of values is 0 (X'0') to 65535 (X'FFFF').

ECSHARING|NOECSHARING
Indicates whether sharing this catalog can be performed through the coupling facility.

ECSHARING
Enhanced catalog sharing (ECS) is allowed. ECS is a method of catalog sharing that makes use of a coupling facility to increase the performance of shared catalog requests. Read about ECS in *z/OS DFSMS Managing Catalogs* before enabling it for a catalog.

Abbreviation: ECSHR

NOECSHARING
Enhanced catalog sharing (ECS) is not allowed. This is the default. Catalog sharing is performed, but the ECS sharing method is not be used.

Abbreviation: NOECSHR

EMPTY|NOEMPTY
Specifies what is to happen when the maximum number of generation data sets has been cataloged. If the GDG is full (the LIMIT is reached), this attribute determines whether all GDSs or just the oldest GDSs are processed.

For an SMS-managed generation data set, if the NOSCRATCH attribute is used, the GDS is uncataloged from its GDG base and is recataloged outside its GDG base as an SMS non-VSAM entry with the rolled-off status.

EMPTY
Specifies that, when the maximum number of generation data sets is exceeded, all the generation data sets are uncataloged or deleted.

Abbreviation: EMP

NOEMPTY
Used when the maximum number of generation data sets is exceeded, only the oldest generation data set is uncataloged or deleted.

Abbreviation: NEMP

ERASE|NOERASE
Indicates whether to erase the component when its entry in the catalog is deleted.

ERASE
Overwrites the component with binary zeros when its catalog entry is deleted. If the cluster or alternate index is protected by a RACF generic or discrete profile, use RACF commands to assign an ERASE attribute as part of this profile so that the data component is automatically erased upon deletion.

Abbreviation: ERAS

NOERASE
Specifies the component is not to be overwritten with binary zeros when its catalog entry is deleted. NOERASE resets only the indicator in the catalog entry that was created from a prior DEFINE or ALTER command. If the cluster or alternate index is protected by a RACF generic or discrete profile that specifies the ERASE attribute, it is erased upon deletion. Only RACF commands can be used to alter the ERASE attribute in a profile.

Abbreviation: NERAS

EXCEPTIONEXIT(*entrypoint*)
Is the name of the user-written routine that receives control if an exception

(usually an I/O error) occurs while the entry's object is being processed. An exception is any condition that causes a SYNAD exit. The object's exception exit routine is processed first, then the user's SYNAD exit routine receives control.

Abbreviation: EEXT

FILE(*ddname***)**
Specifies one of the following:

- The name of a DD statement that describes the volume that contains the data set to be altered.
- The name of a DD statement that identifies the volume of an entry that will be renamed. The entry must be a non-VSAM data set or the data or index component of a cluster, alternate index, or page space.
- The name of a DD statement that describes a partitioned data set when a member is to be renamed.

If you identify multiple volumes of different device types with FILE, use concatenated DD statements. If you specify ADDVOLUMES or REMOVEVOLUMES, the volume being added or removed must be identified. If FILE is not specified, an attempt is made to dynamically allocate the object's data set. Therefore, the object's volume must be mounted as permanently resident or reserved.

Note: While the FILE parameter can preallocate a volume where the data set resides, it does not direct the ALTER request to the data set to be altered. Instead, a catalog search is done to locate the data set to be altered.

FILEDATA(TEXT|BINARY)
Use one of the following:

TEXT
Specifies that the data in the data set is text. If the data set is read or written across the network, the data in this data set is EBCDIC on z/OS and ASCII on the workstation.

BINARY
Specifies that data is to be processed as is.

FREESPACE(*CI-percent***[** *CA-percent***])**
Indicates the percent of free space left after any allocation. CI-percent is a percentage of the amount of space to be preserved for adding new records and updating existing records, with an increase in the length of the record. Since a CI is split when it becomes full, the CA might also need to be split when it is filled by CIs created by a CI split. The amounts, as percentages, must be equal to, or less than, 100. If you use 100% of free space, one record is placed in each control interval and one control interval is placed in each control area (CA).

Use this parameter to alter the data component of a cluster, alternate index, or catalog.

If the FREESPACE is altered after the data set has been loaded, and sequential insert processing is used, the allocation of free space is not honored.

Abbreviation: FSPC

FRLOG(NONE|REDO)
Specifies whether VSAM batch logging can be performed for your VSAM data set. VSAM batch logging is available with CICS VSAM Recovery V3R1.

NONE
Disables the VSAM batch logging function for your VSAM data set. Changes made by applications are not written to the MVS log stream indicated in the LOGSTREAMID parameter.

REDO
Enables the VSAM batch logging function for your VSAM data set. Changes made by applications are written to the MVS log stream indicated in the LOGSTREAMID parameter. If you specify FRLOG(REDO), you must also specify LOGSTREAMID for that data set, unless the log stream is already defined.

Restrictions:

1. Use the FRLOG parameter only if you want to enable (REDO) or disable (NONE) VSAM batch logging. Do not use the FRLOG parameter for data sets that are not intended for use with VSAM batch logging.
2. If FRLOG is specified, these rules apply to the data set:
   - Must be SMS-managed
   - Cannot be LINEAR or a temporary data set

INHIBIT|UNINHIBIT
Specifies whether the entry being altered can be accessed for any operation or only for read operations.

INHIBIT
Used when the entry being altered is to be read only.

Abbreviation: INH

UNINHIBIT
Indicates that the read-only restriction set by a previous ALTER or EXPORT command is to be removed.

Abbreviation: UNINH

KEYS(*length offset*)
Specifies the length and offset of the object's key. If the altered entry defines an alternate index, offset applies to the alternate key in the data records in the base cluster.

Restrictions: Use KEYS if all the following are true:

- The object whose entry is being altered is an alternate index, a path, a key-sequenced cluster, or a data component of a key-sequenced cluster or alternate index.
- The object whose entry is being altered contains no data records.
- The values for KEYS in the object's catalog entry are default values. For default values, see the DEFINE command for the object.
- The new values for KEYS do not conflict with the control interval size specified when the object was defined.
- The key fits within the record whose length is specified by the RECORDSIZE parameter.
- The key fits in the first record segment of a spanned record.

length offset
Is the length of the key (between 1 and 255), in bytes, and its displacement from the beginning of the data record, in bytes.

If the values for KEYS in the object's catalog entry are not default values and ALTER KEYS specifies those same values, processing continues for any other parameters specified in the command, and no error message is issued.

LOG(NONE|UNDO|ALL)
Establishes whether the sphere to be accessed with VSAM RLS or DFSMStvs is recoverable or nonrecoverable. It also indicates whether a recovery log is available for the sphere. LOG applies to all components in the VSAM sphere.

NONE
Indicates that neither an external backout nor a forward recovery capability is available for the spheres accessed in VSAM RLS or DFSMStvs mode. If you use this, VSAM RLS and DFSMStvs consider the sphere to be nonrecoverable.

UNDO
Specifies that changes to the sphere accessed in VSAM RLS or DFSMStvs mode can be backed out using an external log. VSAM RLS and DFSMStvs consider the sphere recoverable when you use LOG(UNDO).

ALL
Specifies that changes to the sphere accessed in VSAM RLS or DFSMStvs mode can be backed out and forward recovered using external logs. VSAM RLS and DFSMStvs consider the sphere recoverable when you use LOG(ALL). When you specify LOG(ALL), you must also specify the LOGSTREAMID parameter, unless it is already defined.

VSAM RLS allows concurrent read and update sharing for nonrecoverable spheres through commit (CICS) and noncommit protocol applications. For a recoverable sphere, a noncommit protocol application must use DFSMStvs to be able to open the sphere for update using RLS access.

Restriction: LOG cannot be used with LINEAR.

LOGSTREAMID(*logstream*)
Changes or adds the name of the forward recovery log stream. It applies to all components in the VSAM sphere.

logstream
Is the name of the forward recovery log stream. This can be a fully qualified name up to 26 characters, including separators. This parameter is required if you have specified LOG(ALL).

For information about defining log streams for CICS use, see the CICS Transaction Server for z/OS Information Center at http://publib.boulder.ibm.com/infocenter/cicsts/v3r1/index.jsp.

Abbreviation: LSID

Restriction: LOGSTREAMID cannot be used with LINEAR.

LIMIT(*limit*)
Used to modify the maximum number (between 1 and 255) of active generation data sets that might be associated with a generation data group base.

limit
If the limit is less than the current number of active generations, the oldest generations are rolled off until the new limit is satisfied. Any generation data sets that are rolled off by this command are listed showing their new

status (recataloged, uncataloged, or deleted). For more information about limit processing of a GDS, see *z/OS DFSMS Managing Catalogs*.

If the limit is greater than the current number of active generations, it does not cause the roll-in of existing rolled off GDSs. For this function, see the ROLLIN parameter.

LOCK|UNLOCK
Controls the setting of the catalog lock attribute, and therefore checks access to a catalog. Use LOCK or UNLOCK when the entry name identifies an integrated catalog facility catalog. If the LOCK|UNLOCK parameter is not specified, the status of the integrated catalog facility catalog lock attribute is not changed. Before you lock a catalog, review the information on locking catalogs in *z/OS DFSMS Managing Catalogs*.

LOCK
Is used when the catalog identified by entryname is to be locked. Locking a catalog makes it inaccessible to all users without read authority to RACF facility class profile IGG.CATLOCK (including users sharing the catalog on other systems).

For protected catalogs, locking an unlocked catalog requires ALTER authority for the catalog being locked, and read authority to RACF facility profile IGG.CATLOCK. For unprotected catalogs, locking an unlocked catalog requires read authority to RACF facility class profile IGG.CATLOCK.

UNLOCK
Specifies that the catalog identified by entryname is to be unlocked. For RACF and nonprotected catalogs, unlocking a locked catalog requires read authority to RACF facility class profile IGG.CATLOCK.

MANAGEMENTCLASS(*class*)
For SMS-managed data sets: Gives the name, 1 to 8 characters, of the management class for a data set. Your storage administrator defines the names of the management classes you can include. If MANAGEMENTCLASS is used for a non-SMS-managed data set, or if SMS is inactive, the ALTER command is unsuccessful.

When the storage or management class is altered for a DFSMShsm migrated data set, ALTER will not recall the data set to make the change, provided no other parameters are specified.

You must have RACF access authority to alter the management class.

Abbreviation: MGMTCLAS

NEWNAME(*newname*)
Indicates that the entry to be altered is to be given a new name.

When you rename an SMS-managed data set residing on DASD, the MGMTCLAS ACS routine is called and lets you reassign a new management class.

You can use ALTER NEWNAME to rename SMS-managed generation data sets (GDS). Table 8 on page 72 shows how NEWNAME resolves renaming a GDS under different conditions. You can successfully rename the following:

- An SMS-managed GDS to an SMS-managed non-VSAM data set
- An SMS-managed non-VSAM data set to an SMS-managed GDS
- An SMS-managed GDS to another SMS-managed GDS

Restriction: You cannot alter the data portion of a page space data set to a new name. Also, catalog names and catalog component names cannot be renamed.

You might not be able to rename a data set if you are changing the high-level qualifiers of the data set's name and those qualifiers are an alias name of a catalog. (The number of high-level qualifiers used to form an alias can be one to four, depending on the multilevel alias search level used at your installation.)

If you are changing a high-level qualifier, NEWNAME acts differently, depending on whether the data set being renamed is SMS-managed or non-SMS-managed, and whether the data set has aliases or not. Table 8 shows how NEWNAME resolves under different conditions.

Table 8. How NEWNAME resolves when change of catalog is required. This table shows how NEWNAME resolves when change of catalog is required.

Data set type	SMS	Non-SMS
VSAM	ALTER unsuccessful—entry not renamed	ALTER successful—entry remains in the source catalog
non-VSAM with no aliases	ALTER successful—entry is recataloged in target catalog.	ALTER successful—entry remains in the source catalog
non-VSAM with aliases	ALTER unsuccessful—entry not renamed	ALTER successful—entry remains in the source catalog
GDS with no aliases	ALTER successful—entry is recataloged in target catalog.	ALTER unsuccessful—entry not renamed
GDS with aliases	ALTER unsuccessful—entry not renamed	ALTER unsuccessful—entry not renamed
Note: The source catalog is the catalog containing the original entry. The target catalog is the catalog in which the new name would normally be cataloged according to a catalog alias search.		

If you want to define a data set into a particular catalog, and that catalog is not the one chosen according to the regular search, then you must have authority to RACF STGADMIN.IGG.DIRCAT facility class. For more information on this facility class see *z/OS DFSMSdfp Storage Administration*.

To give an altered entry a new name:

- Unless the data set being renamed is a path, the data set's volume must be mounted because the volume table of contents (VTOC) is modified.

  You can use the FILE parameter to supply a JCL DD statement to allocate the data set. If you do not supply a DD statement, an attempt is made to allocate the data set dynamically. The volume must be mounted as either permanently resident or reserved.

  If another program has access to the data set while this is being done, the program might not be able to access the data set after it is renamed. This can result in an error.
- If you include generic names, you must define both entryname and newname as generic names.
- If you are renaming a member of a non-VSAM partitioned data set, the newname must be specified in the format: pdsname(membername).
- If you are renaming a VSAM data set that is RACF protected, the existing RACF data set profile will be renamed.
- If you are using ALTER NEWNAME, you must have one of these:

- ALTER authority for the data set or for the catalog
- ALTER authority for the new name (generic profile) or CREATE authority for the group

- If there is a data set profile for the new data set name prior to the ALTER command, the command ends, and the data set name and protection attributes remain unchanged.

  If the old profile is not found or cannot be altered to the new name, the NEWNAME action is not completed in the catalog, and an error message indicates why the action is not completed.

  If renaming is unsuccessful, it is possible that either the object exists with both the original name and the new name, or that the data set was not closed.

Abbreviation: NEWNM

NULLIFY([AUTHORIZATION(MODULE|STRING)]

[BWO][CODE][EXCEPTIONEXIT]
[LOG][LOGSTREAMID][OWNER]
[RETENTION])

Specifies that the protection attributes identified by Subparameters of NULLIFY are to be nullified. Attributes are nullified before any respecification of attributes is done.

Abbreviation: NULL

AUTHORIZATION(MODULE|STRING)

Is used when the user authorization routine or the user authorization record is to be nullified.

Abbreviation: AUTH

MODULE

Removes the module name from the catalog record, but the module itself is not to be deleted. Both the user authorization routine and the user authorization record (character string) are nullified.

Abbreviation: MDLE

STRING

Nullifies the authorization record, but the corresponding module is not nullified.

Abbreviation: STRG

BWO

Use this parameter to remove the BWO specification from the sphere.

CODE

Nullifies the code name used for prompting.

EXCEPTIONEXIT

Nullifies the entry's exception exit. The module name is removed from the catalog record, but the exception-exit routine itself is not deleted.

Abbreviation: EEXT

LOG

Nullifies the log parameter.

Access to the RLS sphere is not permitted when the log parameter is nullified.

LOGSTREAMID
When you use this, the name of the forward recovery log stream is nullified. NULLIFY(LOGSTREAMID) is not allowed if the data set has a value of LOG(ALL).

Abbreviation: LSID

OWNER
Nullifies the owner identification.

RETENTION
Nullifies the retention period that was used in a TO or FOR parameter.

Abbreviation: RETN

OWNER(*ownerid***)**
Specifies the owner identification for the entry being altered.

RECORDSIZE(*average maximum***)**
Specifies new average and maximum lengths for data records contained in the object whose entry is being altered.

If the object whose entry is being altered is a path pointing to the alternate index, the alternate index is altered; if it is a path pointing directly to the base cluster, the base cluster is altered.

If the object whose entry is being altered is an alternate index, the length of the alternate key must be within the limit specified by maximum.

Restrictions: RECORDSIZE is used only if all the following are true:

- The object whose entry is being altered is an alternate index, a cluster, a path, or a data component.
- The object whose entry is being altered contains no data records.
- The maximum RECORDSIZE in the object's catalog entry is the default. For defaults, see the DEFINE command for the object.
- If NONUNIQUEKEY is used for an alternate index, the record length to be specified accounts for the increased record size; this results from the multiple prime key pointers in the alternate index data record.
- Use a maximum record length of at least seven bytes less than the control interval size, unless the record is a spanned record.
- Use a record length large enough to contain all prime and alternate keys previously defined.

If RECORDSIZE in the object's catalog entry is not the default, and ALTER RECORDSIZE specifies that same value, processing continues for any other parameters given in the command, and there is no error message.

Abbreviation: RECSZ

REMOVEVOLUMES(*volser***[** *volser***])**
Specifies volumes to be removed from the list of candidate volumes associated with the entry being altered. The name of the data or index component must be specified in the ENTRYNAME parameter. If you are also adding volumes, the volumes to be removed are removed after the new volumes are added to the candidate list. Only nonspecific volumes can be removed from SMS-managed, non-VSAM data sets, and GDS data sets. For information on volume cleanup, see *z/OS DFSMS Managing Catalogs*.

SMS might not use candidate volumes for which you request specific volsers. Some user-specified volsers for an SMS-managed data set can result in an error. To avoid candidate volume problems with SMS, you can request that

SMS choose the given volser used for a candidate volume. To do this, you can code an * for each volser that you request. If, however, you request both specified and unspecified volsers in the same command, you must enter the specified volsers first in the command syntax.

To ensure that the operation has completed correctly, the execution of ALTER REMOVEVOLUMES should be followed by a listing of the VTOC on the target volume. If ALTER REMOVEVOLUMES did not scratch any data sets allocated to job steps, it can still complete with return code zero. In the integrated catalog facility environment, both the basic catalog structure (BCS) and the VSAM volume data set (VVDS) might be allocated to another job or TSO/E user. If so, these entities are not scratched, and any future access method services commands that depend on ALTER REMOVEVOLUMES completing normally might be unsuccessful. To ensure that the operation has completed correctly, follow the execution of ALTER REMOVEVOLUMES with a listing of the VTOC on the target volume.

Exceptions:

1. If a volume to be removed contains data that belongs to the entry being altered, the volume is not removed.
2. Volume cleanup is not supported if the volume is SMS managed.

Abbreviation: RVOL

REUSE|NOREUSE
Controls setting the REUSE indicator for VSAM data sets. A data set that requires the REUSE attribute be changed to "reusable" cannot be an alternate index nor can it have an associated alternate index. The data set also cannot be a key-sequenced data set (KSDS) with one or more key ranges.

ROLLIN
Indicates whether a generation data set is to be rolled-in. The generation data set must be SMS managed and in either a deferred rolled-in or rolled-off state. For more information, see *z/OS DFSMS Using Data Sets*.

Abbreviation: ROL

SCRATCH|NOSCRATCH
Specifies whether generation data sets, when they are uncataloged, are to be removed from the VTOC of the volume where they reside.

SCRATCH
Removes the data set's format-1 DSCB from the VTOC so that the data set can no longer be accessed, and, for SMS-managed data sets, the non-VSAM volume record (NVR) is removed from the VVDS.

Abbreviation: SCR

NOSCRATCH
Indicates that the data set's format-1 DSCB is not to be removed from the VTOC and, for SMS-managed data sets, the NVR entry remains in the VVDS.

Abbreviation: NSCR

SHAREOPTIONS(*crossregion***[** *crosssystem***])**
Is used when a data or index component of a cluster, alternate index, or the data component of a catalog can be shared among users. However, SMS-managed volumes, and catalogs containing SMS-managed data sets, must not be shared with non-SMS systems. (For a description of data set sharing, see *z/OS DFSMS Using Data Sets*.)

The value of SHAREOPTIONS is assumed to be (3,3) when the data set is accessed in VSAM RLS or DFSMStvs mode.

crossregion
Specifies the amount of sharing allowed among regions within the same system or within multiple systems using global resource serialization (GRS). Independent job steps in an operating system, or multiple systems in a GRS ring, can access a VSAM data set concurrently. For a description of GRS, see *z/OS MVS Planning: Global Resource Serialization*. Option 3 is the only one applicable for altering a catalog. To share a data set, each user must code DISP=SHR in the data set's DD statement. You can use the following options:

OPT 1 The data set can be shared by any number of users for read processing, or the data set can be accessed by only one user for read and write processing. VSAM ensures complete data integrity for the data set. This setting does not allow any non-RLS access when the data set is already open for VSAM RLS or DFSMStvs processing. A VSAM RLS or DFSMStvs open will fail with this option if the data set is already open for any processing.

OPT 2 The data set can be accessed by any number of users for read processing, and it can also be accessed by one user for write processing. It is the user's responsibility to provide read integrity. VSAM ensures write integrity by obtaining exclusive control for a control interval while it is being updated. A VSAM RLS or DFSMStvs open is not allowed while the data set is open for non-RLS output.

If the data set has already been opened for VSAM RLS or DFSMStvs processing, a non-RLS open for input is allowed; a non-RLS open for output fails. If the data set is opened for input in non-RLS mode, a VSAM RLS or DFSMStvs open is allowed.

OPT 3 The data set can be fully shared by any number of users. The user is responsible for maintaining both read and write integrity for the data the program accesses. This setting does not allow any non-RLS access when the data set is already open for VSAM RLS or DFSMStvs processing. If the data set is opened for input in non-RLS mode, a VSAM RLS or DFSMStvs open is allowed.

This option is the only one applicable to a catalog.

OPT 4 The data set can be fully shared by any number of users. For each request, VSAM refreshes the buffers used for direct processing. This setting does not allow any non-RLS access when the data set is already open for RLS processing. If the data set is opened for input in non-RLS mode, a VSAM RLS or DFSMStvs open is allowed.

As in SHAREOPTIONS 3, each user is responsible for maintaining both read and write integrity for the data the program accesses.

crosssystem
Is the amount of sharing allowed among systems. Job steps of two or more operating systems can gain access to the same VSAM data set regardless of the disposition specified in each step's DD statement for the data set. To get exclusive control of the data set's volume, a task in one system issues the RESERVE macro. The level of cross-system sharing allowed by VSAM applies only in a multiple operating system environment.

The cross-system sharing options are ignored by VSAM RLS or DFSMStvs processing. The values follow:

1 Reserved.

2 Reserved.

3 Specifies that the data set can be fully shared. With this option, each user is responsible for maintaining both read and write integrity for the data the program accesses. User programs that ignore write integrity guidelines can cause VSAM program checks, uncorrectable data set problems, and other unpredictable results. The RESERVE and DEQ macros are required with this option to maintain data set integrity. (For information on using RESERVE and DEQ, see *z/OS MVS Programming: Authorized Assembler Services Reference ALE-DYN* and *z/OS MVS Programming: Authorized Assembler Services Reference LLA-SDU*.)

4 Specifies that the data set can be fully shared. For each request, VSAM refreshes the buffers used for direct processing. This option requires that you use the RESERVE and DEQ macros to maintain data integrity while sharing the data set. Improper use of the RESERVE macro can cause problems similar to those described under SHAREOPTIONS 3. (For information on using RESERVE and DEQ, see *z/OS MVS Programming: Authorized Assembler Services Reference ALE-DYN* and *z/OS MVS Programming: Authorized Assembler Services Reference LLA-SDU*.)

Output processing is limited to update or add processing that does not change either the high-used relative byte address (RBA) or the RBA of the high key data control interval if DISP=SHR is specified.

Abbreviation: SHR

STORAGECLASS(*class*)

For SMS-managed data sets: Gives the name, 1 to 8 characters, of the storage class. Your storage administrator defines the names of the storage classes you can assign. A storage class is assigned when you specify STORAGECLASS or an installation-written automatic class section (ACS) routine selects a storage class when the data set is created. Use the storage class to provide the storage service level to be used by SMS for storage of the data set. The storage class provides the storage attributes that are specified on the UNIT and VOLUME operand for non-SMS-managed data sets.

When the storage or management class is altered for a DFSMShsm migrated data set, ALTER will not recall the data set to make the change, provided no other parameters are specified.

You must have RACF access authority to alter the storage class.

If STORAGECLASS is used for a non-SMS-managed data set or if SMS is inactive, the ALTER command is unsuccessful.

Abbreviation: STORCLAS

STRNO(*number*)

Specifies the number of concurrent catalog positioning requests that VSAM should manage. Use this parameter to alter the data component of a catalog. The STRNO setting is ignored when the data set is opened for RLS or DFSMStvs.

number
Is the number of concurrent requests VSAM must manage. The minimum number is 2, the maximum is 255.

TO(*date***)|FOR(***days***)**
Specifies the retention period for the entry being altered.

You cannot use these parameters for the data or index components of clusters or alternate indexes. For catalogs, you must use the data component name. The expiration date in the catalog is updated, and, for SMS-managed data sets, the expiration date in the format-1 DSCB is changed. Enter a LISTCAT command to see the correct expiration date.

The MANAGEMENTCLASS maximum retention period, if specified, limits the retention period specified by this parameter.

TO(*date***)**
Specifies the date up to which an entry should be kept before it is allowed to be deleted. The date is specified in the form [yy]yyddd, where yyyy is a four-digit year, yy is a two-digit year, and ddd is the three-digit (001 through 366) day of the year. Two-digit years are treated as if "19" is specified as the first two digits of yyyy. The dates (19)99365 and (19)99366 are considered never-expire dates.

For expiration dates of January 1, 2000, and later, you must use the form TO(yyyyddd).

FOR(*days***)**
Is used to choose the number of days to keep the entry. The maximum number is 9999. If the number is 0 through 9998, the entry is retained for the number of days indicated; if the number is 9999, the entry is retained indefinitely.

TYPE(LINEAR)
Specifies that the VSAM data set type of an entry-sequenced data set (ESDS) is to be changed to linear. The contents of the data set are not modified. Only an ESDS with a CI size of 4096 is eligible to be a linear data set. A linear data set's type cannot be changed. After you have changed an ESDS set to a linear data set, the data set must remain a linear data set; you cannot change it back into an ESDS.

LINEAR
Changes the VSAM data type ESDS to a linear data set (LDS).

Abbreviation: LIN

UNIQUEKEY|NONUNIQUEKEY
Specifies whether the alternate key value can be found in more than one of the base cluster's data records.

UNIQUEKEY
Makes each alternate key value unique. If the same alternate key value is found in more than one of the base cluster's data records, an error results.

You can use UNIQUEKEY for an empty alternate index (that is, an alternate index that is defined but not yet built).

Abbreviation: UNQK

NONUNIQUEKEY
Allows an alternate key value to point to more than one data record in the cluster. NONUNIQUEKEY can be specified for an alternate index at any time.

If the alternate index is empty, you should also consider defining RECORDSIZE to ensure that each alternate index record is large enough to contain more than one data record pointer.

Abbreviation: NUNQK

UPDATE|NOUPDATE
Specifies whether a base cluster's alternate index upgrade set is to be allocated when the path's name is allocated.

The NOUPDATE setting is ignored when the data set is opened for VSAM RLS or DFSMStvs. Alternate indexes in the upgrade set are opened as if UPDATE was specified.

UPDATE
Allocates the cluster's alternate index upgrade set when the path's name is allocated with a DD statement.

Abbreviation: UPD

NOUPDATE
Specifies that the cluster's alternate index upgrade set is not to be allocated but the path's cluster is to be allocated. You can use NOUPDATE to open a path. If the path shares a control block structure that uses UPDATE, this indicates the upgrade set has been allocated and, in this case, the upgrade set can be updated.

Abbreviation: NUPD

UPGRADE|NOUPGRADE
Shows whether an alternate index is to be upgraded (to reflect the changed data) when its base cluster is modified.

UPGRADE
Indicates that the cluster's alternate index is upgraded (to reflect the changed data) when the cluster's records are added to, updated, or erased. If UPGRADE is used when the cluster is open, the upgrade attribute does not apply to the alternate index until the cluster is closed and then opened (that is, a new set of VSAM control blocks describes the cluster and its attributes).

Use UPGRADE for an empty alternate index (that is, an alternate index that is defined but not built). However, the UPGRADE attribute is not effective for the alternate index until the alternate index is built (see the BLDINDEX command).

Abbreviation: UPG

NOUPGRADE
Specifies the alternate index is not to be modified when the its base cluster is modified. NOUPGRADE can be use as an alternate index at any time.

Abbreviation: NUPG

WRITECHECK|NOWRITECHECK
Specifies whether a data or index component is to be checked by a machine action called write check when a record is written into it. This parameter can be specified to alter the data or index components of a cluster, an alternate index, or catalog.

The WRITECHECK setting is ignored when the data set is opened for VSAM RLS or DFSMStvs access.

WRITECHECK
Writes and reads a record without data transfer, to test for the data check condition.

Abbreviation: WCK

NOWRITECHECK
Writes the record only

Abbreviation: NWCK

ALTER examples

Alter a cluster's attributes using SMS keywords: Example 1: In this example, the ALTER command is used with the MANAGEMENTCLASS and STORAGECLASS keywords.

```
//ALTER    JOB  ...
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *
    ALTER  -
           CLUS.ALTER.EXAMPLE -
           MANAGEMENTCLASS(VSAM) -
           STORAGECLASS(FAST) -
           LOG(ALL) -
           LOGSTREAMID(LogA)
/*
```

The ALTER command modifies some of the attributes of SMS-managed data set CLUS.ALTER.EXAMPLE. It is expected to grow and require an increase in the frequency of backup, availability and performance. The parameters are MANAGEMENTCLASS, indicating a new management class of VSAM, and STORAGECLASS, indicating a storage class of FAST.

LOG(ALL) specifies that changes to the sphere accessed in RLS and DFSMStvs mode can be backed out and forward recovered using external logs. LOGSTREAMID gives the name of the forward recovery log stream.

Roll-In a generation data set: Example 2: In this example, the ALTER command is used with the ROLLIN keyword to roll-in a generation data set currently in the deferred roll-in state.

```
//ALTER    JOB  ...
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *
     ALTER -
           DATA.G0001V05 -
           ROLLIN
/*
```

The ALTER command rolls the SMS-managed generation data set, DATA.G0001V05, into the GDG base.

Alter the entry names of generically named clusters: Example 3: In this example, several clusters with similar names, GENERIC.*.BAKER (where * is any 1- to 8-character simple name), are renamed so that their entry names are GENERIC.*.ABLE. The name “GENERIC.*.BAKER” is called a generic name.

```
//ALTER2   JOB   ...
//STEP1    EXEC  PGM=IDCAMS
//SYSPRINT DD    SYSOUT=A
//SYSIN    DD    *
```

```
     ALTER -
           GENERIC.*.BAKER -
           NEWNAME(GENERIC.*.ABLE)
/*
```

The ALTER command changes each generic entry name, GENERIC.*.BAKER, to GENERIC.*.ABLE. Its parameters are:

- GENERIC.*.BAKER identifies the objects to be modified.
- NEWNAME changes each generic entry name GENERIC.*.BAKER to GENERIC.*.ABLE.

Alter the attributes of a generation data group: Example 4: This example modifies the attributes of a generation data group. Because the attributes are cataloged in the generation data group's base catalog entry, only this entry is modified.

```
//ALTER3   JOB   ...
//STEP1    EXEC  PGM=IDCAMS
//SYSPRINT DD    SYSOUT=A
//SYSIN    DD    *
     ALTER -
           GDG01 -
           NOEMPTY -
           SCRATCH
/*
```

The ALTER command modifies some of the attributes of generation data group GDG01. The new attributes override any previously used for the GDG. Its parameters are:

- GDG01 identifies the object to be modified.
- NOEMPTY uncatalogs only the oldest generation data set when the maximum number of cataloged generation data sets is exceeded.
- SCRATCH removes the generation data set's DSCB from the volumes' VTOC when the data set is uncataloged. If the data set is SMS-managed, the NVR is also removed.

Alter a data set expiration date: Example 6: In this example, an ALTER command is used to modify the expiration date of data set MOD.ALTER.EXAMPLE with the keyword TO.

```
//ALTER5   JOB    ...
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *
     ALTER -
           MOD.ALTER.EXAMPLE -
           TO(1989123)
/*
```

The command's parameters follow:

- MOD.ALTER.EXAMPLE is the name of the data set.
- TO changes the expiration date of the data set by name. The year (1989) is a four-digit number, concatenated with the day (123). You can also use two digits (89) to indicate the year. For expiration dates beyond the year 1999, a four-digit year must be specified.

Migrate a DB2® cluster to a linear data set cluster: Example 7: In this example, ALTER is used to alter a DB2 cluster EXAMPLE.ABC01, to a linear data set cluster.

```
//DB2TOLDS  JOB    ...
//STEP1     EXEC PGM=IDCAMS
//SYSPRINT  DD   SYSOUT=A
//SYSIN     DD   *
     ALTER  -
            EXAMPLE.ABC01 -
            TYPE(LINEAR)
/*
```

The command's parameter TYPE(LINEAR) requests ALTER change the data set type from ESDS to LDS.

Alter a cluster name and the associated data and index names: Example 8: In this example, ALTER is used to rename a cluster and its associated data and index entries.

```
//EXAMPL   JOB   ...
//STEP1    EXEC  PGM=IDCAMS
//SYSPRINT DD    SYSOUT=A
//SYSIN    DD    *
    DEFINE CLUSTER -
           (NAME(EXAMPLE.KSDS) -
           TRK(1 1) -
           VOL (338001)) -
           DATA -
             (NAME(EXAMPLE.KSDS.DATA)) -
           INDEX -
             (NAME(EXAMPLE.KSDS.INDEX))
        ALTER -
           EXAMPLE.KSDS -
           NEWNAME(EXAMPLE.TEST)
        ALTER -
           EXAMPLE.KSDS.* -
           NEWNAME(EXAMPLE.TEST.*)
/*
```

In the first part of the example, DEFINE CLUSTER defines a cluster and its data and index components with the same high-level qualifier, with these names:

- EXAMPLE.KSDS
- EXAMPLE.KSDS.DATA
- EXAMPLE.KSDS.INDEX

In the second part of the example, ALTER renames the cluster and its components.

The first ALTER command parameters are:

- EXAMPLE.KSDS identifies the object to be modified (cluster component previously defined).
- NEWNAME changes the entry name EXAMPLE.KSDS to EXAMPLE.TEST. This alters the cluster name to:
  - EXAMPLE.TEST

The second ALTER command parameters are:

- EXAMPLE.KSDS.* identifies the objects to be modified (data and index components previously defined).
- NEWNAME changes each generic entry name EXAMPLE.KSDS.* to EXAMPLE.TEST.*. This alters the data and index names to:
  - EXAMPLE.TEST.DATA
  - EXAMPLE.TEST.INDEX

Attention: Use the second example of the ALTER command with caution. *Any* data set with the first two qualifiers EXAMPLE.KSDS will be altered.

Defining alternate indexes

The DEFINE ALTERNATEINDEX command defines an alternate index. Use it to show attributes for the alternate index as a whole and for the components of the alternate index. The syntax of the DEFINE ALTERNATEINDEX command follows:

- **DEFINE ALTERNATEINDEX** (*parameters*)
  - [**DATA**(*parameters*)]
  - [**INDEX**(*parameters*)]
  - [**CATALOG**(*subparameters*)]

The syntax of the DEFINE ALTERNATEINDEX command.

Command	Parameters
DEFINE	**ALTERNATEINDEX** **(NAME(***entryname***)** **RELATE(***entryname***)** **{CYLINDERS(***primary***[** *secondary***])** | **KILOBYTES(***primary***[** *secondary***])** | **MEGABYTES(***primary***[** *secondary***])** | **RECORDS(***primary***[** *secondary***])** | **TRACKS(***primary***[** *secondary***])}** **VOLUMES(***volser***[** *volser***...])** **[BUFFERSPACE(***size***)]** **[CONTROLINTERVALSIZE(***size***)]** **[DATACLASS(***class***)]** **[ERASE | NOERASE]** **[EXCEPTIONEXIT(***entrypoint***)]** **[FILE(***ddname***)]** **[FREESPACE(***CI-percent***[** *CA-percent***]** | &cont; **0 0)]** **[KEYS(***length offset* | **64 0)]** **[MODEL(***entryname***[** *catname***])]** **[OWNER(***ownerid***)]** **[RECATALOG | NORECATALOG]** **[RECORDSIZE(***average maximum* | &cont; **4086 32600)]** **[REUSE | NOREUSE]** **[SHAREOPTIONS(***crossregion***[** *crosssystem***]** | &cont; **1 3)]** **[SPEED | RECOVERY]** **[TO(***date***) | FOR(***days***)]** **[UNIQUEKEY | NONUNIQUEKEY]** **[UPGRADE | NOUPGRADE]** **[WRITECHECK | NOWRITECHECK])** **[DATA (** **{CYLINDERS(***primary***[** *secondary***])** | **KILOBYTES(***primary***[** *secondary***])** | **MEGABYTES(***primary***[** *secondary***])** | **RECORDS(***primary***[** *secondary***])** | **TRACKS(***primary***[** *secondary***])}** **[VOLUMES(***volser***[** *volser***...])]** **[ATTEMPTS(***number***)]** **[AUTHORIZATION(***entrypoint***[** *string***])]** **[BUFFERSPACE(***size***)]** **[CODE(***code***)]** **[CONTROLINTERVALSIZE(***size***)]**

The syntax of the DEFINE ALTERNATEINDEX command.

Command	Parameters
	[ERASE | NOERASE] **[EXCEPTIONEXIT(***entrypoint***)]** **[FILE(***ddname***)]** **[FREESPACE(***CI-percent***[** *CA-percent***])]** **[KEYS(***length offset***)]** **[MODEL(***entryname* **[** *catname*&cont; **])]** **[NAME(***entryname***)]** **[OWNER(***ownerid***)]** **[RECORDSIZE(***average maximum***)]** **[REUSE | NOREUSE]** **[SHAREOPTIONS(***crossregion***[** *crosssystem***])]** **[SPEED | RECOVERY]** **[UNIQUEKEY | NONUNIQUEKEY]** **[WRITECHECK | NOWRITECHECK])]** **[INDEX** (**{CYLINDERS(***primary***[** *secondary***]) |** **KILOBYTES(***primary***[** *secondary***]) |** **MEGABYTES(***primary***[** *secondary***]) |** **RECORDS(***primary***[** *secondary***]) |** **TRACKS(***primary***[** *secondary***])}** **[VOLUMES(***volser***[** *volser***...])]** **[ATTEMPTS(***number***)]** **[AUTHORIZATION(***entrypoint***[** *string***])]** **[CODE(***code***)]** **[CONTROLINTERVALSIZE(***size***)]** **[EXCEPTIONEXIT(***entrypoint***)]** **[FILE(***ddname***)]** **[MODEL(***entryname***[** *catname*&cont; **])]** **[NAME(***entryname***)]** **[OWNER(***ownerid***)]** **[REUSE | NOREUSE]** **[SHAREOPTIONS(***crossregion***[** *crosssystem***])]** **[WRITECHECK | NOWRITECHECK])]** **[CATALOG(***catname***)]**

DEFINE can be abbreviated: DEF

Restriction: If IMBED, KEYRANGE, ORDERED, or REPLICATE is specified, it is ignored.

DEFINE ALTERNATEINDEX parameters

Required parameters:

ALTERNATEINDEX
Defines an alternate index or recatalogs an alternate index entry.

The ALTERNATEINDEX keyword is followed by the parameters for the alternate index as a whole. These parameters are enclosed in parentheses and, optionally, are followed by parameters given separately for the DATA and INDEX components.

Abbreviation: AIX

NAME(*entryname***)**
Is the alternate index's entryname or the name of each of its components. The entry name specified for the alternate index as a whole is not propagated to the alternate index's components.

You can define a separate entry name for the alternate index, its data component, and its index component. If you do not give a name for the data or index component, one is generated. For more information about the system-generated name format, see *z/OS DFSMS Managing Catalogs*.

When the alternate index, data component, and index component are individually named, each can be addressed.

RELATE(*entryname***)**
Names the alternate index base cluster. The base cluster is an entry-sequenced cluster or a key-sequenced cluster to which the alternate index is to be related. You cannot relate an alternate index to a reusable cluster, to a relative record cluster, to an extended addressability ESDS, or to a VVDS (data set name 'SYS1.VVDS.Vvolser'). An SMS-managed alternate index has the same management class and storage class as its base cluster.

Select the *entryname* so that the multilevel alias facility selects the same catalog as the one containing the related data set name.

Abbreviation: REL

CYLINDERS(*primary***[** *secondary***])|**

KILOBYTES(*primary***[** *secondary***])|**

MEGABYTES(*primary***[** *secondary***])|**

RECORDS(*primary***[** *secondary***])|**

TRACKS(*primary***[** *secondary***])**
Is the amount of space in cylinders, kilobytes, megabytes, records, or tracks allocated to the alternate index from the volume's available space. A kilobyte and megabyte allocation resolves to either tracks or cylinders; records are allocated to the nearest track boundary.

Exception: If allocation resolves to tracks, the space is contiguous. For more information, see *z/OS DFSMS Using Data Sets*.

Requests for space are directed to DADSM and result in a format-1 DSCB for the data and index component entries.

If you do not use the MODEL parameter or the RECATALOG parameter, you must include one, and only one, of these parameters: CYLINDERS, KILOBYTES, MEGABYTES, RECORDS, or TRACKS.

The space parameter is optional if the cluster is SMS-managed, but if you do not use it, space can be modeled or defaulted by SMS. If it is not determined, the DEFINE is unsuccessful.

To maintain device independence, do not use the TRACKS or CYLINDERS parameters. If you do not use TRACKS or CYLINDERS for an SMS-managed alternate index, space is allocated on the volume selected by SMS.

When you do not divide the data component into key ranges, and more than one volume is given, the primary amount of space is allocated only on the first volume when the component is defined. When the component increases to extend to additional volumes, the first allocation on each overflow volume is the primary amount.

Secondary amounts can be allocated on all volumes available to contain parts of the alternate index, regardless of the key ranges when the alternate index is extended.

You can include the amount of space as a parameter of ALTERNATEINDEX, as a parameter of DATA, or as a parameter of both DATA and INDEX:

- If the space is specified as a parameter of ALTERNATEINDEX, the amount specified is divided between the data and index components. The division algorithm is a function of control interval size, record size, device type, and other data set attributes.

  If the division results in an allocation for the data component that is not an integral multiple of the required control area size, the data component's allocation is rounded up to the next higher control area multiple. This rounding can result in a larger total allocation for your alternate index than what you specified.
- If the space is specified as a parameter of DATA, the entire amount given is allocated to the data component. An additional amount of space, depending on control interval size, record size, device type, and other data set attributes, is allocated to the index component.

To determine the exact amount of space allocated to each component, list the alternate index's catalog entry, using the LISTCAT command.

The primary and each secondary allocation must be able to be satisfied within five extents; otherwise, your DEFINE or data set extension is unsuccessful.

You can use these keywords for both SMS-managed and non-SMS-managed data sets.

primary
Allocates the initial amount of space to the alternate index.

secondary
Allocates the amount of space each time the alternate index extends, as a secondary extent. If the secondary space allocation is greater than 4.0 gigabytes, it is reduced to an amount as close to 4.0 GB as possible, without going over. This is not true for extended addressability data sets, which have no such space limitation. When you use secondary, space for the alternate index's data and index components can be expanded to a maximum of 123 extents.

Abbreviations: CYL, KB, MB, REC, and TRK

VOLUMES(*volser***[** *volser***...])**
Specifies the volumes on which an alternate index's components are to have space. This parameter is not required if the cluster is modeled or if the cluster is SMS-managed. You can specify VOLUMES for SMS-managed data sets; however, the volumes specified might not be used and, in some cases, can result in an error.

For SMS-managed data sets, you can use up to 59 volumes. If the combined number of volumes for a cluster and its associated alternate indexes exceeds 59, unpredictable results can occur.

You can let SMS choose the volumes for SMS-managed data sets by coding an * for the volser with the VOLUMES parameter. If both user-specified and SMS-specified volumes are requested, the user-specified volser must be input first in the command syntax. The default is one volume.

If you do not use the MODEL parameter, VOLUMES must be placed as a parameter of ALTERNATEINDEX, or as a parameter of both DATA and INDEX.

If the data and index components are to reside on different device types, you must include VOLUMES as a parameter of both DATA and INDEX. If more than one volume is listed with a single VOLUMES parameter, the volumes must be the same device type.

You can repeat a volume serial number in the list only if you use the KEYRANGE parameter. This can place more than one key range on the same volume. However, repetition is valid only if all duplicate occurrences are used for the primary allocation of some key range.

The VOLUMES parameter interacts with other DEFINE ALTERNATEINDEX parameters. Ensure that the volumes you define for the alternate index are consistent with the alternate index's other attributes:

- CYLINDERS, RECORDS, TRACKS: The volumes contain enough available space to satisfy the component's primary space requirement.
- FILE: To define an alternate index, the volume information supplied with the DD statement pointed to by FILE must be consistent with the information listed for the alternate index and its components.

Abbreviation: VOL

Optional parameters: The DEFINE ALTERNATEINDEX command has the following optional parameters.

BUFFERSPACE(*size*)

Provides the minimum space for buffers. VSAM determines the data component's and index component's control interval size. If you do not use BUFFERSPACE, VSAM provides enough space to contain two data component control intervals and, if the data is key-sequenced, one index component control interval.

size

Is the amount of buffer space. You can use decimal (n), hexadecimal (X'n'), or binary (B'n'), not to exceed 16,776,704. The size cannot be less than enough space to contain two data component control intervals and if the data is key sequenced, one index control interval.

If the buffer size is less than VSAM requires to run your job, the size is set to the default value, as though the parameter were not specified.

Exception: When you use VSAM RLS or DFSMStvs access, DFSMS ignores BUFFERSPACE.

Abbreviations: BUFSP or BUFSPC

CATALOG(*catname*)

Identifies the catalog in which the alternate index is defined. The catalog also contains the base cluster's entry (see the description of the RELATE parameter in preceding text).

Before you can assign catalog names for SMS-managed data sets, you must have access to the RACF STGADMIN.IGG.DIRCAT facility class. For the order in which a catalog is selected if the catalog's name is not specified and more information, see *z/OS DFSMS Access Method Services Commands*.

catname

Names the catalog.

If the catalog's volume is physically mounted, it is dynamically allocated. Mount the volume as permanently resident or reserved.

Abbreviation: CAT

CONTROLINTERVALSIZE(*size*)
Defines the size of the alternate index's control intervals. This depends on the maximum size of data records, and on the amount of buffer space given.

LSR/GSR buffering technique users can ensure buffer pool selection by explicitly defining data and index control interval sizes.

When you do not specify the control interval size, VSAM determines it. If you have not specified BUFFERSPACE and the size of your records permits, VSAM selects the optimum size for the data control interval size and 512 bytes for the index control interval size.

size
Is the size of the alternate index's data and index components.

Because an alternate index always has the spanned attribute, the control interval size can be less than the maximum record length. You can define a size from 512, to 8K in increments of 512 or from 8K to 32K in increments of 2K (where K is 1024 in decimal notation). If you use a size that is not a multiple of 512 or 2048, VSAM chooses the next higher multiple.

The index control interval should be large enough to accommodate all of the compressed keys in a data control area. If the index control interval size is too small, unnecessary control area splits can occur. After the first define (DEFINE), a catalog listing (LISTC) shows the number of control intervals in a control area and the key length of the data set. To make a general estimate of the index control interval size needed, multiply one-half of the key length (KEYLEN) by the number of data control intervals per control area (DATA CI/CA):

(KEYLEN/2) * DATA CI/CA ≤ INDEX CISIZE

For information about the relationship between control interval size and physical block size, see *z/OS DFSMS Using Data Sets*. This document also includes restrictions that apply to control interval size and physical block size.

Abbreviations: CISZ or CNVSZ

DATACLASS(*class*)
Is the 1- to 8-character name of the data class for the data set. The DATACLASS parameter provides the allocation attributes for new data sets. Your storage administrator defines the data class. However, you can override the parameters defined for DATACLASS by explicitly defining other attributes. For the order of precedence (filtering) that the system uses to select which attribute to assign, see *z/OS DFSMS Access Method Services Commands*. The record organization attribute of DATACLASS is not used for DEFINE ALTERNATEINDEX.

DATACLASS parameters apply to both SMS-managed and non-SMS-managed data sets. If DATACLASS is used and SMS is inactive, the DEFINE is unsuccessful.

You cannot use DATACLASS as a subparameter of DATA or INDEX.

Abbreviation: DATACLAS

ERASE|NOERASE
indicates if the records of the alternate index components are erased when the alternate index is deleted.

ERASE
Requires the records of the alternate index components are overwritten with binary zeros when the alternate index is deleted. If the base cluster of the alternate index is protected by a RACF generic or discrete profile and the base cluster is cataloged in an integrated catalog facility catalog, you can use RACF commands to specify an ERASE attribute as part of this profile so that the component is automatically erased upon deletion.

Abbreviation: ERAS

NOERASE
Specifies that the records of the alternate index components are not to be overwritten with binary zeros. NOERASE prevents the component from being erased if the base cluster of the alternate index is protected by a RACF generic or discrete profile that specifies the ERASE attribute and if the base cluster is cataloged in an integrated catalog facility catalog. You can use RACF commands to alter the ERASE attribute in a profile.

Abbreviation: NERAS

EXCEPTIONEXIT(*entrypoint*)
Is the name of your exception exit routine, that receives control when an exceptional I/O error condition occurs during the transfer of data between your program's address space and the alternate index's direct access storage space. (An exception is any condition that causes a SYNAD exit to be taken.) The component's exception exit routine is processed first; then SYNAD exit routine receives control. If an exception exit routine is loaded from an unauthorized library during access method services processing, an abnormal termination occurs.

Abbreviation: EEXT

FILE(*ddname*)
Names the DD statement that identifies the direct access devices and volumes on which to allocate space to the alternate index. If more than one volume is specified in a volume list, all volumes must be the same device type.

When the data component and index component are to reside on different devices, you can create a separate FILE parameter as a parameter of DATA and INDEX to point to different DD statements.

If the FILE parameter is not used, an attempt is made to dynamically allocate the required volumes. The volumes must be mounted as permanently resident or reserved.

The DD statement you specify must be:

```
//ddname DD UNIT=(devtype[,unitcount]),
//  VOL=SER=(volser1,volser2,volser3,...),DISP=OLD
```

Restriction: When FILE refers to more than one volume of the same device type, the DD statement that describes the volumes cannot be a concatenated DD statement.

FREESPACE(*CI-percent*[*CA-percent*]|0 0)
Designates the amount of empty space left after any primary or secondary allocation and any split of control intervals *(CI-percent)* and control areas *(CA-percent)* when the alternate index is built (see *z/OS DFSMS Access Method*

Services Commands). The empty space in the control interval and control area is available for data records that are updated and inserted after the alternate index is initially built. The amounts are specified as percentages. *CI-percent* translates into a number of bytes that is either equal to, or slightly less than, the percentage value of *CI-percent*. *CA-percent* translates into a number of control intervals that is either equal to, or less than, the percentage of *CA-percent*.

The percentages must be equal to, or less than, 100. When you use 100% of free space, one data record is placed in the first control interval of each control area when the alternate index is built.

Abbreviation: FSPC

IMBED|NOIMBED
The IMBED | NOIMBED parameter is no longer supported. If you specify this parameter, VSAM ignores it, and no message is issued.

KEYRANGES((*lowkey highkey***)[(***lowkey highkey***)...])**
The KEYRANGES parameter is no longer supported. If you specify this parameter, VSAM ignores it, and no message is issued.

KEYS(*length offset***|64 0)**
Describes the alternate-key field in the base cluster's data record.

The key field of an alternate index is called an alternate key. The data record's alternate key can overlap or be contained entirely within another (alternate or prime) key field.

The length plus offset cannot be greater than the length of the base cluster's data record.

When the base cluster's data record spans control intervals, the record's alternate-key field is within the record's first segment (that is, in the first control interval).

length offset
Gives the length of the alternate key, in bytes (between 1 and 255), and its displacement from the beginning of the base cluster's data record, in bytes.

MODEL(*entryname***[&cont; ***catname***])**
Uses existing entry as a model for the entry being defined or recataloged.

DATACLASS, MANAGEMENTCLASS, and STORAGECLASS cannot be modeled. For information about how the system selects modeled attributes, see *z/OS DFSMS Access Method Services Commands*.

You can use an existing alternate index's entry as a model for the attributes of the alternate index being defined. For details about how a model is used, see *z/OS DFSMS Managing Catalogs*.

You can use some attributes of the model and override others by defining them in the cluster or component. If you do not want to add or change any attributes, use only the entry type of the model (alternate index, data, or index) and the name of the entry to be defined.

When you use an alternate index entry as a model for an alternate index, the model entry's data and index components are used as models for the to-be-defined entry's data and index components, unless another entry is specified with the MODEL parameter as a subparameter of DATA or INDEX.

entryname
Names the entry to be used as a model.

catname
Names the model entry's catalog. You must identify the catalog that contains the model entry when:

- The model entry's catalog is not identified with a JOBCAT or STEPCAT DD statement, and is not the master catalog.

If the catalog's volume is physically mounted, it is dynamically allocated. The volume must be mounted as permanently resident or reserved. See *z/OS DFSMS Access Method Services Commands* for information about the order in which a catalog is selected when the catalog's name is not specified.

ORDERED|UNORDERED
The ORDERED|UNORDERED parameter is no longer supported. If you specify this parameter, VSAM ignores it, and no message is issued.

OWNER(*ownerid***)**
Gives the identification of the alternate index's owner.

For TSO/E users, if the OWNER parameter does not identify the owner, the TSO/E user ID becomes the *ownerid* value.

RECATALOG|NORECATALOG
Specifies whether the catalog entries for the alternate index components are re-created from information in the VVDS.

RECATALOG
Recreates the catalog entries if valid VVDS entries are found on the primary VVDS volume. If not, the command ends.

Use of RECATALOG requires that the NAME, RELATE, and VOLUMES parameters be specified as they were when the alternate index was originally defined. If you use RECATALOG, you are not required to include CYLINDERS, RECORDS, or TRACKS.

If ATTEMPTS, AUTHORIZATION, CATALOG, CODE, FOR, MODEL, NOUPGRADE, OWNER, or TO parameters were used during the original define, they must be entered again with RECATALOG to restore their original values; otherwise, their default values are used.

Abbreviation: RCTLG

NORECATALOG
Specifies that the catalog entries are not to be re-created from VVDS entries. Catalog entries are created for the first time.

Abbreviation: NRCTLG

RECORDSIZE(*average maximum***|4086 32600)**
Is the average and maximum length, in bytes, of an alternate index record.

An alternate index record can span control intervals, so RECORDSIZE can be larger than CONTROLINTERVALSIZE. The formula for the maximum record size of spanned records as calculated by VSAM is:

```
MAXLRECL = CI/CA * (CISZ - 10)
```

where:

- MAXLRECL is the maximum spanned record size
- CI/CA represents the number of control intervals per control area
- CA is the number of control areas
- CISZ is the quantity control interval size

You can use either of the following formulas to determine the size of the alternate-index record.

- When the alternate index supports a key-sequenced base cluster, use this formula:

  ```
  RECSZ = 5 + AIXKL + (n x BCKL)
  ```

- When the alternate index supports an entry-sequenced base cluster, use this formula:

  ```
  RECSZ = 5 + AIXKL + (n x 4)
  ```

Variables in the formulas represent these values:

- RECSZ is the average record size.
- AIXKL is the alternate-key length (see the KEYS parameter).
- BCKL is the base cluster's prime-key length. (You can enter the LISTCAT command to determine this prime-key length.)
- n = 1 when UNIQUEKEY is specified (RECSZ is also the maximum record size).
- n = the number of data records in the base cluster that contain the same alternate-key value, when NONUNIQUEKEY is specified.

When you use NONUNIQUEKEY, give a record size large enough to allow for as many key pointers or RBA pointers as you might need. The record length values apply only to the alternate index's data component.

Note: REPRO and EXPORT do not support data sets with record sizes greater than 32760.

Abbreviation: RECSZ

REPLICATE|NOREPLICATE
The REPLICATE|NOREPLICATE parameter is no longer supported. If you specify this parameter, VSAM ignores it, and no message is issued.

REUSE|NOREUSE
Indicates whether or not the alternate index can be used again as a new alternate index.

REUSE
Indicates that the alternate index can be used over again as a new alternate index. When a reusable alternate index is opened, its high-used RBA can be set to zero. Open it with an access control block using the RESET attribute.

When you use BLDINDEX to build a reusable alternate index, the high-used RBA is always reset to zero when the alternate index is opened for BLDINDEX processing.

Reusable alternate indexes can be multivolumed and might have up to 123 physical extents.

Exception: If you use the keyword **UNIQUE** with **REUSE**, the DEFINE command is unsuccessful.

Abbreviation: RUS

NOREUSE
Specifies that the alternate index cannot be used again as a new alternate index.

Abbreviation: NRUS

SHAREOPTIONS(*crossregion***[*crosssystem*]|1 3)**

Specifies how an alternate index's data or index component can be shared among users. However, SMS-managed volumes, and catalogs containing SMS-managed data sets, must not be shared with non-SMS systems. For data integrity, ensure that share options defined for data and index components are the same. For a description of data set sharing, see *z/OS DFSMS Using Data Sets*.

crossregion

Indicates the amount of sharing allowed among regions within the same system or within multiple systems using global resource serialization (GRS). Independent job steps in an operating system, or multiple systems in a GRS ring, can access a VSAM data set concurrently. For more information about GRS, see *z/OS DFSMS Using Data Sets*. To share a data set, each user must include DISP=SHR in the data set's DD statement. You can use the following options:

OPT 1 The data set can be shared by any number of users for read processing, or the data set can be accessed by only one user for read and write processing. This setting does not allow any non-RLS access when the data set is already open for VSAM RLS or DFSMStvs processing. A VSAM RLS or DFSMStvs open fails with this option if the data set is already open for any processing.

OPT 2 The data set can be accessed by any number of users for read processing, and it can also be accessed by one user for write processing. It is the user's responsibility to provide read integrity. VSAM ensures write integrity by obtaining exclusive control for a control interval while it is being updated. A VSAM RLS or DFSMStvs open is not allowed while the data set is open for non-RLS output.

If the data set has already been opened for VSAM RLS or DFSMStvs processing, a non-RLS open for input is allowed; a non-RLS open for output fails.[1] If the data set is opened for input in non-RLS mode, a VSAM RLS or DFSMStvs open is allowed.

OPT 3 The data set can be fully shared by any number of users. The user is responsible for maintaining both read and write integrity for the data the program accesses. This setting does not allow any non-RLS access when the data set is already open for VSAM RLS or DFSMStvs processing. If the data set is opened for input in non-RLS mode, a VSAM RLS or DFSMStvs open is allowed.

This option is the only one applicable to a catalog.

OPT 4 The data set can be fully shared by any number of users. For each request, VSAM refreshes the buffers used for direct processing. This setting does not allow any non-RLS access when the data set is already open for VSAM RLS or DFSMStvs processing. If the data set is opened for input in non-RLS mode, a VSAM RLS or DFSMStvs open is allowed.

As in SHAREOPTIONS 3, each user is responsible for maintaining both read and write integrity for the data the program accesses.

1. You must apply APARs OW25251 and OW25252 to allow non-RLS read access to data sets already opened for VSAM RLS or DFSMStvs processing.

crosssystem

Specifies the amount of sharing allowed among systems. Job steps of two or more operating systems can gain access to the same VSAM data set regardless of the disposition specified in each step's DD statement for the data set. However, if you are using GRS across systems or JES3, the data set might not be shared depending on the disposition of the system.

To get exclusive control of the data set's volume, a task in one system issues the RESERVE macro. The level of cross-system sharing allowed by VSAM applies only in a multiple operating system environment.

The cross-system sharing options are ignored by VSAM RLS or DFSMStvs processing. The values follow:

1 Reserved.

2 Reserved.

3 Specifies that the data set can be fully shared. Each user is responsible for maintaining both read and write integrity for the data that user's program accesses. User programs that ignore write integrity guidelines can result in:

- VSAM program checks
- Uncorrectable data set errors
- Unpredictable results

The RESERVE and DEQ macros are required with this option to maintain data set integrity. (See *z/OS MVS Programming: Authorized Assembler Services Reference ALE-DYN* and *z/OS MVS Programming: Authorized Assembler Services Reference LLA-SDU* for information on using RESERVE and DEQ.) If the sphere is accessed using VSAM RLS or DFSMStvs protocols, VSAM RLS or DFSMStvs maintains the required integrity.

4 Specifies that the data set can be fully shared. For each request, VSAM refreshes the buffers used for direct processing. This option requires that you use the RESERVE and DEQ macros to maintain data integrity while sharing the data set. Improper use of the RESERVE macro can cause problems similar to those described under SHAREOPTIONS 3. (See *z/OS MVS Programming: Authorized Assembler Services Reference ALE-DYN* and *z/OS MVS Programming: Authorized Assembler Services Reference LLA-SDU* for information on using RESERVE and DEQ.) Output processing is limited to update, or add processing, or both that does not change either the high-used RBA or the RBA of the high key data control interval if DISP=SHR is used.

To ensure data integrity in a shared environment, VSAM provides users of SHAREOPTIONS 4 (cross-region and cross-system) with the following assistance:

- Each PUT writes the appropriate buffer immediately into the VSAM object's DASD. VSAM writes out the buffer in the user's address space that contains the new or updated data record.
- Each GET refreshes the user's input buffers. The contents of each data and index buffer used by the user's program is retrieved from the VSAM object's DASD.

Exception: If you use VSAM RLS or DFSMStvs, SHAREOPTIONS is assumed to be (3,3). If you do not use VSAM RLS or DFSMStvs, the SHAREOPTIONS specification is respected.

Abbreviation: SHR

SPEED|RECOVERY
Specifies whether or not the data component's control areas are preformatted before alternate index records are loaded into them.

This parameter is only considered during the actual loading (creation) of a data set, when the data set is opened and the high-used RBA is equal to zero. After normal CLOSE processing at the completion of the load operation, the physical structure of the data set and the content of the data set extents are exactly the same. Any processing of the data set following that successful load operation is the same and the specification of this parameter is not considered.

If you use RECOVERY, the initial load takes longer because the control areas are first written with either empty or software end-of-file intervals. These preformatted CIs are then updated, using update writes, with the alternate index records. SPEED is recommended, because the initial load is quicker.

SPEED
Does not preformat the data component's space.

If the initial load is unsuccessful, you must load the alternate index records again from the beginning, because VSAM cannot determine the location of your last correctly written record. VSAM cannot find a valid end-of-file indicator when it searches your alternate index records.

RECOVERY
Specifies that the data component's control areas are written with records that indicate end-of-file. When an alternate index record is written into a control interval, it is always followed by a record that identifies the record just written as the last record in the alternate index.

RECOVERY is a way to verify storage used on the device for each CA before the data is actually written.

Abbreviation: RCVY

TO(*date*)|FOR(*days*)
Is the retention period for the alternate index. The alternate index is not automatically deleted when the expiration date is reached. When you do not provide a retention period, the alternate index can be deleted at any time. The MANAGEMENTCLASS maximum retention period, if used, limits the retention period named by this parameter.

For non-SMS-managed data sets, the correct retention period is reflected in the catalog entry. The VTOC entry might not have the correct retention period. Enter a LISTCAT command to see the correct expiration date.

For SMS-managed data sets, the expiration date in the catalog is updated and the expiration date in the format-1 DSCB is changed. Should the expiration date in the catalog not agree with the expiration date in the VTOC, the VTOC entry overrides the catalog entry. In this case, enter a LISTVTOC command to see the correct expiration date.

TO(*date*)
Specifies the earliest date that alternate index can be deleted. The date is specified in the form [yy]yyddd, where yyyy is a four-digit year, yy is a two-digit year, and ddd is the three-digit (001 through 366) day of

the year. Two-digit years are treated as if "19" is specified as the first two digits of yyyy. The dates (19)99365 and (19)99366 are considered never-expire dates.

For expiration dates of January 1, 2000, and later, you must use the form TO(yyyyddd).

The maximum yyyy value is a four-digit year (through 2155), and the maximum ddd value is a three-digit day from 000 through 365 for non-leap years. For leap years, the maximum ddd value is 366.

FOR(*days***)**
Is the number of days to keep the alternate index before it is deleted. The maximum number is 9999. If the number is 0 through 9998, the alternate index is retained for that number of days; if the number is 9999, the alternate index is retained indefinitely.

UNIQUEKEY|NONUNIQUEKEY
Shows whether more than one data record (in the base cluster) can contain the same key value for the alternate index.

UNIQUEKEY
Points each alternate index key to only one data record. When the alternate index is built (see *z/OS DFSMS Access Method Services Commands*) and more than one data record contains the same key value for the alternate index, the BLDINDEX processing ends with an error message.

Abbreviation: UNQK

NONUNIQUEKEY
Points a key value for the alternate index to more than one data record in the base cluster. The alternate index's key record points to a maximum of 32768 records with non-unique keys.

When you include NONUNIQUEKEY, the maximum record size should be large enough to allow for alternate index records that point to more than one data record.

Abbreviations: NUNQK

UPGRADE|NOUPGRADE
Specifies whether or not the alternate index is to be upgraded (that is, kept up to date) when its base cluster is modified.

UPGRADE
Upgrades the cluster's alternate index to reflect changed data when the base cluster's records are added to, updated, or erased.

When UPGRADE is specified, the alternate index's name is cataloged with the names of other alternate indexes for the base cluster. The group of alternate index names identifies the upgrade set that includes all the base cluster's alternate indexes that are opened when the base cluster is opened for write operations.

The UPGRADE attribute is not effective for the alternate index until the alternate index is built (see *z/OS DFSMS Access Method Services Commands*). If the alternate index is defined when the base cluster is open, the UPGRADE attribute takes effect the next time the base cluster is opened.

Abbreviation: UPG

NOUPGRADE
Specifies that the alternate index does not upgrade when its base cluster is modified.

Abbreviation: NUPG

WRITECHECK|NOWRITECHECK
Determines whether an alternate index or component is checked by a machine action called write-check when a record is written into it.

WRITECHECK
Indicates that a record is written and then read, without data transfer, to test for the data check condition.

Exception: When you use VSAM RLS or DFSMStvs access, the WRITECHECK parameter is ignored.

Abbreviation: WCK

NOWRITECHECK
Does not write-check the alternate index or component.

Abbreviation: NWCK

Data and index components of an alternate index

Attributes can be specified separately for the alternate index's data and index components. There is a list of the DATA and INDEX parameters at the beginning of this topic. These are described in detail as parameters of the alternate index as a whole. Restrictions are noted with each description.

DEFINE ALTERNATEINDEX examples

Define an alternate index using SMS data class specification: Example 1: In this example, an SMS-managed alternate index is defined. Because a data class is specified and no overriding attributes are explicitly specified, this define will be unsuccessful if SMS is inactive.

```
//DEFAIX   JOB  ...
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *
      DEFINE ALTERNATEINDEX -
           (NAME(EXMP1.AIX) -
           RELATE(EXAMPLE.SMS1) -
           DATACLAS(VSALLOC) -
           NONUNIQUEKEY -
           UPGRADE)
/*
```

The DEFINE ALTERNATEINDEX command creates an alternate index entry, a data entry, and an index entry to define the alternate index EXMP1.AIX. The parameters are:

- NAME indicates that the alternate index's name is EXMP1.AIX.
- RELATE identifies the alternate index base cluster, EXAMPLE.SMS1. Because an SMS-managed alternate index is being defined, the base cluster must also be SMS-managed.
- DATACLAS is an installation-defined name of an SMS data class. The data set assumes the RECORG or RECFM, LRECL, KEYLEN, KEYOFF, AVGREC, SPACE, EXPDT or RETPD, VOLUME, CISIZE, FREESPACE, and SHAREOPTIONS

parameters assigned to this data class by the ACS routines. This parameter is optional. If it is not used, the data set will assume the data class default assigned by the ACS routines.

- NONUNIQUEKEY specifies that the alternate key value might be the same for two or more data records in the base cluster.
- UPGRADE specifies that the alternate index is to be opened by VSAM and upgraded each time the base cluster is opened for processing.

Define an SMS-managed alternate index: Example 2: In this example, an SMS-managed alternate index is defined. Data class is not used, and explicitly defined attributes override any attributes in the default data class.

```
//DEFAIX   JOB  ...
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *
      DEFINE ALTERNATEINDEX -
           (NAME(EXMP2.AIX) -
           RELATE(EXAMPLE.SMS2) -
           KEYS(3 0) -
           RECORDSIZE(40 50) -
           KILOBYTES(1600 200) -
           NONUNIQUEKEY -
           UPGRADE)
/*
```

The DEFINE ALTERNATEINDEX command creates an alternate index entry, a data entry, and an index entry to define the alternate index EXMP2.AIX. The command's parameters are:

- NAME indicates that the alternate index's name is EXMP2.AIX.
- RELATE identifies the alternate index base cluster, EXAMPLE.SMS2. Because an SMS-managed alternate index is being defined, the base cluster must also be SMS-managed.
- KEYS specifies the length and location of the alternate key field in each of the base cluster's data records. The alternate key field is the first three bytes of each data record.
- RECORDSIZE specifies that the alternate index's records are variable length, with an average size of 40 bytes and a maximum size of 50 bytes.
- KILOBYTES allocates the minimum number of tracks required to contain 1600 kilobytes for the alternate index's space. When the alternate index is extended, it is to be extended by the minimum number of tracks required to contain 200 kilobytes.
- NONUNIQUEKEY means the alternate key value might be the same for two or more data records in the base cluster.
- UPGRADE opens the alternate index by VSAM and upgrades it each time the base cluster is opened for processing.

Define an alternate index: Example 3: In this example, an alternate index is defined. An example for DEFINE CLUSTER illustrates the definition of the alternate index's base cluster, EXAMPLE.KSDS2. A subsequent example illustrates the definition of a path, EXAMPLE.PATH, that lets you process the base cluster's data records using the alternate key to locate them. The alternate index, path, and base cluster are defined in the same catalog, USERCAT.

```
//DEFAIX1  JOB   ...
//STEP1    EXEC  PGM=IDCAMS
//SYSPRINT DD    SYSOUT=A
//SYSIN    DD    *
```

```
     DEFINE ALTERNATEINDEX -
           (NAME(EXAMPLE.AIX) -
           RELATE(EXAMPLE.KSDS2) -
           KEYS(3 0) -
           RECORDSIZE(40 50) -
           VOLUMES(VSER01) -
           CYLINDERS(3 1) -
           NONUNIQUEKEY -
           UPGRADE) -
         CATALOG(USERCAT)
/*
```

The DEFINE ALTERNATEINDEX command creates an alternate index entry, a data entry, and an index entry to define the alternate index EXAMPLE.AIX. The DEFINE ALTERNATEINDEX command also obtains space for the alternate index from one of the VSAM data spaces on volume VSER01, and allocates three cylinders for the alternate index's use. The parameters are:

- NAME indicates that the alternate index's name is EXAMPLE.AIX.
- RELATE identifies the alternate index's base cluster, EXAMPLE.KSDS2.
- KEYS identifies the length and location of the alternate key field in each of the base cluster's data records. The alternate key field is the first three bytes of each data record.
- RECORDSIZE specifies that the alternate index's records are variable length, with an average size of 40 bytes and a maximum size of 50 bytes.
- VOLUMES indicates that the alternate index is to reside on volume VSER01. This example assumes that the volume is already cataloged in the user catalog, USERCAT.
- CYLINDERS allocates three cylinders for the alternate index's space. The alternate index is extended in increments of one cylinder.
- NONUNIQUEKEY specifies that the alternate key value might be the same for two or more data records in the base cluster.
- UPGRADE specifies that the alternate index is opened by VSAM and upgraded each time the base cluster is opened for processing.
- CATALOG defines the alternate index in the user catalog, USERCAT.

Define an alternate index with RECATALOG: Example 4: In this example, an alternate index is redefined into a catalog:

```
//DEFAIXR  JOB   ...
//STEP1    EXEC  PGM=IDCAMS
//SYSPRINT DD    SYSOUT=A
//SYSIN    DD    *
     DEFINE ALTERNATEINDEX -
           (NAME(DEFAIXR.AIX01) -
           RELATE(DEFKSDS.KSDS03) -
           CYLINDERS(2 1) -
           VOLUMES(333001) -
           RECATALOG) -
         CATALOG(USERCAT4)
/*
```

This DEFINE ALTERNATEINDEX command recatalogs an alternate index entry, a data entry, and an index entry to redefine the alternate index, DEFAIXR.AIX01. The VSAM volume record (VVR) entry and the corresponding VTOC entry for the alternate index must exist. Only the catalog entry is recataloged, so no space is allocated. The command's parameters are:

- NAME indicates the alternate index's name, DEFAIXR.AIX01.
- RELATE identifies the alternate index base cluster, DEFKSDS.KSDS03.

- CYLINDERS allocates two cylinders for the alternate index's space. The alternate index is extended in increments of one cylinder.
- VOLUMES places the alternate index on volume 333001. This example assumes that a VTOC entry already exists for this object.
- RECATALOG recatalogs the alternate index and uses the existing VVR entry and VTOC entry.
- CATALOG defines the alternate index in the user catalog, USERCAT4.

Defining attributes for clusters and cluster components

Using access method services, you can set up jobs to execute a sequence of commands with a single invocation of IDCAMS. Model command execution is based on the success or failure of prior commands.

Use DEFINE CLUSTER to define attributes for the cluster as a whole and for the components of the cluster. The syntax of the DEFINE CLUSTER command follows:

- **DEFINE CLUSTER** (*parameters*)
  - [**DATA**(*parameters*)]
  - [**INDEX**(*parameters*)]
  - [**CATALOG**(*subparameters*)]

The syntax of the DEFINE CLUSTER command.

Command	Parameters
DEFINE	**CLUSTER** **(NAME(***entryname***)** **{CYLINDERS(***primary***[** *secondary***])** | **KILOBYTES(***primary***[** *secondary***])** | **MEGABYTES(***primary***[** *secondary***])** | **RECORDS(***primary***[** *secondary***])** | **TRACKS(***primary***[** *secondary***])}** **VOLUMES(***volser***[** *volser***...])** **[ACCOUNT(***account–info***)]** **[BUFFERSPACE(***size***)]** **[BWO(TYPECICS | TYPEIMS | NO)]** **[CONTROLINTERVALSIZE(***size***)]** **[DATACLASS(***class***)]** **[ERASE | NOERASE]** **[EXCEPTIONEXIT(***entrypoint***)]** **[FILE(***ddname***)]** **[FREESPACE(***CI-percent***[** &cont; *CA-percent***] | 0 0)]** **[FRLOG(NONE[** | &cont; **REDO])]** **[INDEXED | LINEAR | NONINDEXED | NUMBERED]** **[KEYS(***length offset* | &cont; **64 0]** **[LOG(NONE | UNDO | ALL)]** **[LOGSTREAMID(***logstream***)]** **[MANAGEMENTCLASS(***class***)]** **[MODEL(***entryname***[** *catname***])]** **[OWNER(***ownerid***)]** **[RECATALOG | NORECATALOG]** **[RECORDSIZE(***average maximum***)]** **[REUSE | NOREUSE]** **[SHAREOPTIONS(***crossregion***[** &cont; *crosssystem***] | 1 3)]** **[SPANNED | NONSPANNED]** **[SPEED | RECOVERY]**

The syntax of the DEFINE CLUSTER command.

Command	Parameters
	[STORAGECLASS(*class***)]** **[TO(***date***) | FOR(***days***)]** **[WRITECHECK | NOWRITECHECK])** **[DATA** (**{CYLINDERS(***primary***[** *secondary***]) |** **KILOBYTES(***primary***[** *secondary***]) |** **MEGABYTES(***primary***[** *secondary***]) |** **RECORDS(***primary***[** *secondary***]) |** **TRACKS(***primary***[** *secondary***])}** **[VOLUMES(***volser***[** *volser***...])]** **[BUFFERSPACE(***size***)]** **[CONTROLINTERVALSIZE(***size***)]** **[ERASE | NOERASE]** **[EXCEPTIONEXIT(***entrypoint***)]** **[FILE(***ddname***)]** **[FREESPACE(***CI-percent***[**&cont; *CA-percent***])]** **[KEYS(***length offset***)]** **[MODEL(***entryname***[** *catname***])]** **[NAME(***entryname***)]** **[OWNER(***ownerid***)]** **[RECORDSIZE(***average maximum***)]** **[REUSE | NOREUSE]** **[SHAREOPTIONS(***crossregion***[** &cont; *crosssystem***])]** **[SPANNED | NONSPANNED]** **[SPEED | RECOVERY]** **[WRITECHECK | NOWRITECHECK])]** **[INDEX** (**{CYLINDERS(***primary***[** *secondary***]) |** **KILOBYTES(***primary***[** *secondary***]) |** **MEGABYTES(***primary***[** *secondary***]) |** **RECORDS(***primary***[** *secondary***]) |** **TRACKS(***primary***[** *secondary***])}** **[VOLUMES(***volser***[** *volser***...])]** **[CONTROLINTERVALSIZE(***size***)]** **[EXCEPTIONEXIT(***entrypoint***)]** **[FILE(***ddname***)]** **[MODEL(***entryname***&cont; [** *catname***])]** **[NAME(***entryname***)]** **[OWNER(***ownerid***)]** **[REUSE | NOREUSE]** **[SHAREOPTIONS(***crossregion***[**&cont; *crosssystem***])]** **[WRITECHECK | NOWRITECHECK])]** **[CATALOG(***catname***)]**

DEFINE Abbreviation: DEF

A sequence of commands commonly used in a single job step includes DELETE—DEFINE—REPRO or DELETE—DEFINE—BLDINDEX. You can specify either a DD name or a data set name with these commands. When you refer to a DD name, however, allocation occurs at job step initiation. This could result in a job failure if a command such as REPRO follows a DELETE—DEFINE sequence

that changes the location (volser) of the data set. A failure can occur with either SMS-managed data sets or non-SMS-managed data sets.

Attention: IBM does not recommend doing a delete and define for the same data set inside a single step, or even in the same job, with DFSMStvs. The delete throws up an exclusive ENQ that is not released until the job terminates. This is not a problem most of the time because the job owns the ENQ, so it has no trouble allocating the data set. If, however, the unit of recovery ended up in backout for any reason, TVS would be unable to allocate the data set, and the UR would be shunted.

To avoid potential failures with a model command sequence in your IDCAMS job, take either of the following actions:

- Specify the data set name instead of the DD name
- Use a separate job step to perform any sequence of commands that follow a DEFINE command (for example, REPRO, IMPORT, BLDINDEX, PRINT, or EXAMINE).

Recommendation: DB2 uses the access method services DEFINE CLUSTER command for STOGROUP-defined data sets. This can result in performance problems for partitioned table spaces if multiple partitions are defined on the same volume. DB2 uses software striping on partitioned table spaces to improve performance of sequential queries. The throughput is then gated by the data delivery capability of each volume. Because each partition is a separate data set, you can be avoid this problem by allocating all the partitions in a single JCL step in an IEFBR14 (not IDCAMS) job; for details, see *z/OS DFSMS Using Data Sets*. Allocating all the partitions in this manner works if enough volumes are available with the requested space quantity in a single SMS storage group to satisfy all the partitions.

Restriction: If you specify IMBED, KEYRANGES, ORDERED, or REPLICATE, it will be ignored.

DEFINE CLUSTER parameters

The DEFINE CLUSTER command uses the following required and optional parameters.

Required parameters:

CLUSTER
Defines or recatalogs a cluster or cluster entry.

Parameters that follow the CLUSTER keyword are specified for the cluster as a whole. These parameters are enclosed in parentheses and, optionally, are followed by parameters given separately for the DATA and INDEX components.

Abbreviation: CL

NAME(*entryname*)
Defines the cluster's entryname or the name of each of its components. The entryname used for the cluster as a whole is not propagated to the cluster's components.

For SMS and non-SMS-managed clusters, the component names must resolve to the same catalog as the data set's cluster name.

You can define a separate entryname for the cluster, its data component, and its index component. If no name is specified for the data or index component,

a name is generated. When the cluster, data component, and index component are individually named, each can be addressed. For information on system-generated names, see *z/OS DFSMS Using Data Sets*.

When you define a VSAM volume data set (VVDS), the entryname for the cluster or the data component must be in the form SYS1.VVDS.Vvolser, in which volume serial number is the volume serial number specified by the VOLUMES parameter. The default primary and secondary allocation is 10 tracks. For information on defining a VVDS, see *z/OS DFSMS Managing Catalogs*.

CYLINDERS(*primary***[** *secondary***])|**

KILOBYTES(*primary***[** *secondary***])|**

MEGABYTES(*primary***[** *secondary***])|**

RECORDS(*primary***[** *secondary***])|**

TRACKS(*primary***[** *secondary***])|**

Specifies the amount of space in cylinders, kilobytes, megabytes, records, or tracks allocated to the cluster from the volume's available space. A kilobyte or megabyte allocation resolves to either tracks or cylinders; record allocation resolves to tracks. If allocation resolves to tracks, the space is contiguous. For more information, see *z/OS DFSMS Using Data Sets*.

Requests for space are directed to DADSM and result in a format-1 DSCB for all entries.

If the cluster is not SMS-managed, you must use the amount of space allocated, either through this parameter, or through the DATACLASS, MODEL, or RECATALOG parameters. This parameter is optional if the cluster is managed by SMS. If it is used, it overrides the DATACLASS space specification. If it is not used, it can be modeled or defaulted by SMS. If it cannot be determined, the DEFINE is unsuccessful.

If you select KILOBYTES or MEGABYTES, the amount of space allocated is the minimum number of tracks or cylinders required to contain the specified number of kilobytes or megabytes.

If you select RECORDS, the amount of space allocated is the minimum number of tracks that are required to contain the given number of records. The maximum number of records is 16,777,215. If RECORDS is specified for a linear data set, space is allocated with the number of control intervals equal to the number of records.

To maintain device independence, do not use the TRACKS or CYLINDERS parameters. If you use them for an SMS-managed data set, space is allocated on the volumes selected by SMS in units equivalent to the device default geometry. If there is an allocation failure due to lack of space, SMS retries allocation with a reduced space quantity. However, any retry, including reduced space quantity, is only attempted if Space Constraint Relief = Y is specified. SMS also removes other limitations if the data class allows space constraint relief.

Regardless of the allocation type, the calculation of the CA (control area) size is based on the smaller of the two allocation quantities (primary or secondary) in the DEFINE command. A CA is never greater than a single cylinder, it might be less (that is, some number of tracks), depending on the allocation amount and type used. When tracks or records are used, the space allocation unit (the CA size) can be adjusted to one cylinder. This adjustment is made if the

calculated CA size contains more tracks than exist in a single cylinder of the device being used. The CA area size assigned by VSAM is the smallest of:

- One cylinder
- The primary space quantity
- The secondary space quantity

If the CA size assigned is not evenly divisible into either the primary or secondary space quantity, VSAM increases that space to a value evenly divisible by the CA size. If you are defining an extended format data set, you should review "Defining an Extended Format Key-Sequenced Data Set" in z/OS DFSMS Using Data Sets for information about additional space requirements.

DEFINE RECORDS allocates sufficient space to the specified number of records, but factors unknown at define time (such as key compression or method of loading records) can result in inefficient use of the space allocated. This might prevent every data CA from being completely used, and you might be unable to load the specified number of records without requiring secondary allocation.

When multiple volumes are used for a data set, these rules and conditions apply:

- The first volume is defined as the prime volume. The initial allocation of a data set is on the prime volume. The remaining volumes are defined as candidate volumes.
- A data set's primary space allocation (defined for each data set) is the amount of space initially allocated on both the prime volume and on any candidate volumes the data set extends to.
- A data set's secondary space allocation (if it is defined) is the space allocated when the primary space is filled and the data set needs additional space on the same volume.
- If a data set extends to a candidate volume, the amount of space initially allocated on the candidate volume is the primary space allocation. If the data set extends beyond the primary allocation on the candidate volume, then the amount of space allocated is the secondary space allocation.
- With a DEFINE request, the primary space allocation must be fulfilled in five DASD extents unless the Space Constraint Relief option is specified in the associated SMS data class.

  However, the request is not successful if you do not fulfill each secondary space allocation in five DASD extents. A DASD extent is the allocation of one available area of contiguous space on a volume. For example, if a data set's primary space allocation is 100 cylinders, you must allocate a maximum of five DASD extents that add up to 100 cylinders.

Secondary amounts can be allocated on all volumes available to contain parts of the cluster regardless of the key ranges.

You can specify the amount of space as a parameter of CLUSTER, as a parameter of DATA, or as a parameter of both. When a key-sequenced cluster is being defined, and the space is a parameter of:

- CLUSTER, the amount is divided between the data and index components. The division algorithm is a function of control interval size, record size, device type, and other data set attributes.

  If the division results in an allocation for the data component that is not an integral multiple of the required control area size, the data component's

allocation is rounded up to the next higher control area multiple. This rounding can result in a larger total allocation for your cluster.

- DATA, the entire amount specified is allocated to the data component. An additional amount of space, depending on control interval size, record size, device type, and other data set attributes, is allocated to the index component.

To determine the exact amount of space allocated to each component, list the cluster's catalog entry, using the LISTCAT command.

The primary and each secondary allocation must be able to be satisfied in five DASD extents; otherwise, your DEFINE or data set extension is unsuccessful.

primary
Allocates the initial amount of space to the cluster.

secondary
Allocates an amount of space each time the cluster extends, as a secondary extent. You can use this secondary allocation to add space for the data or index components of the cluster. A VSAM data set can be expanded to 123 extents per volume. If this is a multi-volume VSAM data set, then the VSAM component can be extended to a maximum of 255 extents combined over all volumes.

VOLUMES(*volser***[** *volser***...])**
Specifies the volumes on which a cluster's components are to have space. If you do not use the MODEL parameter, or if the cluster is not SMS-managed, VOLUMES must be used either as a parameter of CLUSTER, or as a parameter of both DATA and INDEX.

VOLUMES can be specified or modeled for a data set that is to be SMS-managed; know that the volumes specified might not be used and result in an error. See *z/OS DFSMSdfp Storage Administration* for information about SMS volume selection.

Volumes are always allocated in the order specified. If there is not enough space on the volume, the allocation is not successful. For non-SMS-managed data sets, the primary space is allocated on the first volume in the list. When you extend the data set because the first allocation is full, the volumes are used in the order in which they appeared in the DEFINE command.

Letting SMS select the volume from the storage group reduces the chances of allocation errors caused by insufficient space. If the data set is SMS-managed with guaranteed space, SMS places the primary quantity on all the volumes with sufficient space for later extensions. If the SMS-managed data set does not have guaranteed space or is a key range data set, primary space is allocated only on the first volume. For SMS-managed VSAM data sets, the primary space might be allocated on a different volume from the one you specified.

You can let SMS choose the volumes for SMS-managed data sets by coding an * for the volser with the VOLUMES parameter. If both user-specified and SMS-specified volumes are requested, the user-specified volser must be input first in the command syntax. The default is one volume.

For SMS-managed and non-SMS-managed data sets, you can specify up to 59 volume serial numbers. If the combined number of volumes for a cluster and its associated alternate indexes exceeds 59, unpredictable results can occur.

If the data and index components are to reside on different device types, you must specify VOLUMES as a parameter of both DATA and INDEX. If more than one volume is listed with a single VOLUMES parameter, the volumes must be of the same device type.

For SMS-managed data sets, if you want the data and index components to be on separate volumes for nonguaranteed-space storage-class requests, code two different dummy names in the VOLUME parameter for each component. If there are not enough volumes in the storage group to satisfy this requirement, the allocation will fail.

If a guaranteed-space storage class is assigned to the data sets (cluster) and volume serial numbers are used, space is allocated on all specified volumes if the following conditions are met:

- All defined volumes are in the same storage group.
- The storage group to which these volumes belong is in the list of storage groups selected by the ACS routines for this allocation.
- The data set is not a key range data set.

The volume serial number is repeated in the list only if the KEYRANGE parameter is used. You can use this to have more than one key range on the same volume. Repetition is valid when duplicate occurrences are used for the primary allocation of some key range.

If a VVDS is being defined, only one volume can be specified and that volume serial number must be reflected in the name indicated in the NAME parameter.

The VOLUMES parameter interacts with other DEFINE CLUSTER parameters. Ensure that the volume you give for the cluster is consistent with the cluster's other attributes:

- FILE: The volume information supplied with the DD statement pointed to by FILE must be consistent with the information specified for the cluster and its components.

AbbreviationsCYL, KB, MB, REC, TRK

Abbreviation: VOL

Optional parameters: The DEFINE CLUSTER command can execute the following optional parameters.

ACCOUNT(*account–info***)**
Defines up to 32 bytes of accounting information and user data for the data set. It *must* be between 1 and 32 bytes; otherwise, you will receive an error message.

account–info
Is supported only for SMS-managed VSAM and non–VSAM data sets. It is used only for the data set level (not member level) of PDSE/PDS.

Abbreviation: ACCT

BUFFERSPACE(*size***)**
Specifies the minimum space for buffers. The buffer space size helps VSAM determine the data component's and index component's control interval size. If BUFFERSPACE is not coded, VSAM attempts to get enough space to contain two data component control intervals and, if the data is key sequenced, one index component control interval.

If the data set being defined is a KSDS, and the BUFFERSPACE specified is not large enough to contain two data and one index CI's, VSAM increases the

specified buffer space and completes the define. VSAM also increases index CISIZE and, if necessary, increases the buffer space to accommodate the larger index CISIZE.

size
Is the space for buffers. *Size* can be given in decimal (n), hexadecimal (X'n'), or binary (B'n') form, but must not exceed 16,776,704.

The size cannot be less than enough space to contain two data component control intervals and, if the data is key sequenced, one index control interval. If size is too small for VSAM buffers, VSAM ends your DEFINE and provides an appropriate error message.

Note: The BUFFERSPACE setting is ignored when the data set is opened for VSAM RLS or DFSMStvs mode.

Abbreviations: BUFSP or BUFSPC

BWO(TYPECICS|TYPEIMS|NO)
Use this parameter if backup-while-open (BWO) is allowed for the VSAM sphere. BWO applies only to SMS data sets and cannot be used with TYPE(LINEAR). If BWO, LOG, or LOGSTREAMID is specified (or an RLS cell exists for the data set), access from DFSMS/MVS 1.2 or a lower-level system is denied.

If BWO is specified in the SMS data class, the specified BWO value is used as part of the data set definition, unless BWO was previously defined with an explicitly specified or modeled DEFINE attribute.

TYPECICS
Use TYPECICS to specify BWO in a CICS or DFSMStvs environment. For RLS processing, this activates BWO processing for CICS or DFSMStvs, or both. For non-RLS processing, CICS determines whether to use this specification or the specification in the CICS FCT. For more information about the use of TYPECICS, see *z/OS DFSMSdfp Storage Administration* and the CICS Transaction Server for z/OS Information Center at http://publib.boulder.ibm.com/infocenter/cicsts/v3r1/index.jsp.

Exception: If CICS determines that it will use the specification in the CICS FCT, the specification might override the TYPECICS parameter for CICS processing.

Abbreviation: TYPEC

TYPEIMS
Enables BWO processing for IMS data sets. You can use this capability only with DFSMS 1.3 or higher-level DFSMS systems. If you attempt to open a cluster that has the TYPEIMS specification of a DFSMS 1.2 (or lower-level) system, the open will not be successful.

Abbreviation: TYPEI

NO Use this when BWO does not apply to the cluster.

Exception: If CICS determines that it will use the specification in the CICS FCT, the specification might override the NO parameter for CICS processing.

CATALOG(*catname*)
Identifies the catalog in which the cluster is to be defined.

Before you can specify catalog names for SMS-managed data sets, you must have authority for the RACF STGADMIN.IGG.DIRCAT facility class. For the order in which catalogs are selected and more information, see *z/OS DFSMS Access Method Services Commands*.

catname
Is the name of the catalog in which the entry is to be defined.

If the catalog's volume is physically mounted, it is dynamically allocated. The volume must be mounted as permanently resident or reserved.

Abbreviation: CAT

CONTROLINTERVALSIZE(*size***)**
Specifies the size of the control interval for the cluster or component.

For linear data sets, the specified value in bytes is rounded up to a 4K multiple, up to a maximum of 32K. If the size is not specified, the value specified in the data class that is assigned to the data set is used. Otherwise a default value of 4K is used.

If CONTROLINTERVALSIZE is given on the cluster level, it propagates to the component level at which no CONTROLINTERVALSIZE has been specified.

The size of the control interval depends on the maximum size of the data records and the amount of buffer space you provide.

LSR/GSR buffering technique users can ensure buffer pool selection by explicitly defining data and index control interval sizes.

If CONTROLINTERVALSIZE is not coded, VSAM determines the size of control intervals. VSAM selects a control interval size for the data component that will optimize direct access storage usage. It will then select an index control interval size based on the number of data control intervals in the data control area.

size
Indicates a cluster's data and index component size.

If SPANNED is not used, the size of a data control interval must be at least 7 bytes larger than the maximum record length.

If the control interval specified is less than maximum record length plus a 7-byte overhead, VSAM increases the data control interval size to contain the maximum record length plus the needed overhead.

If SPANNED is specified, the control interval size can be less than the maximum record length. You can select a size from 512 to 8K in increments of 512, or from 8K to 32K in increments of 2K. When you choose a size that is not a multiple of 512 or 2048, VSAM chooses the next higher multiple. For a linear data set, the size specified is rounded up to 4096 if specified as 4096 or less. It is rounded to the next higher multiple of 4096 if specified as greater than 4096.

The size of the index control interval is the number of data control intervals in a data control area that need indexing at the sequence set level of the index component. The size of each entry depends on an average compression value for a user key. The keys will compress to 1/3 of the length of the actual key value. In some cases, the general compressed key length on which the algorithm is based will be affected by the actual values and ordering of the user key. The result is that each entry can occupy more space in the index record than that provided. This may result in additional control area splits and in all cases, wasted space in the data set. If after loading the data sets, this

condition exists; noted by more than anticipated space to store the data set on the direct access device. You should increase the index control interval size. The size can be increased incrementally until it is felt that this condition no longer exists. The guideline formula documented in the past is as follows:

(KEYLEN/2) * DATA CI/CA less than or equal to INDEX CISIZE.

You should be aware that this is only a guideline and does not take into account the actual algorithm for determining the index control interval size requirement. However, the 2:1 compression of key length in this formula provides some additional overhead over the actual 3:1 formula used during the actual algorithm. Using this formula can result in an index control interval size that is too large. This may increase I/O transfer time for each index component record, or it may be too small to address these conditions.

For a discussion of control interval size and physical block size, see *z/OS DFSMS Using Data Sets*.

Abbreviation: CISZ or CNVSZ

DATACLASS(*class***)**
Identifies the name, 1 to 8 characters, of the data class for the data set. It provides the allocation attributes for new data sets. Your storage administrator defines the data class. However, you can override the parameters defined for DATACLASS by explicitly using other attributes. See *z/OS DFSMS Access Method Services Commands* for the order of precedence (filtering) the system uses to select which attribute to assign.

DATACLASS parameters apply to both SMS-managed and non-SMS-managed data sets. If DATACLASS is specified and SMS is inactive, DEFINE is unsuccessful.

DATACLASS cannot be used as a subparameter of DATA or INDEX.

Abbreviation: DATACLAS

ERASE|NOERASE
Specifies whether the cluster's components are to be erased when its entry in the catalog is deleted.

ERASE
Overwrites each component of the cluster with binary zeros when its catalog entry is deleted. If the cluster is protected by a RACF generic or discrete profile and is cataloged in an integrated catalog facility catalog, you can use RACF commands to specify an ERASE attribute. If you do this, the data component is automatically erased upon deletion.

Abbreviation: ERAS

NOERASE
Specifies that each component of the cluster is not to be overwritten with binary zeros. NOERASE will not prevent erasure if the cluster is protected by a RACF generic or discrete profile that specifies the ERASE attribute and if the cluster is cataloged in a catalog. Use RACF commands to alter the ERASE attribute in a profile.

Abbreviation: NERAS

EXCEPTIONEXIT(*entrypoint***)**
Specifies the name of a user-written exception-exit routine, that receives control when an exceptional I/O error condition occurs during the transfer of data between your program's address space and the cluster's DASD space. An exception is any condition that causes a SYNAD exit to be taken. The

component's exception-exit routine is processed first, then the user's SYNAD exit routine receives control. If an exception-exit routine is loaded from an unauthorized library during access method services processing, an abnormal termination occurs. See *z/OS DFSMS Using Data Sets*.

Abbreviation: EEXT

FILE(*ddname***)**
Names the DD statement that identifies and allocates the DASD and volumes that must be available for space allocation on the volumes specified by the VOLUMES keyword. If more than one volume is specified, all volumes must be the same device type.

If data and index components are to reside on separate devices, you can specify a separate FILE parameter as a parameter of DATA and INDEX to point to different DD statements.

If the FILE parameter is not specified, an attempt is made to dynamically allocate the required volumes. The volume must be mounted as permanently resident or reserved. When the FILE parameter is used, the specified volumes are directly allocated before access method services gets control.

An example DD statement is:

```
//ddname DD UNIT=(devtype[,unitcount]),
//    VOL=SER=(volser1,volser2,volser,...),DISP=OLD
```

Restriction: When FILE refers to more than one volume of the same device type, the DD statement that describes the volumes cannot be a concatenated DD statement.

FREESPACE(*CI-percent***[&cont; ***CA-percent***]|0 0)**
Specifies the percentage of each control interval and control area to be set aside as free space when the cluster is initially loaded or when a mass insert is done. CI-percent is a percentage of the amount of space to be preserved for adding new records and updating existing records with an increase in the length of the record. Since a CI is split when it becomes full, the CA might also need to be split when it is filled by CIs created by a CI split. The empty space in the control interval and control area is available for data records that are updated and inserted after the cluster is initially loaded. This parameter applies only to key-sequenced clusters, and variable-length relative records with variable-length records. *CI-percent* is the number of bytes that is equal to, or slightly less than, the percentage value of *CI-percent*. *CA-percent* is the number of control intervals equal to, or less than, the percentage of *CA-percent*.

CI-percent and *CA-percent* must be equal to, or less than, 100. When you use FREESPACE(100 100), one data record is placed in each control interval used for data. One control interval in each control area is used for data (that is, one data record is stored in each control area when the data set is loaded). If you do not use FREESPACE, the default reserves no free space when the data set is loaded.

When you define the cluster using the RECORDS parameter, the amount of free space specified is not considered in the calculations to determine primary allocation.

Abbreviation: FSPC

FRLOG(NONE|REDO)
Specifies that VSAM batch logging can be performed for your VSAM data set. VSAM batch logging is available with CICS VSAM Recovery V3R1.

There is no default value for FRLOG. If FRLOG is left out, the data set cannot be used for VSAM batch logging. See the ALTER command for enabling VSAM batch logging after a data set is created.

NONE
Indicates that the data set can be used for VSAM batch logging. However, the function should be disabled. The LOGSTREAMID parameter indicates changes that are made by applications that are written to the MVS log stream. Specifying FRLOG(NONE) implies that you can use the data set for RLS processing; omitting it indicates that RLS processing will not occur.

REDO
Enables the VSAM batch logging function for your VSAM data set. The LOGSTREAMID parameter indicates changes that are made by applications that are written to the MVS log stream. When specifying FRLOG(REDO), you must also specify LOGSTREAMID.

Restrictions:

1. If you do not want VSAM batch logging for your data set, do not specify the FRLOG parameter. If you specify FRLOG(NONE), the data set must support VSAM batch logging, but logging is not in effect.
2. If FRLOG is specified, the following restrictions apply:
   - The data set must be SMS-managed.
   - The data set cannot be a linear or temporary data set.

INDEXED|LINEAR|NONINDEXED|NUMBERED
Shows the type of data organization for the cluster.

If you want a data organization other than INDEXED (the default), you must explicitly use it with this parameter.

When a cluster is defined, you indicate whether the data is to be indexed (key sequenced), nonindexed (entry sequenced), numbered (relative record), or linear.

Certain parameters apply only to key-sequenced clusters, as noted in the description of each of these parameters.

Linear data set clusters are treated as ESDS clusters that must be processed using control interval access.

If you do not choose either the data organization or the MODEL parameter, your cluster defaults to key-sequenced (indexed).

If you want to define an entry-sequenced or a relative record cluster, you must specify the NONINDEXED, the NUMBERED, or the MODEL parameter.

The data organization you select must be consistent with other parameters you specify.

INDEXED
Shows that the cluster being defined is for key-sequenced data. If INDEXED is specified, an index component is automatically defined and cataloged. The data records can be accessed by key or by relative-byte address (RBA).

Abbreviation: IXD

LINEAR
Specifies that the cluster being defined is for linear data. Because linear data set clusters are treated as ESDS clusters that must be processed using

control interval access, you can use most of the commands and parameters you use to manipulate ESDS clusters. There are two exceptions:

- Parameters that refer to logical records are not allowed (except RECORDS).
- Use partial printing by specifying the RBA syntax.

Space is allocated for a linear data set with the number of control intervals equal to the number of records.

Restriction: Linear data sets cannot be accessed for VSAM RLS or DFSMStvs processing. The LOG, LOGSTREAMID, and BWO parameters do not apply to linear data sets.

Abbreviation: LIN

NONINDEXED
Indicates that the cluster being defined is for entry-sequenced data. The data records can be accessed sequentially or by relative-byte address (RBA).

Abbreviation: NIXD

NUMBERED
Specifies that the cluster's data organization is for relative record data. A relative record cluster, which is similar to an entry-sequenced cluster, has fixed-length records or variable-length records that are stored in slots. The RECORDSIZE parameter determines if the records are fixed-length or variable-length. Empty slots hold space for records to be added later. The data records are accessed by relative record number (slot number).

Abbreviation: NUMD

KEYS(*length offset***|64 0)**
gives information about the prime key field of a key-sequenced data set's data records.

This parameter overrides any KEYS specification in the DATACLASS parameter.

This parameter applies only to key-sequenced clusters. The default is a key field of 64 bytes, beginning at the first byte (byte 0) of each data record.

The key field of the cluster's index is called the prime key to distinguish it from other keys, called alternate keys. See “Defining alternate indexes” on page 83 for more details on how to choose alternate indexes for a cluster.

When the data record spans control intervals, the record's key field must be within the part of the record that is in the first control interval.

length offset
Specifies the length of the key and its displacement (in bytes) from the beginning of the record. The sum of length plus offset cannot exceed the length of the shortest record. The length of the key can be 1 to 255 bytes.

LOG(NONE|UNDO|ALL)
Establishes whether the sphere to be accessed with VSAM RLS or DFSMStvs is recoverable or nonrecoverable. It also indicates whether a forward recovery log is available for the sphere. LOG applies to all components in the VSAM sphere. VSAM uses LOG in the following way:

Nonrecoverable Sphere
The sphere is considered nonrecoverable if LOG(NONE) is specified. VSAM allows concurrent read and update sharing across multiple resource managers and other applications.

Recoverable Sphere
The sphere is considered recoverable if LOG(UNDO) or LOG(ALL) is specified. For a recoverable sphere, VSAM does not allow applications that do not support commit and backout to open a data set in the sphere for output using RLS access, but applications can open the sphere for output using DFSMStvs access. The applications can, however, open the sphere for RLS access for input processing only.

If the LOG parameter is not specified, the value in the catalog is undefined. RLS and DFSMStvs OPEN of the data set is not allowed unless the LOG parameter has been specified.

If BWO, LOG, or LOGSTREAMID is specified (or an RLS cell exists for the data set), access from DFSMS/MVS 1.2 or lower-level systems is denied.

If LOG is specified in the SMS data class, the value defined is used as the data set definition, unless it has been previously defined with an explicitly specified or modeled DEFINE attribute.

LOG cannot be used with LINEAR.

LOGSTREAMID cannot be used with LINEAR.

NONE
Indicates that neither an external backout nor a forward recovery capability is available for the sphere accessed in VSAM RLS or DFSMStvs mode. If you use LOG(NONE), RLS and DFSMStvs consider the sphere to be nonrecoverable.

UNDO
Specifies that changes to the sphere accessed in VSAM RLS or DFSMStvs mode can be backed out using an external log. RLS and DFSMStvs consider the sphere to be recoverable when you use LOG(UNDO).

ALL
Specifies that changes to the sphere accessed in RLS and DFSMStvs mode can be backed out and forward recovered using external logs. DFSMStvs and RLS consider the sphere recoverable when you use LOG(ALL). When you specify LOG(ALL), you must also specify the LOGSTREAMID parameter.

VSAM RLS and DFSMStvs allow concurrent read or update sharing for nonrecoverable spheres through commit (CICS) and noncommit protocol applications. For a recoverable sphere, a noncommit protocol application must use DFSMStvs to be able to open the sphere for update using RLS access.

LOGSTREAMID(*logstream*)
Gives the name of the forward recovery log stream. It applies to all components in the VSAM sphere. This parameter is used when the cluster is accessed with MACRF=RLS.

If BWO, LOG, or LOGSTREAMID is specified (or an RLS cell exists for the data set), access from DFSMS/MVS 1.2 or lower-level systems is denied.

If LOGSTREAMID is specified in the SMS data class, the value defined is used as the data set definition, unless it has been previously defined with an explicitly specified or modeled DEFINE attribute.

logstream
Is the name of the forward recovery log stream. This can be a fully qualified name up to 26 characters, including separators. If LOG(ALL) is specified, LOGSTREAMID(name) must be specified. For information about defining log streams for CICS use, see the CICS Transaction Server for z/OS Information Center at http://publib.boulder.ibm.com/infocenter/cicsts/v3r1/index.jsp.

Abbreviation: LSID

Note: LOGSTREAMID cannot be used with LINEAR.

MANAGEMENTCLASS(*class***)**
For SMS-managed data sets, specifies the name, 1 to 8 characters, of the management class for a new data set. Your storage administrator defines the names of the management classes you can use. If MANAGEMENTCLASS is not used, but STORAGECLASS is used or defaulted, MANAGEMENTCLASS is derived from automatic class selection (ACS). If MANAGEMENTCLASS is specified and STORAGECLASS is not specified **or** derived, the DEFINE is unsuccessful. If SMS is inactive and MANAGEMENTCLASS is specified, the DEFINE will be unsuccessful. MANAGEMENTCLASS cannot be listed as a subparameter of DATA or INDEX.

Abbreviation: MGMTCLAS

MODEL(*entryname***[** *catname***])**
Specifies an existing entry to be used as a model for the entry being defined. See *z/OS DFSMS Access Method Services Commands* for information on how the system selects modeled attributes.

A VVDS cannot be modeled.

The DATACLASS, MANAGEMENTCLASS, and STORAGECLASS attributes are not modeled.

You can use an existing cluster's entry as a model for the attributes of the cluster being defined. For details about how a model is used, see *z/OS DFSMS Managing Catalogs*.

You can use some attributes of the model and override others by explicitly specifying them in the definition of the cluster or component. If you do not want to add or change any attributes, you need specify only the entry type (cluster, data, or index) of the model to be used and the name of the entry to be defined.

See *z/OS DFSMS Access Method Services Commands* for more information about the order in which the system selects an attribute.

When you use a cluster entry as a model for the cluster, the data and index entries of the model cluster are used as models for the data and index components of the cluster still to be defined, unless another entry is specified with the MODEL parameter as a subparameter of DATA or INDEX.

entryname
Specifies the name of the cluster or component entry to be used as a model.

catname
Names the model entry's catalog. You identify the catalog that contains the model entry if the model entry's catalog is not identified with a JOBCAT or STEPCAT DD statement, and is not the master catalog.

If a catalog's volume is physically mounted, it is dynamically allocated. The volume must be mounted as permanently resident or reserved. For information about the order in which a catalog is selected when the catalog name is not specified, see *z/OS DFSMS Access Method Services Commands*.

FRLOG(NONE|REDO)
Option to request that VSAM interface with CICSVR to log changed data to an MVS Log Stream. The log data and a previous HSM backup can be used for forward recovery.

NONE
Specifies that the forward recovery capability is not available for the sphere.

REDO
Specifies that the forward recovery capability is available for the sphere. Batch logging is supported for NSR, LSR, and GSR and does not require modification of application programs. When you specify FRLOG(REDO), you must also specify the LOGSTREAMID parameter.

Note:

1. FRLOG cannot be used with LINEAR.
2. This support is available with CICSVR 2.4

OWNER(*ownerid*)
identifies the cluster's owner.

For TSO/E users, if the owner is not identified with the OWNER parameter, the TSO/E user's userid becomes the ownerid.

RECATALOG|NORECATALOG
Indicates whether the catalog entries for the cluster components are to be re-created from information in the VVDS.

RECATALOG
Re-creates the catalog entries if valid VVDS entries are found on the primary VVDS volume. If they are not, the command ends.

Catalog entries can be re-created only in the catalog specified in the VVR except for entries that are swap space, page space, or SYS1 data sets.

The RECORDSIZE parameter is required when doing a DEFINE RECATALOG of a variable-length relative record data set (VRRDS).

Identification of RECATALOG requires that NAME, INDEXED, LINEAR, NONINDEXED, NUMBERED, and VOLUMES be used as they were when the cluster was originally defined. If you specify RECATALOG, you are not required to use CYLINDERS, RECORDS, or TRACKS.

If the ATTEMPTS, AUTHORIZATION, CATALOG, CODE, FOR, MODEL, OWNER, or TO parameter is used during the original define, they must be respecified with RECATALOG to restore their original values; otherwise, their default values are used.

When you use the TO parameter with RECATALOG, only the cluster's expiration date is updated. The DATA and INDEX components are not updated.

If the RACF user has ADSP specified, a profile is defined to RACF for the data set being recataloged.

If the cluster was SMS-managed, the volume serials should be the same as the volumes actually selected by SMS.

The catalog for the entries being re-created must have the same name as the catalog that contained the original entries.

Abbreviation: RCTLG

NORECATALOG
Indicates that the catalog entries are not re-created from VVDS entries. Catalog entries are created for the first time.

Abbreviation: NRCTLG

RECORDSIZE(*average maximum*|*default***)**
Specifies the average and maximum lengths, in bytes, of the records in the data component. The minimum record size is 1 byte.

RECORDSIZE can be given as a parameter of either CLUSTER or DATA.

This parameter overrides the LRECL specification on the DATACLASS parameter.

For nonspanned records, the maximum record size + 7 cannot exceed the data component's control interval size (that is, the maximum nonspanned record size, 32761, + 7 equals the maximum data component control interval size, 32768).

When you use a record size that is larger than one control interval, you must also specify spanned records (SPANNED). The formula for the maximum record size of spanned records as calculated by VSAM is as follows:

```
MAXLRECL = CI/CA * (CISZ - 10)
```

where:

- MAXLRECL is the maximum spanned record size.
- CI/CA represents the number of control intervals per control area.
- CA is the number of control areas.
- CISZ is the quantity control interval size.

When you select NUMBERED, you identify a data set as a relative record data set. If you use NUMBERED and select the same value for average as for maximum, the relative records must be fixed-length. If you specify NUMBERED and select two different values for the average and maximum record sizes, the relative records can be variable-length. If you know that your relative records are fixed-length, however, be sure to define them as fixed-length. Performance is affected for relative record data sets defined as variable-length. Each variable-length relative record is increased internally in length by four.

When your records are fixed length, you can use the following formula to find a control interval size that contains a whole number (n) of records:

```
CISZ = (n x RECSZ) + 10
or
n = (CISZ - 10)
      RECSZ
```

If you select SPANNED or NUMBERED for your fixed-length records:

```
CISZ =(n x (RECSZ + 3)) + 4
or
n = (CISZ - 4)
    (RECSZ + 3)
```

Variables in the example represent these values:

- *n* is the number of fixed-length records in a control interval.
- CISZ is the control interval size (see also the CONTROLINTERVALSIZE parameter).
- RECSZ is the average record size.

default
When SPANNED is used, the default is RECORDSIZE(4086 32600). Otherwise, the default is RECORDSIZE(4089 4089).

Important:

```
REC(sec) x RECSZ(avg) > RECSZ(max)
```

- Variables in the example represent these values:
  - REC(sec) is the secondary space allocation quantity, in records.
  - RECSZ(avg) is the average record size (default = 4086 or 4089 bytes).
  - RECSZ(max) is the maximum record size (default = 4089 or 32600 bytes).

When the SPANNED record size default prevails (32600 bytes), the secondary allocation quantity should be at least 8 records.

Restriction: REPRO and EXPORT do not support data sets with record sizes greater than 32760.

Abbreviation:RECSZ

REUSE|NOREUSE
Specifies whether the cluster can be opened again and again as a reusable cluster.

If REUSE or NOREUSE is specified in the SMS data class, the value defined is used as the data set definition, unless it has been previously defined with an explicitly specified or modeled DEFINE attribute.

REUSE
Specifies that the cluster can be opened again and again as a reusable cluster. When a reusable cluster is opened, its high-used RBA is set to zero if you open it with an access control block that specifies the RESET attribute.

REUSE lets you create an entry-sequenced, key-sequenced, or relative-record work data set.

When you create a reusable cluster, you cannot build an alternate index to support it. Also, you cannot create a reusable cluster with key ranges. Reusable data sets can be multivolume and can have up to 123 physical extents.

Restriction: If you select REUSE and your command also contains the keyword **UNIQUE**, you must remove the UNIQUE keyword or the DEFINE command will be unsuccessful.

Abbreviation: RUS

NOREUSE
indicates that the cluster cannot be opened again as a new cluster.

Abbreviation: NRUS

SHAREOPTIONS(*crossregion***[&cont; ***crosssystem***]|1 3)**

Shows how a component or cluster can be shared among users. However, SMS-managed volumes, and catalogs containing SMS-managed data sets, must not be shared with non-SMS systems. For a description of data set sharing, see *z/OS DFSMS Using Data Sets*. To ensure integrity, you should be sure that share options specified at the DATA and INDEX levels are the same.

The value of SHAREOPTIONS is assumed to be (3,3) when the data set is accessed in VSAM RLS or DFSMStvs mode.

crossregion

Specifies the amount of sharing allowed among regions within the same system or within multiple systems using global resource serialization (GRS). Independent job steps in an operating system, or multiple systems in a GRS ring, can access a VSAM data set concurrently. For more information about GRS, see *z/OS MVS Planning: Global Resource Serialization*. To share a data set, each user must use DISP=SHR in the data set's DD statement. You can use the following options:

OPT 1 The data set can be shared by any number of users for read processing, or the data set can be accessed by only one user for read and write processing. VSAM ensures complete data integrity for the data set. This setting does not allow any non-RLS access when the data set is already open for VSAM RLS or DFSMStvs processing. A VSAM RLS or DFSMStvs open will fail with this option if the data set is already open for any processing.

OPT 2 The data set can be accessed by any number of users for read processing, and it can also be accessed by one user for write processing. It is the user's responsibility to provide read integrity. VSAM ensures write integrity by obtaining exclusive control for a control interval while it is being updated. A VSAM RLS or DFSMStvs open is not allowed while the data set is open for non-RLS output.

If the data set has already been opened for VSAM RLS or DFSMStvs processing, a non-RLS open for input is allowed; a non-RLS open for output fails. [2] If the data set is opened for input in non-RLS mode, a VSAM RLS or DFSMStvs open is allowed.

OPT 3 The data set can be fully shared by any number of users. Each user is responsible for maintaining both read and write integrity for the data the program accesses. This setting does not allow any non-RLS access when the data set is already open for VSAM RLS or DFSMStvs processing. If the data set is opened for input in non-RLS mode, a VSAM RLS or DFSMStvs open is allowed.

OPT 4 The data set can be fully shared by any number of users. For each request, VSAM refreshes the buffers used for direct processing. This setting does not allow any non-RLS access

2. You must apply APARs OW25251 and OW25252 to allow non-RLS read access to data sets already opened for VSAM RLS or DFSMStvs processing.

when the data set is already open for VSAM RLS or DFSMStvs processing. If the data set is opened for input in non-RLS mode, a VSAM RLS or DFSMStvs open is allowed.

As in SHAREOPTIONS 3, each user is responsible for maintaining both read and write integrity for the data the program accesses.

crosssystem
Specifies the amount of sharing allowed among systems. Job steps of two or more operating systems can gain access to the same VSAM data set regardless of the disposition indicated in each step's DD statement for the data set. However, if you are using GRS across systems or JES3, the data set might not be shared depending on the disposition of the system.

To get exclusive control of the data set's volume, a task in one system issues the RESERVE macro. The level of cross-system sharing allowed by VSAM applies only in a multiple operating system environment.

The cross-system sharing options are ignored by VSAM RLS or DFSMStvs processing. The values follow:

1 Reserved

2 Reserved

3 Specifies that the data set can be fully shared. With this option, each user is responsible for maintaining both read and write integrity for the data that user's program accesses. User programs that ignore write integrity guidelines can cause VSAM program checks, uncorrectable data set errors, and other unpredictable results. This option requires each user to be responsible for maintenance. The RESERVE and DEQ macros are required with this option to maintain data set integrity. (For information on using RESERVE and DEQ, see *z/OS MVS Programming: Authorized Assembler Services Reference ALE-DYN* and *z/OS MVS Programming: Authorized Assembler Services Reference LLA-SDU*.)

4 Indicates that the data set can be fully shared. For each request, VSAM refreshes the buffers used for direct processing. This option requires that you use the RESERVE and DEQ macros to maintain data integrity while sharing the data set. Improper use of the RESERVE macro can cause problems similar to those described under SHAREOPTIONS 3. (For information on using RESERVE and DEQ, see *z/OS MVS Programming: Authorized Assembler Services Reference ALE-DYN* and *z/OS MVS Programming: Authorized Assembler Services Reference LLA-SDU*.) Output processing is limited to update, or add processing, or both that does not change either the high-used RBA or the RBA of the high key data control interval if DISP=SHR is specified.

To ensure data integrity in a shared environment, VSAM provides users of SHAREOPTIONS 4 (cross-region and cross-system) with the following assistance:

- Each PUT request immediately writes the appropriate buffer to the VSAM cluster's DASD space. That is, the buffer in the user's address space that

contains the new or updated data record, and the buffers that contain new or updated index records when the user's data is key-sequenced.

- Each GET request refreshes all the user's input buffers. The contents of each data and index buffer being used by the user's program is retrieved from the VSAM cluster's DASD.

Abbreviation: SHR

SPANNED|NONSPANNED

Specifies whether a data record is allowed to cross control interval boundaries.

If SPANNED or NONSPANNED is specified in the SMS data class, the value defined is used as the data set definition, unless it has been previously defined with an explicitly specified or modeled DEFINE attribute.

This parameter cannot be used when defining a linear data set cluster.

SPANNED

Specifies that, if the maximum length of a data record (as specified with RECORDSIZE) is larger than a control interval, the record is contained on more than one control interval. This allows VSAM to select a control interval size that is optimum for the DASD.

When a data record that is larger than a control interval is put into a cluster that allows spanned records, the first part of the record completely fills a control interval. Subsequent control intervals are filled until the record is written into the cluster. Unused space in the record's last control interval is not available to contain other data records.

Attention: Using this parameter for a variable-length relative record data set causes an error.

Abbreviation: SPND

NONSPANNED

Indicates that the record must be contained in one control interval. VSAM selects a control interval size that accommodates your largest record.

Abbreviation:NSPND

SPEED|RECOVERY

Specifies whether the data component's control areas are to be preformatted before alternate index records are loaded into them.

This parameter is only considered during the actual loading (creation) of a data set. Creation occurs when the data set is opened and the high-used RBA is equal to zero. After normal CLOSE processing at the completion of the load operation, the physical structure of the data set and the content of the data set extents are exactly the same. Any processing of the data set after the successful load operation is the same, and the specification of this parameter is not considered.

If you use RECOVERY, the initial load takes longer because the control areas are first written with either empty or software end-of-file intervals. These preformatted CIs are then updated, using update writes, with the alternate index records. SPEED is recommended, because the initial load is quicker.

SPEED

Does not preformat the data component's space.

If the initial load is unsuccessful, you must load the alternate index records again from the beginning, because VSAM cannot determine the location of

your last correctly written record. VSAM cannot find a valid end-of-file indicator when it searches your alternate index records.

RECOVERY
Specifies that the data component's control areas are written with records that indicate end-of-file. When an alternate index record is written into a control interval, it is always followed by a record that identifies the record just written as the last record in the alternate index.

RECOVERY is a way to verify storage that is used on the device for each CA before the data is actually written.

Abbreviation: RCVY

STORAGECLASS(*class*)
For SMS-managed data sets: Gives the name, 1 to 8 characters, of the storage class.

Your storage administrator defines the names of the storage classes you can use. A storage class is assigned either when you use STORAGECLASS, or an ACS routine selects a storage class for the new data set. The storage class provides the storage attributes that are specified on the UNIT and VOLUME operand for non-SMS managed data sets. Use the storage class to select the storage service level to be used by SMS for storage of the data set. If SMS is inactive and STORAGECLASS is used, the DEFINE will be unsuccessful.

STORAGECLASS cannot be selected as a subparameter of DATA or INDEX.

Abbreviation: STORCLAS

TO(*date*)|FOR(*days*)
Specifies the retention period for the cluster being defined. If neither TO nor FOR is used, the cluster can be deleted at any time. The MANAGEMENTCLASS maximum retention period, if selected, limits the retention period specified by this parameter.

For non-SMS-managed data sets, the correct retention period is reflected in the catalog entry. The VTOC entry cannot contain the correct retention period. Enter a LISTCAT command for the correct expiration date.

For SMS-managed data sets, the expiration date in the catalog is updated and the expiration date in the format-1 DSCB is changed. If the expiration date in the catalog does not agree with the expiration date in the VTOC, the VTOC entry overrides the catalog entry.

TO(*date*)
Specifies the date up to which to keep the cluster before it is allowed to be deleted. The date is given in the form [*yy*]*yyddd*, in which *yyyy* is a four-digit year, *yy* is a two-digit year, and *ddd* is a three-digit day of the year (001 through 366). Two-digit years are treated as if 19 is specified as the first two digits of *yyyy*. The dates (19)99365 and (19)99366 are considered never-expire dates.

For expiration dates of January 1, 2000, and later, you must use the form TO(*yyyyddd*).

The maximum value of *yyyy* is 2155, and the maximum value of *ddd* is 365 for nonleap years and 366 for leap years.

FOR(*days*)
Shows the number of days to keep the cluster being defined. The

maximum number is 9999. If the number is 0 through 9998, the cluster is retained for the number of days; if the number is 9999, the cluster is retained indefinitely.

WRITECHECK|NOWRITECHECK
Indicates whether the cluster or component is to be checked by a machine action called write check when a record is written into it.

The WRITECHECK setting is ignored when the data set is opened for VSAM RLS or DFSMStvs access.

WRITECHECK
Shows that a record is written and then read, without data transfer, to test for the data check condition.

Abbreviation: WCK

NOWRITECHECK
Specifies that the cluster or component is not to be checked by a write check.

Abbreviation: NWCK

Data and index components of a cluster

You should use attributes separately for the cluster's data and index components. A list of the DATA and INDEX parameters is provided at the beginning of this topic. These parameters are described in detail as parameters of the cluster as a whole. Restrictions are noted with each parameter's description.

DEFINE CLUSTER examples

The following examples show functions that the DEFINE CLUSTER command can perform.

Define an SMS-managed key-sequenced cluster: Example 1: In this example, an SMS-managed key-sequenced cluster is defined. The DEFINE CLUSTER command builds a catalog entry and allocates space to define the key-sequenced cluster SMS04.KSDS01.

```
//DEFINE   JOB  ...
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *
    DEFINE CLUSTER -
           (NAME (SMS04.KSDS01) -
           STORAGECLASS (FINCE02) -
           MANAGEMENTCLASS (MC1985) -
           DATACLASS (VSAMDB05))
/*
```

The parameters for this command follow:

- STORAGECLASS specifies an installation-defined name of a storage class, FINCE02, to be assigned to this cluster.
- MANAGEMENTCLASS specifies an installation-defined name of a management class, MC1985, to be assigned to this cluster. Attributes of MANAGEMENTCLASS control the data set's retention, backup, migration, etc.
- DATACLASS specifies an installation-defined name of a data class, VSAMDB05, to be assigned to this cluster. Record size, key length and offset, space allocation, etc., are derived from the data class and need not be specified.

Define an SMS-managed key-sequenced cluster specifying data and index parameters: Example 2: In this example, an SMS-managed key-sequenced cluster is defined. The SMS data class space allocation is overridden by space allocations at the data and index levels. The DEFINE CLUSTER command builds a catalog entry and allocates space to define the key-sequenced cluster SMS04.KSDS02.

```
//DEFINE   JOB  ...
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *
    DEFINE CLUSTER -
           (NAME (SMS04.KSDS02) -
           STORAGECLASS (FINCE02) -
           MANAGEMENTCLASS (MC1985) -
           DATACLASS (VSAMDB05)) -
           LOG(ALL) -
           LOGSTREAMID(LogA) -
         DATA -
           (MEGABYTES (10 2)) -
         INDEX -
           (KILOBYTES (25 5))
/*
```

The parameters for this command are as follows:

- STORAGECLASS is an installation defined name of a storage class, FINCE02, to be assigned to the cluster.
- MANAGEMENTCLASS is an installation defined name of a management class, MC1985, to be assigned to the cluster. Attributes associated with a management class control the cluster's retention, backup, migration, etc.
- DATACLASS is an installation-defined name of a data class, VSAMDB05, assigned to the cluster. Record size, key length and offset, etc., are derived from the data class and need not be specified. If MAXVOLUMES or the space parameters (MEGABYTES and KILOBYTES) were not specified, the values in the data class would be used.
- LOG(ALL) specifies that changes to the sphere accessed in RLS and DFSMStvs mode can be backed out and forward recovered using external logs.
- LOGSTREAMID gives the name of the forward recovery log stream.

The DATA and INDEX parameters follow:

- MEGABYTES, used for DATA, allocates a primary space of 10 megabytes to the data component. A secondary space of 2 megabytes is specified for extending the data component.
- KILOBYTES, used for INDEX, allocates a primary space of 25 kilobytes to the index component. A secondary space of 5 kilobytes is specified for extending the index component.

Define a key-sequenced cluster specifying data and index parameters: Example 3: In this example, a key-sequenced cluster is defined. The DATA and INDEX parameters are specified and the cluster's data and index components are explicitly named. This example assumes that an alias name VWX is defined for the catalog RSTUCAT1. This naming convention causes VWX.MYDATA to be cataloged in RSTUCAT1.

```
//DEFCLU1  JOB    ...
//STEP1    EXEC   PGM=IDCAMS
//SYSPRINT DD     SYSOUT=A
//SYSIN    DD     *
    DEFINE CLUSTER -
           (NAME(VWX.MYDATA) -
           VOLUMES(VSER02) -
```

```
          RECORDS(1000 500)) -
        DATA -
          (NAME(VWX.KSDATA) -
          KEYS(15 0) -
          RECORDSIZE(250 250) -
          FREESPACE(20 10) -
          BUFFERSPACE(25000)  ) -
        INDEX -
          (NAME(VWX.KSINDEX) -
        CATALOG (RSTUCAT1)
/*
```

The DEFINE CLUSTER command builds a cluster entry, a data entry, and an index entry to define the key-sequenced cluster VWX.MYDATA. The parameters for the cluster as a whole are:

- NAME indicates that the cluster's name is VWX.MYDATA.
- VOLUMES is used when the cluster is to reside on volume VSER02.
- RECORDS specifies that the cluster's space allocation is 1000 data records. The cluster is extended in increments of 500 records. After the space is allocated, VSAM calculates the amount required for the index and subtracts it from the total.

In addition to the parameters specified for the cluster as a whole, DATA and INDEX parameters specify values and attributes that apply only to the cluster's data or index component. The parameters specified for the data component of VWX.MYDATA are:

- NAME indicates that the data component's name is VWX.KSDATA.
- KEYS shows that the length of the key field is 15 bytes and that the key field begins in the first byte (byte 0) of each data record.
- RECORDSIZE specifies fixed-length records of 250 bytes.
- BUFFERSPACE verifies that a minimum of 25 000 bytes must be provided for I/O buffers. A large area for I/O buffers can help to improve access time with certain types of processing. For example, with direct processing if the high-level index can be kept in virtual storage, access time is reduced. With sequential processing, if enough I/O buffers are available, VSAM can perform a read-ahead, thereby reducing system overhead and minimizing rotational delay.
- FREESPACE specifies that 20% of each control interval and 10% of each control area are to be left free when records are loaded into the cluster. After the cluster's records are loaded, the free space can be used to contain new records.

The parameters specified for the index component of VWX.MYDATA are:

- NAME specifies that the index component's name is VWX.KSINDEX.
- CATALOG specifies the catalog name.

Define a key-sequenced cluster and an entry-sequenced cluster: Example 4: In this example, two VSAM clusters are defined. The first DEFINE command defines a key-sequenced VSAM cluster, VWX.EXAMPLE.KSDS1. The second DEFINE command defines an entry-sequenced VSAM cluster, KLM.EXAMPLE.ESDS1. In both examples, it is assumed that alias names, VWX and KLM, have been defined for user catalogs RSTUCAT1 and RSTUCAT2, respectively.

```
//DEFCLU2  JOB    ...
//STEP1    EXEC   PGM=IDCAMS
//SYSPRINT DD     SYSOUT=A
//SYSIN    DD     *
    DEFINE CLUSTER -
           (NAME(VWX.EXAMPLE.KSDS1) -
```

```
          MODEL(VWX.MYDATA) -
          VOLUMES(VSER02) -
          NOIMBED  )
    DEFINE CLUSTER -
          (NAME(KLM.EXAMPLE.ESDS1) -
          RECORDS(100 500) -
          RECORDSIZE(250 250) -
          VOLUMES(VSER03) -
          NONINDEXED  )
/*
```

The first DEFINE command builds a cluster entry, a data entry, and an index entry to define the key-sequenced cluster VWX.EXAMPLE.KSDS1. Its parameters are:

- NAME specifies the name of the key-sequenced cluster, VWX.EXAMPLE.KSDS1. The cluster is defined in the user catalog for which VWX has been established as an alias.
- MODEL identifies VWX.MYDATA as the cluster to use as a model for VWX.EXAMPLE.KSDS1. The attributes and specifications of VWX.MYDATA that are not otherwise specified with the DEFINE command parameters are used to define the attributes and specifications of VWX.EXAMPLE.KSDS1. VWX.MYDATA is located in the user catalog for which VWX has been established as an alias.
- VOLUMES specifies that the cluster is to reside on volume VSER02.
- NOIMBED specifies that space is not to be allocated for sequence-set control intervals within the data component's physical extents.

The second DEFINE command builds a cluster entry and a data entry to define an entry-sequenced cluster, KLM.EXAMPLE.ESDS1. Its parameters are:

- NAME specifies the name of the entry-sequenced cluster, KLM.EXAMPLE.ESDS1. The cluster is defined in the user catalog for which KLM has been established as an alias.
- RECORDS specifies that the cluster space allocation is 100 records. When the cluster is extended, it is extended in increments of 500 records.
- RECORDSIZE specifies that the cluster records are fixed length (the average record size equals the maximum record size) and 250 bytes long.
- VOLUMES specifies that the cluster is to reside on volume VSER03.
- NONINDEXED specifies that the cluster is to be an entry-sequenced cluster.

Define a relative record cluster in a catalog: Example 5: In this example, a relative record cluster is defined.

```
//DEFCLU4  JOB   ...
//STEP1    EXEC  PGM=IDCAMS
//SYSPRINT DD    SYSOUT=A
//SYSIN    DD    *
     DEFINE CLUSTER -
           (NAME(EXAMPLE.RRDS1) -
           RECORDSIZE(100 100) -
           VOLUMES(VSER01) -
           TRACKS(10 5) -
           NUMBERED) -
         CATALOG(USERCAT)
/*
```

The DEFINE CLUSTER command builds a cluster entry and a data entry to define the relative record cluster EXAMPLE.RRDS1 in the user catalog. The DEFINE CLUSTER command allocates ten tracks for the cluster's use. The command's parameters are:

- NAME specifies that the cluster's name is EXAMPLE.RRDS1.
- RECORDSIZE specifies that the records are fixed-length, 100 byte records. Average and maximum record length must be equal for a fixed-length relative record data set, but not equal for a variable-length RRDS.
- VOLUMES specifies that the cluster is to reside on volume VSER01. This example assumes that the volume is already cataloged in the user catalog, USERCAT.
- TRACKS specifies that 10 tracks are allocated for the cluster. When the cluster is extended, it is to be extended in increments of 5 tracks.
- NUMBERED specifies that the cluster's data organization is to be relative record.
- CATALOG specifies the catalog name.

The DEFINE CLUSTER command builds a cluster entry and a data entry to define the relative record cluster EXAMPLE.RRDS1 in the user catalog. The DEFINE CLUSTER command allocates ten tracks for the cluster's use. The command's parameters are:

- NAME specifies that the cluster's name is EXAMPLE.RRDS1.
- RECORDSIZE specifies that the records are fixed-length, 100 byte records. Average and maximum record length must be equal for a fixed-length relative record data set, but not equal for a variable-length RRDS.
- VOLUMES specifies that the cluster is to reside on volume VSER01. This example assumes that the volume is already cataloged in the user catalog, USERCAT.
- TRACKS specifies that 10 tracks are allocated for the cluster. When the cluster is extended, it is to be extended in increments of 5 tracks.
- NUMBERED specifies that the cluster's data organization is to be relative record.
- CATALOG specifies the catalog name.

Define a reusable entry-sequenced cluster in a catalog: Example 6: In this example, a reusable entry-sequenced cluster is defined. You can use the cluster as a temporary data set. Each time the cluster is opened, its high-used RBA can be reset to zero.

```
//DEFCLU5  JOB   ...
//STEP1    EXEC  PGM=IDCAMS
//SYSPRINT DD    SYSOUT=A
//SYSIN    DD    *
     DEFINE CLUSTER -
           (NAME(EXAMPLE.ESDS2) -
           RECORDSIZE(2500 3000) -
           SPANNED -
           VOLUMES(VSER03) -
           CYLINDERS(2 1) -
           NONINDEXED -
           REUSE -
           CATALOG(RSTUCAT2)
/*
```

The DEFINE CLUSTER command builds a cluster entry and a data entry to define the entry-sequenced cluster, EXAMPLE.ESDS2. The DEFINE CLUSTER command assigns two tracks for the cluster's use. The command's parameters are:

- NAME specifies that the cluster's name is EXAMPLE.ESDS2.
- RECORDSIZE specifies that the records are variable length, with an average size of 2500 bytes and a maximum size of 3000 bytes.
- SPANNED specifies that data records can cross control interval boundaries.

- VOLUMES specifies that the cluster is to reside on volume VSER03.
- CYLINDERS specifies that two cylinders are to be allocated for the cluster's space. When the cluster is extended, it is to be extended in increments of 1 cylinder.
- NONINDEXED specifies that the cluster's data organization is to be entry-sequenced. This parameter overrides the INDEXED parameter.
- REUSE specifies that the cluster is to be reusable. Each time the cluster is opened, its high-used RBA can be reset to zero and it is effectively an empty cluster.
- CATALOG specifies that the cluster is to be defined in a user catalog, RSTUCAT2.

Define a key-sequenced cluster in a catalog: Example 7: In this example, a key-sequenced cluster is defined. In other examples, an alternate index is defined over the cluster, and a path is defined that relates the cluster to the alternate index. The cluster, its alternate index, and the path entry are all defined in the same catalog, USERCAT.

```
//DEFCLU6  JOB   ...
//STEP1    EXEC  PGM=IDCAMS
//SYSPRINT DD    SYSOUT=A
//SYSIN    DD    *
     DEFINE CLUSTER -
           (NAME(EXAMPLE.KSDS2)) -
         DATA -
           (RECORDS(500 100) -
           EXCEPTIONEXIT(DATEXIT) -
           ERASE -
           FREESPACE(20 10) -
           KEYS(6 4) -
           RECORDSIZE(80 100) -
           VOLUMES(VSER01) ) -
         INDEX -
           (RECORDS(300 300) -
           VOLUMES(VSER01) ) -
         CATALOG(USERCAT)
/*
```

The DEFINE CLUSTER command builds a cluster entry, a data entry, and an index entry to define the key-sequenced cluster, EXAMPLE.KSDS2. The DEFINE CLUSTER command allocates space separately for the cluster's data and index components.

The parameter that applies to the cluster is NAME, which specifies that the cluster's name is EXAMPLE.KSDS2.

The parameters that apply only to the cluster's data component are enclosed in the parentheses following the DATA keyword:

- RECORDS specifies that an amount of tracks equal to at least 500 records is to be allocated for the data component's space. When the data component is extended, it is to be extended in increments of tracks equal to 100 records.
- EXCEPTIONEXIT specifies the name of the exception exit routine, DATEXIT, that is to be processed if an I/O error occurs while a data record is being processed.
- ERASE specifies that the cluster's data is to be erased (overwritten with binary zeros) when the cluster is deleted.

- FREESPACE specifies the amounts of free space to be left in the data component's control intervals (20%) and the control areas (10% of the control intervals in the control area) when data records are loaded into the cluster.
- KEYS specifies the location and length of the key field in each data record. The key field is 6 bytes long and begins in the fifth byte (byte 4) of each data record.
- RECORDSIZE specifies that the cluster's records are variable length, with an average size of 80 bytes and a maximum size of 100 bytes.
- VOLUMES specifies that the cluster is to reside on volume VSER01.

The parameters that apply only to the cluster's index component are enclosed in the parentheses following the INDEX keyword:

- RECORDS specifies that an amount of tracks equal to at least 300 records is to be allocated for the index component's space. When the index component is extended, it is to be extended in increments of tracks equal to 300 records.
- VOLUMES specifies that the index component is to reside on volume VSER01.

The CATALOG parameter specifies that the cluster is to be defined in a user catalog, USERCAT4.

Define an entry-sequenced cluster using a model: Example 8: In this example, two entry-sequenced clusters are defined. The attributes of the second cluster defined are modeled from the first cluster.

```
//DEFCLU7  JOB    ...
//STEP1   EXEC   PGM=IDCAMS
//SYSPRINT DD     SYSOUT=A
//SYSIN    DD     *
    DEFINE CLUSTER -
           (NAME(GENERIC.A.BAKER) -
           VOLUMES(VSER02) -
           RECORDS(100 100) -
           RECORDSIZE(80 80) -
           NONINDEXED ) -
         CATALOG(USERCAT4)
    DEFINE CLUSTER -
           (NAME(GENERIC.B.BAKER) -
           MODEL(GENERIC.A.BAKER USERCAT4)) -
         CATALOG(USERCAT4)
/*
```

The first DEFINE CLUSTER command defines an entry-sequenced cluster, GENERIC.A.BAKER. Its parameters are:

- NAME specifies the name of the entry-sequenced cluster, GENERIC.A.BAKER.
- VOLUMES specifies that the cluster is to reside on volume VSER02.
- RECORDS specifies that the cluster's space allocation is 100 records. When the cluster is extended, it is extended in increments of 100 records.
- RECORDSIZE specifies that the cluster's records are fixed length (the average record size equals the maximum record size) and 80 bytes long.
- NONINDEXED specifies that the cluster is entry-sequenced.
- CATALOG specifies that the cluster is to be defined in the USERCAT4 catalog.

The second DEFINE CLUSTER command uses the attributes and specifications of the previously defined cluster, GENERIC.A.BAKER, as a model for the cluster still to be defined, GENERIC.B.BAKER. A list of the parameters follows:

- NAME specifies the name of the entry-sequenced cluster, GENERIC.B.BAKER.

- MODEL identifies GENERIC.A.BAKER, cataloged in user catalog USERCAT4, as the cluster to use as a model for GENERIC.B.BAKER. The attributes and specifications of GENERIC.A.BAKER that are not otherwise specified with the DEFINE command's parameters are used to define the attributes and specifications of GENERIC.B.BAKER.
- CATALOG specifies that the cluster is to be defined in the USERCAT4 catalog.

Define a VSAM volume data set: Example 9: In this example, a VVDS is explicitly defined. The cluster is named using the restricted VVDS name format 'SYS1.VVDS.Vvolser'.

```
//DEFCLU8  JOB    ...
//STEP1   EXEC   PGM=IDCAMS
//SYSPRINT DD     SYSOUT=A
//SYSIN    DD     *
    DEFINE CLUSTER -
           (NAME(SYS1.VVDS.VVSER03) -
           VOLUMES(VSER03) -
           NONINDEXED -
           CYLINDERS(1 1)) -
         CATALOG(USERCAT4)
/*
```

This DEFINE CLUSTER command defines an entry-sequenced cluster that is used as a VVDS. The parameters are:

- NAME specifies the name of a VVDS, 'SYS1.VVDS.Vvolser', SYS1.VVDS.VVSER03.
- VOLUMES specifies that the cluster is to reside on volume VSER03. Only one volume serial can be specified.
- NONINDEXED specifies that the cluster is entry-sequenced.
- CYLINDERS specifies that the cluster's space allocation is 1 cylinder. When the cluster is extended, it is extended in increments of 1 cylinder.
- CATALOG specifies that the cluster is to be defined in the USERCAT4 catalog.

Define a relative record data set with expiration date beyond 1999: Example 10: In this example, an entry-sequenced cluster is defined specifying an expiration date beyond the year 1999, using the TO parameter.

```
//DEFCLU8  JOB    ...
//STEP1   EXEC   PGM=IDCAMS
//SYSPRINT DD     SYSOUT=A
//SYSIN    DD     *
     DEFINE CLUSTER -
           (NAME(EXAMPLE.RRDS1) -
           RECORDSIZE(100 100) -
           VOLUMES(VSER01) -
           TRACKS(10 5) -
           NUMBERED -
           TO(2015012) ) -
         CATALOG(USERCAT)
/*
```

The DEFINE CLUSTER command builds a cluster entry and a data entry to define the relative record cluster, EXAMPLE.RRDS1, in the user catalog, USERCAT. The DEFINE CLUSTER command allocates ten tracks for the cluster's use. The expiration date is set to January 12, 2015. The parameters are:

- NAME specifies that the cluster's name is EXAMPLE.RRDS1.

- RECORDSIZE specifies that the records are fixed-length, 100-byte records. Average and maximum record length must be equal for a fixed-length relative record data set, but not equal for a variable-length RRDS.
- VOLUMES specifies that the cluster is to reside on volume VSER01.
- TRACKS specifies that ten tracks are allocated for the cluster. When the cluster is extended, it is to be extended in increments of five tracks.
- NUMBERED specifies that the cluster's data organization is to be relative record.
- TO specifies that the retention period is set to expire January 12, 2015. Note that the year (2015) is specified as a four-digit number and concatenated with the day (012). A four-digit year must be specified when the expiration date is beyond 1999. The retention period could also have been set by using the FOR parameter, followed by the number of days the cluster is to be retained.
- CATALOG specifies that the cluster is to be defined in a user catalog, USERCAT.

Define a linear data set cluster in a catalog: Example 11: In this example, a linear data set cluster is defined in a catalog.

```
//DEFLDS   JOB    ...
//STEP1   EXEC   PGM=IDCAMS
//SYSPRINT DD     SYSOUT=A
//SYSIN    DD     *
    DEFINE CLUSTER -
           (NAME(EXAMPLE.LDS01) -
           VOLUMES(VSER03) -
           TRACKS(20 10) -
           LINEAR -
         CATALOG(USERCAT)
/*
```

The DEFINE CLUSTER command builds a cluster entry and a data entry to define the linear data set cluster EXAMPLE.LDS01. The parameters are:

- NAME specifies that the cluster's name is EXAMPLE.LDS01.
- VOLUMES specifies that the cluster is to reside on volume VSER03.
- TRACKS specifies that 20 tracks are allocated for the cluster's space. When the cluster is extended, it is to be extended in increments of 10 tracks.
- LINEAR specifies that the cluster's data organization is to be linear.
- CATALOG specifies that the cluster is to be defined in a user catalog, USERCAT.

Securing log streams

You must define authorization for system logger resources so that DFSMStvs can access, read, and write to its log streams. DFSMStvs uses undo, shunt, log of logs, and forward recovery log streams. This authorization applies to log streams in the coupling facility and DASD-only log streams. You can use RACF, a component of the z/OS Security Server, or an equivalent security product to secure log streams and implement DFSMStvs access to them.

For more information about authorizing DFSMStvs access to log streams, see *z/OS DFSMStvs Planning and Operating Guide*.

Chapter 3. Customizing the DFSMStvs environment

This topic contains information that is Programming Interface information.

This topic describes DFSMS macro instructions that you can use for DFSMStvs. For more information, see *z/OS DFSMS Macro Instructions for Data Sets*.

Coding VSAM macros

This topic contains VSAM macro formats and examples.

The macros that work at assembly time enable you to specify subparameter values as absolute numeric expressions, character strings, codes, and expressions that generate valid relocatable A-type address constants.

The macros that work at execution also enable you to specify these values as follows:

- Register notation, in which the expression designating a register from 2 through 12 is enclosed in parentheses. For example, (2) and (REG), where REG is a label equated to a number from 2 through 12.
- An expression of the form (S,*scon*), in which scon is an expression valid for an S-type address constant, including the base-displacement form.
- An expression of the form (*,*scon*), in which scon is an expression valid for an S-type address constant, including the base-displacement form, and the address specified by scon is indirect—that is, it gives the location of the area that contains the value for the subparameter.

For most programming applications, you can use register notation or absolute numeric expressions for numbers, character strings for names, and register notation or expressions that generate valid A-type address constants for addresses. "Subparameters with GENCB, MODCB, SHOWCB, and TESTCB" gives all the ways of coding each parameter for the macros that work at execution time.

You can write a reentrant program **only** with execution-time macros. "Use of list, execute, and generate forms of VSAM macros" on page 132 describes alternative ways of coding these macros for reentrant programs. This topic describes the standard form of these macros.

Subparameters with GENCB, MODCB, SHOWCB, and TESTCB

The addresses, names, numbers, and options required with subparameters in GENCB, MODCB, SHOWCB, and TESTCB can be expressed in a variety of ways:

- An **absolute numeric expression**, for example, STRNO=3 and COPIES=10.
- A **code or a list of codes separated by commas and enclosed in parentheses**, for example, OPTCD=KEY or OPTCD=(KEY,DIR,IN).
- A **character string**, for example, DDNAME=DATASET.
- A **register from 2 through 12 that contains an address or numeric value**, for example, SYNAD=(3); equated labels can be used to designate a register, for example, SYNAD=(ERR), where the following equate statement has been included in the program: ERR EQU 3.

- An **expression of the form (S,scon)**, where scon is an expression valid for an S-type address constant, including the base-displacement form. The contents of the base register are added to the displacement to obtain the value of the keyword. For example, if the value of the keyword being represented is a numeric value (that is, COPIES, LENGTH, RECLEN), the contents of the base register are added to the displacement to determine the numeric value. If the value of the keyword being represented is an address constant (that is, WAREA, EXLST, EODAD, ACB), the contents of the base register are added to the displacement to determine the value of the address constant.
- An **expression of the form (*,scon)**, where scon is an expression valid for an S-type address constant, including the base-displacement form. The address specified by scon is **indirect**, that is, it is the address of an area that contains the value of the keyword. The contents of the base register are added to the displacement to determine the address of the fullword of storage that contains the value of the keyword.

  If an indirect S-type address constant is used, the value it points to must meet the following criteria:

  – If the value is a numeric quantity or an address, it must occupy a fullword of storage.
  – If the value is an alphanumeric character string, it must occupy two words of storage, be left aligned, and be filled on the right with blanks.
- An **expression valid for a relocatable A-type address constant**, for example, AREA=MYAREA+4.

The specified keyword determines the type of expressions that can be used. Also, register and S-type address constants cannot be used when MF=L is specified.

Use of list, execute, and generate forms of VSAM macros

The BLDVRP, DLVRP, GENCB, MODCB, SHOWCB, and TESTCB macros build a parameter list describing in codes the actions shown by the subparameters you specify and pass the list to VSAM to take the suggested action.

The list, execute, and generate forms of BLDVRP, DLVRP, GENCB, MODCB, SHOWCB, and TESTCB allow you to write reentrant programs, to share parameter lists, and to modify a parameter list before using it.

Following is a brief description of the list, execute, and generate forms:

- The list form is used to build the parameter list either in line (called a ***simple list***) or in an area remote from the macro expansion (called a ***remote list***). Both the simple- and the remote-list forms allow you to build a single parameter list that can be shared.
- The execute form is used to modify a parameter list and to pass it to VSAM for action.
- The generate form is used to build the parameter list in a remote area and to pass it to VSAM for action.

The list, execute, and generate forms of the BLDVRP, DLVRP, GENCB, MODCB, SHOWCB, and TESTCB macros have the same format as the standard forms, except for:

- An additional keyword, **MF**.
- Keywords that are required in the standard form may be optional in the list, execute, and generate forms or may not be allowed in the execute form. The

meaning of the keywords, however, and the notation that may be used to express addresses, names, numbers, and option codes are the same.

This topic describe the format of the MF keyword and the use of list, execute, and generate forms. They also show the optional and invalid subparameters.

List-form keyword

The format of the MF keyword for the list form is:

MF={L | (L,*address***[,***label***])}**

where:

L Specifies that this is the list form of the macro.

address
Specifies the address of a remote area in which the parameter list is to be built. The area must begin on a fullword boundary. You can specify the address in register notation or as an expression valid for a relocatable A-type address constant or a direct or indirect S-type address constant.

label
Specifies a unique name used in an EQU instruction in the expansion of the macro. *Label* is equated to the length of the parameter list. You do not have to know the length of the parameter list if you code *label*; the expansion of the macro determines the amount of storage required.

Because the MF=L expansion does not include executable code, register notation and expressions that generate S-type address constants cannot be used.

If you code MF=L, the parameter list is built in line, which means that the program is not reentrant if the parameter list is modified at execution.

If you code MF=(L,address), the parameter list is built in the remote area specified, and the area must be large enough for the parameter list.

The size, in fullwords, of a parameter list is:

- For GENCB, 4, plus 3 times the number of ACB, EXLST, or RPL keywords specified (plus 1 for DDNAME, EODAD, JRNAD, LERAD, or SYNAD)
- For MODCB, 3, plus 3 times the number of ACB, EXLST, or RPL keywords specified (plus 1 for DDNAME, EODAD, JRNAD, LERAD, or SYNAD)
- For SHOWCB, 5, plus 2 times the number of fields specified in the FIELDS keyword
- For TESTCB, 8 (plus 1 for either DDNAME, STMST, EODAD, JRNAD, LERAD, or SYNAD).

If you code MF=(L,address,label), the parameter list is built in the remote area specified. The expansion of the macro equates *label* with the length of the parameter list.

Execute-form keyword

The format of the MF keyword for the execute form is:

MF=(E,*address***)**

where:

E Specifies that this is the execute form of the macro.

address
Specifies the address of the parameter list.

Expansion of the execute form of the macro results in executable code that causes:

1. A parameter list to be modified, if requested
2. Control to be passed to a routine that satisfies the request.

You may not use the execute form to add an entry to a parameter list. If you try to add an entry, you receive a return code of 8 in register 15.

Generate-form keyword

The format of the MF keyword for the generate form is:

MF=(G,*address***[,***label***])**

where:

G Specifies that this is the generate form of the macro.

address
Specifies the address of a remote area in which the parameter list is to be built. The area must begin on a fullword boundary.

label
Specifies a unique name that is used in an EQU instruction in the expansion of the macro. *Label* is equated to the length of the parameter list. You do not have to know the length of the parameter list if you code *label*; the expansion of the macro determines the amount of storage required.

If you code MF=(G,address), the parameter list is built in the remote area specified.

If you code MF=(G,address,label), the parameter list is built in the remote area specified. The expansion of the macro equates the length of the parameter list to *label*.

Examples of generate, list, and execute forms

Table 9 shows which forms of GENCB, MODCB, SHOWCB, and TESTCB to use in reentrant or nonreentrant and shared or nonshared environments.

Table 9. Reentrant programming. Reentrant programmingThis table shows which forms of GENCB, MODCB, SHOWCB, and TESTCB to use in reentrant or nonreentrant and shared or nonshared environments.

	Reentrant	**Nonreentrant**
Shared	MF=(L,*address*[,*label*])	MF=L
	MF=(E,*address*)	MF=(E,*address*)
Nonshared	MF=(G,*address*[,*label*])	Standard Form

As Table 9 shows, these guidelines apply to reentrant programming:

- To share parameter lists in a reentrant program, you should use the remote-list form with the execute form.
- To share parameter lists in a nonreentrant program, you should use the simple-list form should be used with the execute form.

- If you do not intend to share parameter lists, you should use the generate form for reentrant programs and the standard form for nonreentrant programs.

The following examples show how the generate, list, and execute forms work.

Example: Generate form (reentrant)

In this example, the generate form of GENCB is used to create a default request parameter list (RPL) in a reentrant environment.

```
LA      10,LEN1          Get length of the parameter list.

GETMAIN R,LV=(10)        Get storage for the area in which      x
                         the parameter list is to be built.     x
LR      2,1              Save address of parameter-list area.

GENCB   BLK=RPL,                                                x
        MF=(G,(2),LEN1)
```

The macro expansion equates LEN1 to the length of the parameter list, as follows:

```
+LEN1 EQU 16
```

The parameter list is built in the area acquired by the GETMAIN macro and pointed to by register 2. This list is used by VSAM to build the RPL. VSAM returns the RPL address in register 1 and the RPL length in register 0. If the WAREA and LENGTH parameters are used, the RPL is built at the WAREA address.

Example: Remote-list form (reentrant)

In this example, the remote-list form of MODCB is used to build a parameter list that will later be used to modify the MACRF bits in the access method control block ANYACB.

```
LA      8,LEN2           Get length of the parameter list.

GETMAIN R,LV=(8)         Get storage for the area in which the    x
                         parameter list is to be built.
LR      3,1              Save address of the parameter-list area.

MODCB   ACB=ANYACB,                                              x
        MACRMF=(L,(3),LEN2)
```

The macro expansion equates the length of the parameter list to LEN2, as follows:

```
+LEN2 EQU 24
```

This parameter list is built in the remote area pointed to by register 3. The list is used by VSAM to modify the ACB when an execute form of MODCB is issued (see next example). The list form only creates a parameter list; it does not modify the ACB.

Example: Execute form (reentrant)

In this example, the execute form of MODCB is used to modify the address of the access method control block and MACRF codes in the parameter list created by the remote-list form of MODCB in the previous example.

```
MODCB  ACB=MYACB,MACRF=(ADR,SEQ,OUT),MF=(E,(3))
```

The parameter list pointed to by register 3 is changed so that the ACB and MACRF parameter values in the execute form override those in the list form. The access method control block, MYACB, is then modified to MACRF=(ADR,SEQ,OUT).

The access method control block at ANYACB is not changed by either of these examples.

ACB—generate an access method control block at assembly time

Use the ACB macro to generate an access method control block at assembly time.

The format of the ACB macro follows.

The format of the ACB macro.

Label	Operand	Parameters
[*label*]	**ACB**	**[AM=VSAM]** **[,BSTRNO=***abs expression***]** **[,BUFND=***abs expression***]** **[,BUFNI=***abs expression***]** **[,BUFSP=***abs expression***]** **[,DDNAME=***character string***]** **[,EXLST=***address***]** **[,MACRF=([ADR][,CNV][,KEY]** **[,CFX | NFX]** **[DDN | DSN]** **[,DFR | NDF]** **[,DIR][,SEQ][,SKP]** **[,ICI | NCI]** **[,IN][,OUT]** **[,LEW | NLW]** **[,NIS | SIS]** **[,NRM | AIX]** **[,NRS | RST]** **[,NSR | LSR | GSR | RLS]** **[,NUB | UBF])]** **[,MAREA=***address***]** **[,MLEN=***abs expression***]** **[,RLSREAD={NRI | CR | CRE | NORD}]** **[,PASSWD=***address***]** **[,RMODE31={ALL | BUFF | CB | NONE}]** **[,SHRPOOL={0 | *abs expression*}]** **[,STRNO=***abs expression***]** **[,SUBSYSNM=***address***]** **[,CTRLACB={YES | NO}]**

Values for ACB macro subparameters can be specified as absolute numeric expressions, character strings, codes, and expressions that generate valid relocatable A-type address constants.

label
Specifies 1 to 8 characters that provide a symbolic address for the access method control block that is assembled. If you omit the DDNAME parameter, *label* serves as the ddname.

AM=VSAM
Specifies that the access method using this control block is VSAM.

BSTRNO=*abs expression*
Specifies the number of strings that are initially allocated for access to the base cluster of a path. BSTRNO must be a number between 0 and 255. The default

is STRNO. BSTRNO is ignored if the object being opened is not a path. If the number that is specified for BSTRNO is insufficient, VSAM dynamically extends the number of strings as needed for access to the base cluster.

BSTRNO can influence performance. The VSAM control blocks for the set of strings that is specified by BSTRNO are allocated in contiguous virtual storage. This is not guaranteed for the strings allocated by dynamic extension.

This parameter is only applicable to MACRF=NSR.

This parameter has no effect for UNIX files. This is the case when an application program uses the VSAM interface to access a UNIX file.

BUFND=*abs expression*
Specifies the number of I/O buffers that VSAM is to use for transmitting data between virtual and auxiliary storage. A buffer is the size of a control interval in the data component. BUFND must be a number between 0 and 65535. The minimum number that you can specify is 1 plus the number that is specified for STRNO. (If you omit STRNO, BUFND must be at least 2, because the default for STRNO is 1.) The number can be supplied through the JCL DD AMP parameter and through the macro. The default is the minimum number that is required. The minimum buffer specification does not provide optimum sequential processing performance. Generally, the more data buffers that are specified, the better the performance.

Additional data buffers benefit direct inserts or updates during control area splits and benefit spanned record accessing. The maximum number of buffers that is allowed is currently 255 (254 data buffers and 1 insert buffer). See *z/OS DFSMS Using Data Sets* for more information on optimizing performance and system-managed buffering.

This parameter is applicable only to MACRF=NSR; it is ignored when MACRF=RLS is specified.

This parameter has no effect for UNIX files.

BUFNI=*abs expression*
Specifies the number of I/O buffers that VSAM is to use for transmitting the contents of index entries between virtual and auxiliary storage for keyed access. A buffer is the size of a control interval in the index. BUFNI must be a number between 0 and 65535. The minimum number is the number that is specified for STRNO (if you omit STRNO, BUFNI must be at least 1, because the default for STRNO is 1). You can supply the number through the JCL DD AMP parameter and through the macro. The default is the minimum number that is required.

Additional index buffers improve performance by providing for the residency of some or all of the high-level index, thereby minimizing the number of high-level index records retrieved from DASD for key-direct processing. For more information on optimizing performance, see *z/OS DFSMS Using Data Sets*.

The default is the minimum number that is required. The maximum number of buffers allowed is currently 255 (254 data buffers and 1 insert buffer).

This parameter is only applicable to MACRF=NSR.

This parameter has no effect for UNIX files.

BUFSP=*abs expression*
Specifies the maximum number of bytes of virtual storage to be used for the data and index I/O buffers. VSAM gets the storage in your program's address

space. If you specify less than the amount of space that was specified in the BUFFERSPACE parameter of the DEFINE command when the data set was defined, VSAM overrides your BUFSP specification upward to the value that was specified in BUFFERSPACE. (BUFFERSPACE, by definition, is the least amount of virtual storage that is ever provided for I/O buffers.) However, if BUFSP is specified and the amount specified is much too small — smaller than the minimum amount of buffer storage required to process the data set — VSAM cannot open the data set. The minimum amount is described under BUFND and BUFNI, in the preceding text.

You can supply BUFSP through the JCL DD AMP parameter and through the macro. If you do not specify BUFSP in either place, the amount of storage that is used for buffer allocation is the *largest* of the following amounts:

- Amount that is specified in the catalog (BUFFERSPACE)
- Amount that is determined from BUFND and BUFNI
- Minimum storage that is required to process the data set with its specified processing options

A valid BUFSP amount takes precedence over the amount BUFND and BUFNI call for. If the BUFSP amount is greater than the amount called for by BUFND and BUFNI, the extra space is allocated under the following conditions:

- When MACRF indicates direct access only, additional index buffers are allocated.
- When MACRF indicates sequential access, one additional index buffer and as many data buffers as possible are allocated.

If the BUFSP amount is less than the amount that is called for by BUFND and BUFNI, the number of data and index buffers is decreased under the following conditions:

- When MACRF indicates direct access only, the number of data buffers is decreased to not fewer than the minimum number. Then, if required, the number of index buffers is decreased until the amount called for by BUFND and BUFNI complies with the BUFSP amount.
- When MACRF indicates sequential access, the number of index buffers is decreased to not fewer than 1 more than the minimum number. Then, if required, the number of data buffers is decreased to not fewer than the minimum number. If still required, 1 more is subtracted from the number of index buffers.
- Neither the number of data buffers nor the number of index buffers is decreased to fewer than the minimum number.

If the index does not exist or is not being opened, only BUFND, and not BUFNI, enters these calculations.

The BUFFERSPACE must not exceed 16776704.

This parameter is applicable only to MACRF=NSR; it is ignored when MACRF=RLS is specified.

This parameter has no effect for UNIX files.

DDNAME=*character string*

Specifies 1 to 8 characters that identify the data set you want to process by specifying the JCL DD statement for the data set. You may omit DDNAME and provide it through the label or through the MODCB macro before opening the data set. MODCB is described in "MODCB—modify an access method control block" on page 168.

If you code CTRLACB=YES, do not code DDNAME. Otherwise, the DDNAME value must come from some source.

EXLST=*address*

Specifies the address of a list of addresses of exit routines that you are providing. The list must be established by the EXLST or GENCB macro. If you use the EXLST macro, you can specify its label here as the address of the exit list. If you use GENCB, you can specify the address returned by GENCB in register 1 or the label of an area you supplied to GENCB for the exit list.

To use the exit list, you must code this EXLST parameter. Omitting this parameter means that you have no exit routines. Exit routines are described in *z/OS DFSMS Using Data Sets*.

MACRF=([ADR][,CNV][,KEY]
[,CFX|NFX]
[,DDN|DSN]
[,DFR|NDF]
[,DIR][,SEQ][,SKP]
[,ICI|NCI]
[,IN][,OUT]
[,LEW|NLW]
[,NIS|SIS]
[,NRM|AIX]
[,NRS|RST]
[,NSR|LSR|GSR|RLS]
[,NUB|UBF])

Specifies the kinds of processing you will do with the data set. The subparameters must be significant for the data set. For example, if you specify keyed access for an entry-sequenced data set (ESDS), you cannot open the data set. You must specify all the types of access you are going to use, whether you use them concurrently or by switching from one to the other. Table 10 gives the subparameters. Each group of subparameters has a default value (shown by underlining). You can specify subparameters in any order. You can specify both ADR and KEY to process a key-sequenced data set (KSDS). You can specify both DIR and SEQ; with keyed access, you may specify SKP as well. If you specify OUT and want merely to retrieve some records and also update, delete, or insert others, you need not also specify IN.

Table 10. MACRF options. MACRF options

Option	Meaning
ADR	Addressed access to a key-sequenced or entry-sequenced data set; RBAs are used as search arguments and sequential access is by entry sequence. VSAM RLS and DFSMStvs do not support ADR access to a KSDS.
CNV	Access is to the entire contents of a control interval rather than to an individual data record. If the data set is password protected, you must supply the address of the control or higher-level password in the ACB PASSWD parameter. **Recommendation:** Use RACF®, a component of the z/OS Security Server, or a functionally equivalent program instead of VSAM passwords. For VSAM RLS and DFSMStvs, CNV is invalid. This parameter is invalid for UNIX files and if it is specified, results in an OPEN failure.

Table 10. MACRF options (continued). MACRF options

Option	Meaning
KEY	Keyed access to a relative record data set (RRDS) or key-sequenced data set. Keys or relative record numbers are used as search arguments and sequential access is by key or relative record number. KEY processing is not affected by VSAM RLS or DFSMStvs.
CFX	OPEN fixes control blocks and I/O buffers, and they remain fixed until the ACB is closed. For VSAM RLS and DFSMStvs, CFX is ignored and NFX is assumed. This subparameter has no effect for UNIX files.
NFX	OPEN fixes control blocks and I/O buffers, and they remain fixed until the ACB is closed. For VSAM RLS and DFSMStvs, NFX is assumed.
DDN	Subtask shared control block connection is based on common ddnames. For VSAM RLS and DFSMStvs, DDN is ignored. This subparameter has no effect for UNIX files.
DSN	Subtask shared control block connection is based on common data set names. For VSAM RLS and DFSMStvs, DSN is ignored. This subparameter has no effect for UNIX files.
DFR	With shared resources, writes for direct PUT requests are deferred until the WRTBFR macro is issued or until VSAM needs a buffer to satisfy a GET request. Deferring writes saves I/O requests in cases where subsequent requests can be satisfied by the data already in the buffer pool. For VSAM RLS and DFSMStvs, DFR is ignored and direct request modified buffers are immediately written to disk and the CF (coupling facility). This subparameter has no effect for UNIX files.
NDF	Writes are not deferred for direct PUTs. For VSAM RLS and DFSMStvs, NDF is ignored and direct request modified buffers are immediately written to disk and the CF (coupling facility).
DIR	Direct access to an RRDS, KSDS, or ESDS.
SEQ	Sequential access to an RRDS, KSDS, or ESDS.
SKP	Skip-sequential access to an RRDS or KSDS. Used only with keyed access in a forward direction.
ICI	Processing is limited to improved control interval processing; access is faster because fewer processor instructions are executed. ICI processing is not allowed for extended format data sets. For VSAM RLS and DFSMStvs, ICI is invalid. This parameter is invalid for UNIX files and if specified, results in an open failure.
NCI	Processing other than improved control interval processing.
IN	Retrieval of records of a RRDS, KSDS, or ESDS; (not allowed for an empty data set). If the data set is password protected, you must supply the address of the read or higher-level password in the ACB PASSWD parameter.
OUT	Storage of new records in a RRDS, KSDS, or ESDS (not allowed with addressed access to a KSDS). Update of records in a RRDS, KSDS, or ESDS. Deletion of records from a RRDS or KSDS. If the data set is password protected, you must supply the address of the update or higher-level password in the ACB PASSWD parameter.
LEW	Using LSR, if an exclusive control conflict is encountered, VSAM defers the request until the resource becomes available. For VSAM RLS and DFSMStvs, LEW is ignored.

Table 10. MACRF options (continued). MACRF options

Option	Meaning
NLW	With this value specified, instead of deferring the request, VSAM returns the exclusive control return code 20 (X'14') to the application program. The application program is then able to determine the next action. For VSAM RLS and DFSMStvs, NLW is ignored.
NIS	Normal insert strategy. This subparameter has no effect for UNIX files.
SIS	Sequential insert strategy (split control intervals and control areas at the insert point rather than at the midpoint when doing direct PUTs); although positioning is lost and writes are done after each direct PUT request, SIS allows more efficient space usage when direct inserts are clustered around certain keys. This subparameter has no effect for UNIX files.
NRM	The object to be processed is the one named in the specified ddname. For VSAM RLS and DFSMStvs, NRM does not allow the direct open of an alternate index.
AIX	The object to be processed is the alternate index of the path specified by ddname, rather than the base cluster through the alternate index. For VSAM RLS and DFSMStvs, the AIX subparameter is invalid. This subparameter has no effect for UNIX files.
NRS	Data set is not reusable.
RST	Data set is reusable (high-used RBA is reset to 0 during OPEN). If the data set is password protected, you must supply the address of the update or higher-level password in the ACB PASSWD parameter.
NSR	Nonshared resources.
LSR	Local shared resources. Each address space can have up to 256 index resource pools and 256 data resource pools independent of other address spaces. Unless you are using the default, SHRPOOL=0, you must specify the SHRPOOL parameter to indicate which resource pool you are using. Specifying LSR causes a data set to use the local resource pool built by the BLDVRP macro. If an index resource pool exists at the time an OPEN macro is issued, the index for a KSDS is connected to the index resource pool. This parameter is invalid for UNIX files and if it is specified, results in an open failure.
GSR	Global shared resources; all address spaces can have local and global resources pools, and each task in an address space with a local resource pool can use either the local resource pool or the global resource pool. This parameter is invalid for UNIX files and if it is specified, results in an open failure. This parameter is invalid for compressed format data sets.
RLS	While a data set can be accessed using both VSAM RLS and DFSMStvs simultaneously, it is not possible to access a data set simultaneously with both VSAM RLS or DFSMStvs protocols and NSR/LSR/GSR protocols. VSAM enforces this restriction. VSAM RLS and DFSMStvs imply that VSAM uses cross-system record-level locking as opposed to CI locking, uses CF for buffer consistency, and manages a system-wide local cache. Both VSAM RLS and DFSMStvs do not support the following data set features: • Linear data sets • ADR access to a KSDS • CNV access to any data set organization • Data sets defined with imbedded indexes This parameter is invalid for UNIX files and if it is specified, results in an open failure.

Table 10. MACRF options (continued). MACRF options

Option	Meaning
NUB	Management of I/O buffers is left up to VSAM. For VSAM RLS and DFSMStvs, you must specify NUB.
UBF	Management of I/O buffers is left up to the user. The work area specified by the RPL (or GENCB) AREA parameter is the I/O buffer. VSAM transmits the contents of a control interval directly between the work area and direct access storage. **UBF** is valid when OPTCD=MVE and MACRF=CNV are specified. When ICI is specified, UBF is assumed. For VSAM RLS and DFSMStvs, UBF is invalid.

MAREA=*address*
Specifies the address of an optional OPEN/CLOSE or TYPE=T option (CLOSE macro) message area.MAREA is ignored for VSAM RLS and DFSMStvs.

MLEN=*abs expression*
Specifies the length of an optional OPEN/CLOSE or TYPE=T option (CLOSE macro) message area. The default is 0. The maximum length is 32KB. MLEN is ignored for RLS.

PASSWD=*address*
Specifies the address of a field containing the highest-level password required for the types of access indicated by the MACRF parameter. The first byte of the field pointed to contains the length (in binary) of the password (maximum of 8 bytes). Zero indicates that no password is supplied. If the data set is password protected and you do not supply a required password in the access method control block, VSAM gives the console operator the opportunity to supply it when you open the data set.

Data sets that are opened for RLS processing must be SMS-managed data sets that are cataloged and have password processing set to be ignored.

This parameter has no effect for UNIX files.

RLSREAD={NRI|CR|CRE|NORD}
Specifies the read integrity option that applies to GET requests that are issued against this ACB. This parameter overrides the read integrity option that is specified in the RLS JCL parameter. You can override the RLSREAD parameter for a specific GET request by specifying the read integrety option in the RPL OPTCD parameter.

For DFSMStvs, you can specify CRE. If you use CRE for DFSMStvs access, specify CRE in the JCL or the ACB. Those requests that do not require CRE can be overridden by the value that is specified in the RPL.

NRI
Specifies no read integrity. NRI is a performance option. When you specify NRI, VSAM does not obtain a lock on the record.

CR Specifies consistent read integrity. CR ensures that only records that have been committed are read.

CRE
Specifies consistent read explicit. It can only be specified with MACRF=RLS. When it is specified, DFSMStvs access is used. This locking allows the application to inhibit update or erase of the record by other transactions or applications until commit or backout.

NORD
Specifies that the read integrity option that is used is determined either by the RLS JCL specification or by options that are specified on the GET request.

For access modes other than RLS or DFSMStvs, RLSREAD is ignored.

RMODE31=[ALL|BUFF|CB|NONE]
Specifies where VSAM OPEN obtains virtual storage (above or below 16 megabytes) for control blocks and I/O buffers.

The values specified by the RMODE31 parameter have an effect on VSAM only at the setting just before an OPEN is issued. At all other times, changing these values has no effect on the residency of the control blocks and I/O buffers.

If MACRF=RLS is specified, RMODE31=ALL is assumed. For RLS and DFSMStvs, VSAM control blocks and buffers are located in a data space owned by the SMSVSAM server address space and are not directly addressable.

RMODE31= can also be specified in the JCL AMP parameter.

ALL
Specifies that both VSAM control blocks and I/O buffers are obtained above 16 megabytes.

BUFF
Specifies that only VSAM I/O buffers are obtained above 16 megabytes.

CB Specifies that only VSAM control blocks are obtained above 16 megabytes.

NONE
Specifies that both I/O buffers and VSAM control blocks are built below 16 megabytes. This is the default.

SHRPOOL={*abs expression*|0}
Specifies which LSR pool is connected to the ACB. This parameter is valid only when MACRF=LSR is also specified. SHRPOOL must be a number between 0 and 255. The default is 0.

STRNO=*abs expression*
Specifies the number of requests requiring concurrent data set positioning that VSAM is prepared to handle. STRNO must be a number between 1 and 255. The default is 1. A request is defined by a given request parameter list or chain of request parameter lists. The string number is equal to the number of requests issued concurrently for all the data sets sharing the resource pool. See "RPL—generate a request parameter list at assembly time" on page 173 and "GENCB—generate a request parameter list at execution time" on page 159 for information on request parameter lists. When records are loaded into an empty data set, the STRNO value in the access method control block must be 1.

VSAM dynamically extends the number of strings as they are needed by concurrent requests for this ACB. This automatic extension can influence performance. The VSAM control blocks for the set of strings specified by STRNO are allocated on contiguous virtual storage, but this is not guaranteed for the strings allocated by dynamic extension. Dynamic string addition cannot be done when using the following options:

- Load mode
- ICI
- LSR or GSR.

For STRNO, you should specify the total number of request parameter lists or chains of request parameter lists that you are using to define requests. (VSAM needs to remember only one position for a chain of request parameter lists.) However, each position beyond the minimum number that VSAM needs to be able to remember requires additional virtual storage space for these parameters:

- A minimum of one data I/O buffer and, for keyed access, one index I/O buffer (the size of an I/O buffer is the control interval size of a data set)
- Internal control blocks and other areas

For RLS, STRNO is ignored. Strings are dynamically acquired up to a limit of 1024.

STRNO >1 is not supported for UNIX files. If you specify a value greater than 1, OPEN fails.

SUBSYSNM=*address*
For VSAM RLS and DFSMStvs, specifies the address of the subsystem name (mapped by IFGSYSNM and unique to the Parallel Sysplex). This parameter specifies a commit protocol application, which supports online transaction processing subsystems. When SUBSYSNM is specified, you must specify LUWID for all RPL requests against the ACB. The subsystem name is valid only for RLS or DFSMStvs processing; otherwise, it is ignored.

SUBSYSNM is used by CICS to specify a unique name for the CICS region. This name and the RPL LUWID are used by RLS to form the lock owner name for record locks obtained for CICS transactions.

CTRLACB={YES |NO}
For RLS, specifies whether the opened ACB is to be used as a control ACB.

This facility is used by commit protocol applications (for example, CICS) for certain record management requests and in support of an RLS sphere quiesce.

YES
Specifies that the ACB is to be used as a control ACB. The ACB cannot specify a DDNAME, but SUBSYSNM must be specified. The control ACB is also required to have an associated EXLST that specifies a QUIESCE exit. For applications that use DFSMStvs, YES should not be specified.

NO Specifies the ACB is *not* to be used as a control ACB. The default is NO.

CTRLACB is ignored if RLS processing is not specified.

Example 1: ACB macro

In this example, the ACB macro is used to identify a data set to be opened and to specify the types of processing to be performed. The access method control block generated by this example is built when the program is assembled.

```
BLOCK    ACB   AM=VSAM,BUFND=4,          BLOCK gives symbolic          x
               BUFNI=3,                  address of the access         x
               BUFSP=19456,              method control block.         x
               DDNAME=DATASETS,                                        x
               EXLST=EXITS,                                            x
               MACRF=(KEY,DIR,SEQ,OUT),                                x
               STRNO=2
```

The parameters of the ACB macro follow:

- BUFND specifies four I/O buffers for data. BUFNI specifies three I/O buffers for index entries. BUFSP specifies 19456 bytes of buffer space, enough space to accommodate control intervals of data that are 4096 bytes and control intervals of index entries that are 1024 bytes.

- DDNAME specifies this access method control block is associated with a DD statement named DATASETS.
- EXLST specifies the exit list associated with this access method control block is named EXITS.
- MACRF specifies keyed-direct and keyed-sequential processing for both insertion and update.
- STRNO specifies two requests will require concurrent positioning.
- Since the type of resources are not specified, NSR is assumed.

Example 2: ACB macro

In this example, the ACB macro is used to identify a data set to be opened and to specify the types of processing to be performed. The access method control block generated by this example is built when the program is assembled. The caller requests that the VSAM control blocks and I/O buffers be obtained above 16 megabytes, if possible.

```
BLOCK2   ACB   AM=VSAM,                  BLOCK2 gives symbolic        x
               DDNAME=DATASETS,          address of the access        x
               EXLST=EXITS,              method control block.        x
               MACRF=(KEY,DIR,SEQ,OUT),                               x
               RMODE31=ALL
```

The ACB macro's parameters follow:

- DDNAME specifies that this access method control block is associated with a DD statement named DATASETS.
- EXLST specifies that the exit list associated with this access method control block is named EXITS.
- MACRF specifies keyed-direct and keyed-sequential processing for both insertion and update.
- RMODE31=ALL specifies that both VSAM control blocks and buffers can reside above 16 megabytes.
- Because the type of resources is not specified, NSR is assumed.

EXLST—generate an exit list at assembly time

Use the EXLST macro to generate an exit list at assembly time. Values for EXLST macro subparameters can be specified as absolute numeric expressions, character strings, codes, and expressions that generate valid relocatable A-type address constants.

See *z/OS DFSMS Using Data Sets* for the factors that determine the addressing mode and the parameter list residency mode set when the exit routine gets control.

EXLST macro syntax

The format of the EXLST macro follows.

The format of the EXLST macro.

Label	Operand	Parameters
[*label*]	**EXLST**	**[AM= VSAM]** **[,EODAD=(***address***[,A | N][,L])]** **[,JRNAD=(***address***[,A | N][,L])]** **[,LERAD=(***address***[,A | N][,L])]** **[,SYNAD=(***address***[,A | N][,L])]** **[,UPAD=(***address***[,A | N][,L])]** **[,QUIESCE=(***address***[,A | N][,L])]** **[RLSWAIT=(***address***[,A | N][,L])]**

label
Specifies 1 to 8 characters that provide a symbolic address for the established exit list.

AM=VSAM
Specifies that the access method using the control block is VSAM.

EODAD=(*address***[,A|N][,L])**
JRNAD=(*address***[,A|N][,L])**
LERAD=(*address***[,A|N][,L])**
SYNAD=(*address***[,A|N][,L])**
UPAD=(*address***[,A|N][,L])**

RLSWAIT=(*address***[,A|N][,L])**
specify that you are supplying a routine for the exit specified.

For more information about user exit routines, see *z/OS DFSMS Using Data Sets*.

The exits and values that can be specified for these routines are:

EODAD
Specifies that an exit is provided for special processing when the end of a data set is reached by sequential access.

JRNAD
Specifies that an exit is provided for journalizing transactions as you process data records. For VSAM RLS or DFSMStvs, JRNAD is not supported and you receive an error if you open the ACB. This parameter has no effect for UNIX files.

LERAD
Specifies that an exit is provided for analyzing logical errors.

SYNAD
Specifies that an exit is provided for analyzing physical errors.

UPAD
Specifies that an exit is provided for user processing during a VSAM request. The GENCB, MODCB, SHOWCB, and TESTCB macros do not support the UPAD user exit routine. For VSAM RLS and DFSMStvs, UPAD is ignored and the RLSWAIT exit is used instead. This parameter has no effect for UNIX files.

QUIESCE
If RLS is specified, this exit is used to notify the application of a quiesce condition. The exit must be associated with a control ACB and is required for commit protocol applications. If the QUIESCE exit is associated with an ACB other than a control ACB, the exit is ignored.

This exit is entered in 31 bit mode.

The QUIESCE exit is always considered to be active. The A|N parameters are ignored for the QUIESCE exit.

RLSWAIT
For VSAM RLS and DFSMStvs, this exit is used instead of UPAD. If you specify a UPAD exit for RLS or DFSMStvs, it is ignored. The RLSWAIT exit is specified on an ACB basis and is entered in 31–bit mode. When the exit is to be used for a record management request the RPL must specify OPTCD=(SYN,WAITX). The RLSWAIT exit is entered after an asynchronous execution unit is scheduled to process the request. The exit is

intended for those applications which issue VSAM RLS requests and can not tolerate VSAM suspending the execution unit which issued the record management request.

address
Specifies the address of a user-supplied exit routine or an I/O prevention identifier. The address must immediately follow the equal sign.

A|N
Specifies that the exit routine is active (A) or not active (N). VSAM does not enter a routine whose exit is marked not active.

L Specifies that the address is an 8-byte field containing the name of an exit routine in a partitioned data set identified by a JOBLIB or STEPLIB DD statement or in SYS1.LINKLIB. VSAM loads the exit routine for exit processing. If **L** is omitted, the address gives the entry point of the exit routine in virtual storage, and the exit routine is entered in the addressing mode of the VSAM caller. except for the QUIESCE exit.

Requirement: The EXLST macro generates an exit list with each entry 5 bytes in length. You must consider the proper alignment of any subsequent data.

Example: EXLST macro

An EXLST macro is used to identify exit routines provided for analyzing logical and physical errors. The label of the EXLST macro (EXITS) is used in an ACB or GENCB macro that generates an access method control block to associate the exit list with an access method control block. The exit list generated by this example is built when the program is assembled.

```
EXITS    EXLST EODAD=(ENDUP,N),        EXITS gives symbolic address     x
               LERAD=LOGICAL,          of the exit list.                x
               SYNAD=(ROUTNAME,L)
ENDUP                                  EODAD routine.
LOGICAL                                LERAD routine.
ROUTNAME DC    C'PHYSICAL'             Pad shorter names with           x
                                       blanks:C'SYN     ' or CL8'SYN'.
```

The parameters of the EXLST macro follow:

- EODAD specifies that the end-of-data routine is located at ENDUP and is not active.
- LERAD specifies that the logical error routine is located at LOGICAL and is active.
- SYNAD specifies that the physical error routine's name is located at ROUTNAME.

GENCB—generate an access method control block at execution time

The format of the GENCB macro used to generate an access method control block follows.

The format of the GENCB macro used to generate an access method control block.

Label	Operand	Parameters
[*label*]	**GENCB**	**BLK=ACB** **[,AM=VSAM]** **[,BSTRNO=***abs expression***]** **[,BUFND=***abs expression***]** **[,BUFNI=***abs expression***]** **[,BUFSP=***abs expression***]** **[,COPIES=***abs expression***]** **[,DDNAME=***character string***]** **[,EXLST=** *address***]** **[,LENGTH=***abs expression***]** **[,LOC=BELOW | ANY]** **[,MACRF=([ADR][,CNV] [,KEY]** **[,CFX | NFX]** **[,DDN | DSN]** **[,DFR | NDF]** **[,DIR][,SEQ][,SKP]** **[,ICI | NCI]** **[,IN][,OUT]** **[,LEW | NLW]** **[,NIS | SIS]** **[,NRM | AIX]** **[,NRS | RST]** **[,NSR | LSR | GSR | RLS]** **[,NUB | UBF])]** **[,MAREA=***address***]** **[,MLEN=***abs expression***]** **[,PASSWD=***address***]** **[,RMODE31={ALL | BUFF | CB | NONE}]** **[,SHRPOOL={0 |** *abs expression***}]** **[,STRNO=***abs expression***]** **[,SUBSYSNM=***address***]** **[,CTRLACB={YES | NO}]** **[,RLSREAD={NRI | CR | CRE | NORD}]** **[,WAREA=***address***]**

The subparameters of the GENCB macro can be expressed as absolute numeric expressions, as character strings, as codes, as expressions that generate valid relocatable A-type address constants, in register notation, as S-type address constants, and as indirect S-type address constants. "Subparameters with GENCB, MODCB, SHOWCB, and TESTCB" on page 131, further defines these operand expressions.

label
Specifies 1 to 8 characters that provide a symbolic address for the GENCB macro.

BLK=ACB
Specifies that you are generating an access method control block.

AM=VSAM
Specifies that the access method using this control block is VSAM.

BSTRNO=*abs expression*
Specifies the number of strings initially allocated for access to the base cluster of a path. BSTRNO must be a number between 0 and 255. The default is STRNO. BSTRNO is ignored if the object being opened is not a path. If the number specified for BSTRNO is insufficient, VSAM dynamically extends the

number of strings as needed for the access to the base cluster. BSTRNO can also influence performance. The VSAM control blocks for the set of strings specified by BSTRNO are allocated on contiguous virtual storage, whereas this is not guaranteed for the strings allocated by dynamic extension.

For VSAM RLS and DFSMStvs, BSTRNO is ignored. This parameter has no effect for UNIX files.

BUFND=*abs expression*
Specifies the number of I/O buffers VSAM uses for transmitting data between virtual and auxiliary storage. A buffer is the size of a control interval in the data component. BUFND must be a number between 0 and 65535. The minimum number you may specify is 1 plus the number specified for STRNO (if you omit STRNO, BUFND must be at least 2 because the default for STRNO is 1). The number can be supplied through the JCL DD AMP parameter and through the macro. The default is the minimum number required. A larger number for BUFND can improve the performance of sequential access.

For VSAM RLS and DFSMStvs, BUFND is ignored. This parameter has no effect for UNIX files.

BUFNI=*abs expression*
Specifies the number of I/O buffers VSAM uses for transmitting index entries between virtual and auxiliary storage for keyed access. A buffer is the size of a control interval in the index. BUFNI must be a number between 0 and 65535. The minimum number is the number specified for STRNO (if you omit STRNO, BUFNI must be at least 1 because the default for STRNO is 1). You can supply the number through the JCL DD AMP parameter and through the macro. The default is the minimum number required. A larger number for BUFNI can improve the performance of keyed-direct retrieval.

For VSAM RLS and DFSMStvs, BUFNI is ignored. This parameter has no effect for UNIX files.

BUFSP=*abs expression*
Specifies the maximum number of bytes of virtual storage used for the data and index I/O buffers. VSAM gets the storage in your program's address space. If you specify less than the amount of space specified in the BUFFERSPACE parameter of the DEFINE command when the data set was defined, VSAM overrides your BUFSP specification upward to the value specified in BUFFERSPACE. (BUFFERSPACE, by definition, is the least amount of virtual storage that is ever provided for I/O buffers.) You can supply BUFSP through the JCL DD AMP parameter and through the macro. If you do not specify BUFSP in either place, the amount of storage used for buffer allocation is the *largest* of:

- The amount specified in the catalog (BUFFERSPACE),
- The amount determined from BUFND and BUFNI, or
- The minimum storage required to process the data set with its specified processing options.

If BUFSP is specified and the amount is smaller than the minimum amount of storage required to process the data set, VSAM cannot open the data set.

A valid BUFSP amount takes precedence over the amount called for by BUFND and BUFNI. If the BUFSP amount is greater than the amount called for by BUFND and BUFNI, the extra space is allocated as follows:

- When MACRF indicates direct access only, additional index buffers are allocated.

- When MACRF indicates sequential access, one additional index buffer and as many data buffers as possible are allocated.

If the BUFSP amount is less than the amount called for by BUFND and BUFNI, the number of data and index buffers is decreased as follows:

- When MACRF indicates direct access only, the number of data buffers is decreased to not less than the minimum number. Then, if required, the number of index buffers is decreased until the amount called for by BUFND and BUFNI complies with the BUFSP amount.
- When MACRF indicates sequential access, the number of index buffers is decreased to not less than 1 more than the minimum number. Then, if required, the number of data buffers is decreased to not less than the minimum number. If still required, 1 more is subtracted from the number of index buffers.
- Neither the number of data buffers nor the number of index buffers is decreased to less than the minimum number.

If the index does not exist or is not being opened, only BUFND, and not BUFNI, enters into these calculations.

For VSAM RLS and DFSMStvs, BUFSP is ignored. This parameter has no effect for UNIX files.

COPIES=*abs expression*
Specifies the number of copies of the access method control block VSAM generates. All the copies are identical. Use MODCB to tailor the individual copies for particular data sets and processing. MODCB is described in "MODCB—modify an access method control block" on page 168.

DDNAME=*character string*
Specifies 1 to 8 characters that identify the data set you want to process by specifying the JCL DD statement for the data set. You may omit DDNAME and provide it through the MODCB macro before opening the data set. MODCB is described in "MODCB—modify an access method control block" on page 168.

EXLST=*address*
Specifies the address of a list of addresses of exit routines you are providing. The list is established by the EXLST or GENCB macro. If you use the EXLST macro, you can specify its label here as the address of the exit list. If you use GENCB, you can specify the address returned by GENCB in register 1. Omitting this parameter indicates that you have no exit routines. VSAM user exit routines are described in *z/OS DFSMS Using Data Sets*.

LENGTH=*abs expression*
Specifies the length, in bytes, of the area, if any, you are supplying for VSAM to generate the access method control blocks. (See the WAREA parameter.) The LENGTH value cannot exceed 65535 (X'FFFF').

LOC={BELOW|ANY}

BELOW
Specifies that VSAM is to construct an ACB in an area of virtual storage below 16 megabytes at execution time. This is the default.

ANY
Specifies that VSAM is to construct an ACB in an area of virtual storage above 16 megabytes, if possible, at execution time.

MACRF=([ADR][,CNV][,KEY]
[,CFX|NFX]
[,DDN|DSN]

[,DFR|NDF]
[,DIR][,SEQ][,SKP]
[,ICI|NCI]
[,IN][,OUT]
[,LEW|NLW]
[,NIS|SIS]
[,NRM|AIX]
[,NRS|RST]
[,NSR|LSR|GSR|RLS]
[,NUB|UBF])

Specifies the kinds of processing you will do with the data set. The subparameters must be significant for the data set. For example, if you specify keyed access for an entry-sequenced data set, you cannot open the data set. You must specify all the types of access you are going to use, whether you use them concurrently or by switching from one to the other. The subparameters are shown in Table 10 on page 139. They are arranged in groups, and each group has a default value (shown by underlining). You may specify subparameters in any order. You may specify both ADR and KEY to process a key-sequenced data set. You may specify both DIR and SEQ; with keyed access, you may specify SKP as well. If you specify OUT and want merely to retrieve some records and also update, delete, or insert others, you need not also specify IN.

MAREA=*address*

Specifies the address of an optional OPEN/CLOSE or TYPE=T option (CLOSE macro) message area.

MAREA is ignored for VSAM RLS and DFSMStvs processing.

MLEN=*abs expression*

Specifies the length of an optional OPEN/CLOSE or TYPE=T option (CLOSE macro) message area.

MLEN is ignored for VSAM RLS and DFSMStvs processing.

PASSWD=*address*

Specifies the address of a field that contains the highest-level password required for the types of access indicated by the MACRF parameter. The first byte of the field contains the length (in binary) of the password (maximum of 8 bytes). Zero indicates that no password is supplied. If the data set is password protected and you do not supply a required password in the access method control block, VSAM may give the console operator the opportunity to supply it when you open the data set. This parameter has no effect for UNIX files.

For VSAM RLS and DFSMStvs, data sets that are opened must be SMS data sets that are cataloged.

RMODE31={ALL|BUFF|CB|NONE}

Specifies where VSAM OPEN is to obtain virtual storage (above or below 16 megabytes) for control blocks and I/O buffers.

The values specified by the RMODE31 parameter only have an effect on VSAM at the setting just before an OPEN is issued. At all other times, changing these values has no effect on the residency of the control blocks and I/O buffers.

The virtual storage location of the ACB is independent of the RMODE31 parameter. An ACB may reside either above or below 16 megabytes.

For VSAM RLS and DFSMStvs, VSAM control blocks and buffers are located in a dataspace owned by the SMSVSAM server address space and are not directly addressable.

RMODE31=ALL is assumed for VSAM RLS and DFSMStvs processing.

ALL
Specifies both VSAM control blocks and I/O buffers are obtained above 16 megabytes.

BUFF
Specifies only VSAM I/O buffers are obtained above 16 megabytes.

CB Specifies only VSAM control blocks are obtained above 16 megabytes.

NONE
Specifies both VSAM control blocks and I/O buffers are obtained below 16 megabytes. This is the default.

SHRPOOL={*abs expression***|0}**
Specifies the identification number of the resource pool used for LSR processing. SHRPOOL must be a number between 0 and 255. The default is SHRPOOL=0. For VSAM RLS and DFSMStvs, SHRPOOL is ignored. This parameter has no effect for UNIX files.

STRNO=*abs expression*
Specifies the number of requests requiring concurrent data set positioning VSAM is prepared to handle. A request is defined by a given request parameter list or chain of request parameter lists. STRNO must be a number between 1 and 255. See "RPL—generate a request parameter list at assembly time" on page 173 and "GENCB—generate a request parameter list at execution time" on page 159 for information on request parameter lists.

For VSAM RLS and DFSMStvs, STRNO is ignored and strings are dynamically acquired up to a limit of 1024. STRNO > 1 is not supported for UNIX files and if it is specified with a value greater than 1, results in an open failure.

SUBSYSNM=*address*
For VSAM RLS, specifies the address of the subsystem name (mapped by IFGSYSNM and unique to the Parallel Sysplex). This parameter specifies a commit protocol application, which supports online transaction processing subsystems. When SUBSYSNM is specified, you must specify LUWID for all RPL requests against the ACB. The susbsystem name is valid only for RLS or DFSMStvs processing; otherwise, it is ignored. For DFSMStvs, this parameter is filled in automatically.

SUBSYSNM is used by CICS to specify a unique name for the CICS region. This name and the RPL LUWID are used by RLS to form the lock owner name for record locks obtained for CICS transactions.

RLSREAD={NRI|CR|CRE|NORD}
For VSAM RLS and DFSMStvs, specifies the read integrity options that apply to GET requests issued against this ACB. This parameter overrides the read integrity options specified in the RLS JCL parameter. Read integrity options can also be specified on the GET request, when they override the RLSREAD specification.

For DFSMStvs, you can specify CRE. Read integrity options can also be specified on the GET request, in which case they override the RLSREAD specification, assuming that the parameter specified is compatible with the type of access to the data set. For example, if the data set were open for RLS

access, rather than DFSMStvs access, it would not be valid to specify CRE later in the RPL. If you want to use CRE for DFSMStvs access, specify CRE in the JCL or the ACB.

Those requests that do not require CRE can be overridden by the value specified in the RPL.

NRI
Specifies no read integrity. NRI is a performance option. When you specify NRI, VSAM does not obtain a lock on the record.

CR Specifies consistent read integrity. CR also ensures that only records that have been committed are read.

CRE
Specifies consistent read explicit integrity. CRE can only be specified when MACRF=RLS. When CRE is specified, DFSMStvs access is used. This locking enables the application to inhibit update or erase of the record by other transactions or applications until commit or backout.

NORD
species the read integrity option used is determined either by the RLS JCL specification or by options specified on the GET request.

For access modes other than RLS or DFSMStvs, , this parameter is ignored.

CTRLACB={YES |NO}
For VSAM RLS and DFSMStvs, specifies if the opened ACB is to be used as a control ACB.

This facility is used by commit protocol applications for certain record management requests and in support of an RLS sphere quiesce.

YES
Specifies the ACB is to be used as a control ACB. The ACB can not specify a DDNAME but SUBSYSNM must be specified. The control ACB is also required to have an associated EXLST that specifies a QUIESCE exit.

YES should never be specified by an application using DFSMStvs.

NO Specifies the ACB is *not* to be used as a control ACB. The default is NO.

If RLS is not specified, CTRLACB is ignored.

WAREA=*address*
Specifies the address of an area in which to generate the access method control blocks.

The area must begin on a fullword boundary.

This parameter is paired with the LENGTH parameter. You must supply the LENGTH parameter if you specify an area address.

If you do not specify an area in which the access method control block is to be generated, VSAM obtains virtual storage space for the area (as specified by the LOC=keyword). Subpool 0 will be requested under the user's key and state. Users executing in key 0 and supervisor state will actually be assigned subpool 252. VSAM returns the address of the area containing the control blocks in register 1 and the length of the area in register 0. You can find out the length of each control block by dividing the length of the area by the number of copies. The address of each control block can then be calculated by this offset from the address in register 1. You can find the length of an access method control block with the SHOWCB macro.

If you are generating control blocks by issuing several GENCBs, specifying an area (WAREA and LENGTH parameters) for them enbles you to address all of them with one base register and to avoid repetitive requests for virtual storage.

Example 1: GENCB macro (generate an access method control block)

In this example, a GENCB macro is used to identify a data set to open and to specify the types of processing to perform. This example specifies that the space for the control block be obtained above 16 megabytes. The access method control block generated by this example is built when the program is executed.

```
GENCB  GENCB   BLK=ACB,AM=VSAM,     One copy generated; VSAM gets the   x
               BUFND=4,BUFNI=3,     storage for it because the          x
               BUFSP=19456,         WAREA LENGTH parameters have        x
               DDNAME=DATASETS,     been omitted.                       x
               EXLST=EXITS,                                             x
               LOC=ANY,                                                 x
               MACRF=(KEY,DIR,                                          x
               SEQ,OUT),                                                x
               RMODE31=ALL,                                             x
               STRNO=2

         ST    1,ACBADDR           Save the address of the access      x
                                   method control block.
ACBADDR  DS    A                   The address of the access method    x
                                   control block is saved in ACBADDR.
```

The parameters of the GENCB macro follow:

- BUFND specifies four I/O buffers for data. BUFNI specifies three I/O buffers for index entries. BUFSP specifies 19456 bytes of buffer space, enough space to accommodate control intervals of data that are 4096 bytes and of index entries that are 1024 bytes.
- DDNAME specifies that this access method control block is associated with a DD statement named DATASETS.
- EXLST specifies that the exit list associated with this access method control block is named EXITS.
- LOC specifies that VSAM obtain virtual storage for the ACB from an area that may be above 16 megabytes.
- MACRF specifies keyed direct and keyed sequential processing for both insertion and update.
- RMODE31 specifies that VSAM obtain storage for the VSAM control blocks and I/O buffers in an area above 16 megabytes when the ACB is opened.
- STRNO specifies that two requests will require concurrent positioning.

Example 2: GENCB macro (generate an access method control block)

The access method control block (ACB) generated by this example is built when the program is executed. In this example, the user provides the storage to contain the ACB. Because the generate form of the macro is used, the GENCB parameter list is built in a remote area and passed to VSAM for action.

```
         LA      10,LEN1            Get length of the GENCB parameter
                                    list returned by the GENCB macro.
         GETMAIN R,LV=(10)          Get storage for the area in which
                                    the GENCB parameter list is to
                                    be built.
         LR      2,1                Save addr of GENCB parameter-list
                                    area.
         LA      10,ACBLNGTH        Get length of the ACB.
```

```
         GETMAIN R,LV=(10)          Get storage for the area in which
                                    the ACB is to be built.
         LR      3,1                Save address of ACB area.

GENCB1   GENCB   BLK=ACB,AM=VSAM,    One copy generated; VSAM builds    x
                 BUFND=4,BUFNI=3,    the ACB in the storage provided    x
                 BUFSP=19456,        at the location pointed to by      x
                 DDNAME=DATASETS,    WAREA.                             x
                 LENGTH=ACBLNGTH,                                       x
                 MACRF=(KEY,DIR,                                        x
                 SEQ,OUT),                                              x
                 RMODE31=ALL,                                           x
                 WAREA=(3),                                             x
                 MF=(G,(2),LEN1)
                 .
                 .
                 .
ANYNAME  DSECT   KEEP ACB model out of CSECT
ACBSTART ACB     AM=VSAM
ACBEND   DS      0F
ACBLNGTH EQU     ACBEND-ACBSTART
```

The parameters of the GENCB macro follow:

- BUFND specifies four I/O buffers for data. BUFNI specifies three I/O buffers for index entries. BUFSP specifies 19456 bytes of buffer space, enough space to accommodate control intervals of data that are 4096 bytes and of index entries that are 1024 bytes.
- DDNAME specifies that this access method control block is associated with a DD statement named DATASETS.
- LENGTH specifies that the length of the storage you provide for the ACB is the value of ACBLNGTH.
- MACRF specifies keyed direct and keyed sequential processing for both insertion and update.
- RMODE31 specifies that VSAM obtain storage for the VSAM control blocks and I/O buffers in an area above 16 megabytes when the ACB is opened.
- WAREA specifies that the address of the storage you provide for the ACB is held in register 3.
- MF specifies that the GENCB parameter list is to be built in the location specified by register 2. Also, the expansion of the GENCB macro will equate LEN1 to the length of the GENCB parameter list.

GENCB—generate an exit list at execution time

The format of the GENCB macro used to generate an exit list follows.

The format of the GENCB macro used to generate an exit list.

Label	Operand	Parameters
[*label*]	**GENCB**	**BLK=EXLST** **[,AM=VSAM]** **[,COPIES=***abs expression*] **[,EODAD=(***address*[,A | N][,L])] **[,JRNAD=(***address*[,A | N][,L])] **[,LENGTH=***abs expression*] **[,LERAD=(***address*[,A | N][,L])] **[,LOC=BELOW	ANY]** **[,SYNAD=(***address*[,A | N][,L])] **[,QUIESCE=(***address*[,A | N][,L])] **[,RLSWAIT=(***address*[,A | N][,L])] **[,WAREA=***address*]

The subparameters of the GENCB macro can be expressed as absolute numeric expressions, as character strings, as codes, as expressions that generate valid relocatable A-type address constants, in register notation, as S-type address constants, and as indirect S-type address constants. "Subparameters with GENCB, MODCB, SHOWCB, and TESTCB" on page 131, further defines these operand expressions.

For the factors that determine the addressing mode and the parameter list residency mode set when the exit routine gets control, see *z/OS DFSMS Using Data Sets*.

label
Specifies 1 to 8 characters that provide a symbolic address for the GENCB macro.

BLK=EXLST
Specifies that you are generating an exit list.

AM=VSAM
Specifies that the access method using this control block is VSAM.

[,EODAD=(*address***[,A|N][,L])]**
[,JRNAD=(*address***[,A|N][,L])]**
[,LERAD=(*address***[,A|N][,L])]**
[,SYNAD=(*address***[,A|N][,L])]**
[,RLSWAIT=(*address***[,A|N][,L])]**
Specifies that you are supplying a routine for the exit named.

For more information about user exit routines, see *z/OS DFSMS Using Data Sets*.

If none of these user exit routines is specified, VSAM generates an exit list with inactive entries for all the exits. The exits and values that can be specified for them are:

COPIES=*abs expression*
Specifies the number of copies of the exit list you want generated. GENCB generates as many copies as you specify (default is 1) when your program is executed. All copies are the same. You can use MODCB to change some or all of the addresses in a list. MODCB is described in "MODCB—modify an access method control block" on page 168.

EODAD
Specifies that an exit is provided for special processing when the end of a data set is reached by sequential access.

JRNAD
Specifies that an exit is provided for journaling as you process data records. For VSAM RLS and DFSMStvs, JRNAD is not supported and you receive an error if you open the ACB. This parameter has no effect for UNIX files.

LERAD
Specifies that an exit is provided for analyzing logical errors.

SYNAD
Specifies that an exit is provided for analyzing physical errors.

QUIESCE
Specifies that an exit is provided for quiescing RLS activity across the Parallel Sysplex.

RLSWAIT
Specifies that an exit is provided for wait processing. For VSAM RLS and DFSMStvs, the UPAD exit is ignored if it is specified, and the RLSWAIT exit is used to perform a similar function.

address
Specifies the address of a user-supplied exit routine. The address must immediately follow the equal sign.

A|N
Specifies that the exit routine is active (A) or not active (N). VSAM does not enter a routine whose exit is marked not active.

L Specifies the address is an 8-byte field containing the name of an exit routine in a partitioned data set identified by a JOBLIB or STEPLIB DD statement or in SYS1.LINKLIB. VSAM is to load the exit routine for exit processing. If **L** is omitted, the address gives the entry point of the exit routine in virtual storage, and the exit routine is entered in the addressing mode of the VSAM caller. except for the QUIESCE exit.

L might precede or follow the **A** or **N** specification.

LENGTH=*abs expression*
Specifies the length, in bytes, of the area, if any, that you are supplying for VSAM to generate the exit lists. (See the WAREA parameter.) The LENGTH value cannot exceed 65535 (X'FFFF').

LOC=BELOW|ANY

BELOW
Specifies that VSAM is to construct an exit list in an area below 16 megabytes at execution time.

ANY
Specifies that VSAM is to construct an exit list in an area above 16 megabytes, if possible, at execution time.

WAREA=*address*
Specifies the address of an area in which to generate the exit lists.

If you did not specify an area in which the exit list is to be generated, VSAM obtains virtual storage space for the area (as specified by the LOC=keyword). Subpool 0 will be requested under the user's key and state. Users executing in

key 0 and supervisor state will actually be assigned subpool 252. VSAM returns the address of the area in which the exit lists is to be generated in register 1, and the length of the area in register 0. You can find the length of each exit list by dividing the length of the area by the number of copies. The address of each exit list can then be calculated by this offset from the address in register 1. You can find the length of an exit list with the SHOWCB macro, described in *z/OS DFSMS Macro Instructions for Data Sets*.

If you are generating control blocks by issuing several GENCBs, specifying an area (WAREA and LENGTH) for them allows you to address all of them with one base register and to avoid repetitive requests for virtual storage.

Example: GENCB macro (generate an exit list)

In this example, a GENCB macro is used to generate an exit list when the program is executed.

```
EXITS    GENCB BLK=EXLST,                                          x
               EODAD=(EOD,N),                                      x
               LERAD=LOGICAL,                                      x
               SYNAD=(ERROR,                                       x
               A,L)

         LTR   15,15
         BNZ   ERROR
         ST    1,EXLSTADR         Address of the exit list is saved.
EOD      EQU   *                  EODAD routine.
LOGICAL  EQU   *                  LERAD routine.
ERROR    DC    C'PHYSICAL'        Name of the SYNAD module.
EXLSTADR DS    A                  Save area for exit-list address.
```

The GENCB macro's parameters are:

- BLK specifies an exit list is generated.
- EODAD specifies the end-of-data routine is located at EOD and is not active.
- LERAD specifies that the logical error routine is located at LOGICAL. Because neither **A** nor **N** is specified, the LERAD routine is marked active by default.
- SYNAD specifies that the physical error routine's name is located at ERROR.

Because no area is specified in which the exit list is to be generated, VSAM obtains virtual storage for the exit list and returns the address in register 1. Immediately after the GENCB macro, the address of the exit list, contained in register 1, is moved to EXLSTADR. EXLSTADR may be specified in a GENCB macro that generates an access method control block or in a MODCB, SHOWCB, or TESTCB macro that modifies, displays, or tests fields in an exit list.

GENCB—generate a request parameter list at execution time

The format of the GENCB macro used to generate a request parameter list follows.

The format of the GENCB macro used to generate a request parameter list.

Label	Operand	Parameters
[*label*]	**GENCB**	**BLK=RPL** **[,ACB=***address***]** **[,AM=VSAM]** **[,AREA=***address***]** **[,AREALEN=***abs expression***]** **[,ARG=***address***]** **[,COPIES=***abs expression***]** **[,LUWID=***address***][,TIMEOUT=***number***]** **[,ECB=***address***]** **[,KEYLEN=***abs expression***]** **[,LENGTH=***abs expression***]** **[,LOC=BELOW|ANY]** **[,MSGAREA=***address***]** **[,MSGLEN=***abs expression***]** **[,NXTRPL=***address***]** **[,OPTCD=([ADR|CNV|KEY]** **[,DIR|SEQ|SKP]** **[,ARD|LRD]** **[,FWD|BWD]** **[,ASY|SYN]** **[,NSP|NUP|UPD]** **[,KEQ|KGE]** **[,FKS|GEN]** **[,LOC|MVE]** **[,NRI|CR|CRE] [,NRI|CR|CRE [,RBA|XRBA])]** **[,RECLEN=***abs expression***]** **[,TRANSID=***abs expression***]** **[,WAREA=***address***]**

The subparameters of the GENCB macro can be expressed as absolute numeric expressions, as character strings, as codes, as expressions that generate valid relocatable A-type address constants, in register notation, as S-type address constants, and as indirect S-type address constants. “Subparameters with GENCB, MODCB, SHOWCB, and TESTCB” on page 131, further defines these operand expressions.

The parameters of the GENCB macro to generate a request parameter list are optional sometimes, but required in others. It is not necessary to omit parameters that are not required for a request; they are ignored. Thus, if you switch from direct to sequential retrieval with a request parameter list, you do not have to zero out the address of the field containing the search argument (ARG=address).

label
: Specifies 1 to 8 characters that provide a symbolic address for the GENCB macro. For addressing lists generated by GENCB, see the COPIES parameter.

BLK=RPL
: Specifies that you are generating a request parameter list.

ACB=*address*
: Specifies the address of the access method control block that identifies the data set to which access will be requested. If you omit this parameter, you must

issue MODCB to specify the address of the access method control block before you issue a request. MODCB is described in "MODCB—modify an access method control block" on page 168.

AM=VSAM
Specifies that the access method using this control block is VSAM.

AREA=*address*
Specifies the address of a work area to and from which VSAM moves a data record if you request it to do so (with the RPL parameter OPTCD=MVE). If you request that records be processed in the I/O buffer (OPTCD=LOC), VSAM puts into this work area the address of a data record within the I/O buffer.

AREALEN=*abs expression*
Specifies the length, in bytes, of the work area whose address is specified by the AREA parameter. Its minimum for OPTCD=MVE is the size of a data record (or the largest data record, for a data set with records of variable length). For OPTCD=LOC, the area should be 4 bytes to contain the address of a data record within the I/O buffer.

ARG=*address*
Specifies the address of a field containing the search argument for direct retrieval, skip-sequential retrieval, and positioning. For a fixed-length or variable-length RRDS, the ARG field must be 4 bytes long. For direct or skip-sequential processing, this field contains your search argument, a relative record number. For sequential processing (OPTCD=(KEY,SEQ)), the 4 bytes are required for VSAM to return the feedback RRN. For keyed access (OPTCD=KEY), the search argument is a full or generic key. For addressed access (OPTCD=ADR), the search argument is an RBA. If you specify a generic key (OPTCD=GEN), you must also specify in the KEYLEN parameter how many of the bytes of the full key you are using for the generic key.

COPIES=*abs expression*
Specifies the number of copies of the request parameter list to generate. GENCB generates as many copies as you specify (default is 1) when your program is executed.

The copies of a request parameter list can be used to:

- Chain lists together to gain access to many records with one request
- Define many requests to gain access to many parts of a data set concurrently.

All copies generated are identical; you must use MODCB to tailor them to specific requests. MODCB is described in "MODCB—modify an access method control block" on page 168.

ECB=*address*
Specifies the address of an event control block (ECB) that you may supply. VSAM indicates in the ECB whether a request is complete or not (using standard completion codes, which *z/OS MVS System Codes* describes). You can use the ECB to determine that an asynchronous request is complete before issuing a CHECK macro. This parameter is always optional.

KEYLEN=*abs expression*
Specifies the length, in bytes, of the generic key (OPTCD=GEN) you are using for a search argument (given in the field addressed by the ARG parameter). This parameter is required with a search argument that is a generic key. The number can be 1 through 255. For full-key searches, VSAM knows the key length, which is taken from the catalog definition of the data set when you open the data set. This parameter has no effect for UNIX files.

LUWID=*luwid*
For &rls only, *logical unit of work identifier*. The LUWID together with the SUBSYSNM (specified when the ACB is OPENed) forms the ownership for record locks. CICS uses the LUWID parameter to pass a CICS transaction identifier to RLS. A non-CICS application does not specify the LUWID parameter. VSAM assigns a single LUWID for all requests issued by a non-CICS application.

LENGTH=*abs expression*
Specifies the length, in bytes, of the area, if any, that you are supplying for VSAM to generate the request parameter lists. (See the WAREA parameter.) The LENGTH value cannot exceed 65535 (X'FFFF').

You can find out how long a request parameter list is with the SHOWCB macro, described in *z/OS DFSMS Macro Instructions for Data Sets*.

LOC=BELOW|ANY

BELOW
Specifies that storage for the RPL be obtained from virtual storage below 16 megabytes.

ANY
Specifies that storage be obtained from virtual storage above 16 megabytes if possible.

MSGAREA=*address*
Specifies the address of an area you are supplying for VSAM to send you a message if a physical error occurs. The format of a physical error message is given under "Reason code (physical errors)" on page 215 in the topic "Understanding VSAM macro return and reason codes" on page 190.

MSGLEN=*abs expression*
Specifies the size, in bytes, of the message area indicated in the MSGAREA parameter. The size of a message is 128 bytes. If you provide less than 128 bytes, no message is returned to your program. This parameter is required when MSGAREA is coded.

NXTRPL=*address*
Specifies the address of the next request parameter list in a chain. Omit this parameter from the macro that generates the only or last list in the chain. When you issue a request defined by a chain of request parameter lists, indicate in the request macro the address of the first parameter list in the chain. A single request macro can be defined by multiple request parameter lists. For example, a GET can cause VSAM to retrieve two or more records. This parameter has no effect for UNIX files and if it is specified with a non-zero value, results in an error on a subsequent GET, PUT, or POINT.

```
OPTCD=([ADR|CNV|KEY]
  [,DIR|SEQ|SKP]
  [,ARD|LRD]
  [,FWD|BWD]
  [,ASY|SYN]
  [,NSP|NUP|UPD]
  [,KEQ|KGE]
  [,FKS|GEN]
   [,LOC|MVE])
[,UPD[,KL|NOKL]]
[,CR|CRE|NRI]
  [,CR|CRE|NRI]
```

[,RBA|XRBA])
Specifies the subparameters that govern the request defined by the request parameter list. Each group of subparameters has a default; subparameters are shown in Table 11 on page 176 with defaults underlined. Only one subparameter from each group is effective for a request. Some requests do not require an subparameter from all of the groups to be specified. The groups that are not required are ignored. Thus, you can use the same request parameter list for a combination of requests (GET, PUT, POINT, for example) without zeroing out the inapplicable subparameters each time you go from one request to another.

RECLEN=*abs expression*
Specifies the length, in bytes, of a data record being stored. If the records you are storing are all the same length, you do not need to change RECLEN after you set it. This parameter is required for PUT requests. For GET requests, VSAM puts the length of the record retrieved in this field in the request parameter list. It will be there if you update and store the record.

TIMEOUT=*number*
For VSAM RLS or DFSMStvs only, specifies the time in seconds that your program is to wait to obtain a lock on a VSAM record when a lock on the record is already held by another program.

A nonzero value for TIMEOUT specifies the time (in seconds) this program waits for the other program(s) to release the lock.

A value of zero specifies TIMEOUT processing is *not* to be performed by VSAM for this request. That is, if the record lock required by the request is held by another program, the program waits until the other program releases the lock regardless of how long that might be.

TRANSID=*abs expression*
Specifies a number that relates modified buffers in a buffer pool. Use in shared resource applications and a description are in *z/OS DFSMS Using Data Sets*. This parameter has no effect for UNIX files.

For RLS or DFSMStvs, this parameter is ignored. LUWID replaces this function.

WAREA=*address*
Specifies the address of an area in which the request parameter lists are generated.

If you did not specify an area in which the request parameter list is to be generated, VSAM obtains virtual storage space for the area (as specified by the LOC=*keyword*). Subpool 0 will be requested under the user's key and state. Users executing in key 0 and supervisor state will actually be assigned subpool 252. VSAM returns the address of the area in which the request parameter lists are generated in register 1, and the length of the area in register 0. You can find the length of each list by dividing the length of the area by the number of copies. You can then calculate the address of each list by using the length of each list as an offset.

If you are generating control blocks by issuing several GENCBs, specifying an area (WAREA and LENGTH parameters) for them allows you to address all of them with one base register and to avoid repetitive requests for virtual storage.

Building a chain of request parameter lists: When GENCB is used to build a chain of request parameter lists, the request parameter lists may be chained using only GENCB macros or using GENCB and MODCB macros together. When only GENCB is used, the request parameter lists are created in reverse order, as follows:

```
SECOND  GENCB  BLK=RPL
        LR     2,1
FIRST   GENCB  BLK=RPL,NXTRPL=(2)
```

SECOND GENCB creates the second request parameter list, which makes its address available for the first request parameter list. The address of the request parameter list is returned in register 1 and is loaded into register 2. FIRST GENCB creates the first request parameter list and supplies the address of the next request parameter list using register notation. GENCB and MODCB macros may be used together to create a chain of request parameter lists, as follows:

```
      GENCB    BLK=RPL,COPIES=2
      LR       2,0
      SRL      2,1
      LR       3,1
      LA       4,0(2,3)
      MODCB    RPL=(3),NXTRPL=(4)
```

The GENCB macro creates two request parameter lists. The length of the parameter lists is returned in register 0 and loaded into register 2. The address of the area in which the lists were created (and, therefore, the address of the first one) is returned in register 1 and loaded into register 3. The SRL statement divides the total length of the area (register 2) by 2. The LA statement loads the address of the second request parameter list into register 4. The MODCB macro modifies the first request parameter list (register 3) by supplying the address of the second request parameter list (register 4) in the NXTRPL parameter.

Each request parameter list in a chain should have the same OPTCD subparameters. Having different subparameters may cause logical errors. You cannot chain request parameter lists for updating or deleting records—only for retrieving records or storing new records. You cannot process records in the I/O buffer with chained request parameter lists. (OPTCD=UPD and LOC are invalid for chained request parameter lists.)

Example: GENCB macro (generate a request parameter list): In this example, a GENCB macro is used to generate a request parameter list.

```
ACCESS  GENCB  BLK=RPL,                                          x
               ACB=ACCESS,                                       x
               AM=VSAM,                                          x
               AREA=WORK,                                        x
               AREALEN=125,                                      x
               ARG=SEARCH,                                       x
               LOC=ANY,                                          x
               MSGAREA=MESSAGE,                                  x
               MSGLEN=128,                                       x
               OPTCD=(SKP,UPD)

ACCESS  ACB    MACRF=(SKP,OUT)
WORK    DS     CL125
SEARCH  DS     CL8
MESSAGE DS     CL128
```

The GENCB macro's parameters are:

- BLK specifies a request parameter list is generated.
- ACB specifies that the request parameter list is associated with a data set and processing options identified by ACCESS.
- AREA and AREALEN specify a 125-byte work area used for processing records.
- ARG specifies the address of the search argument.

- LOC specifies that VSAM obtain storage for the request parameter list in an area above 16 megabytes.
- MSGAREA and MSGLEN specify a 128-byte area used for physical-error messages.
- OPTCD specifies the subparameters that govern the request defined by the request parameter list identified by SKP and UPD.

Example: GENCB macro (generate a request parameter list): In this example, a GENCB macro is used to generate a request parameter list (RPL). In this example the user provides the storage to contain the RPL. Because the generate form of the macro is used, the GENCB parameter list is built in a remote area and passed to VSAM for action.

```
         LA    10,LEN2              Get length of the GENCB parameter
                                    list returned by the GENCB macro.
         GETMAIN R,LV=(10)          Get storage for the area in which
                                    the GENCB parameter list is to
                                    be built.
         LR    2,1                  Save addr of GENCB parameter-list
                                    area.
GENCB1   GENCB BLK=RPL,             One copy generated; VSAM builds    x
               ACB=ACCESS,          the RPL in the storage provided    x
               AM=VSAM,             at the location pointed to by      x
               AREA=WORK,           WAREA.                             x
               AREALEN=125,                                            x
               ARG=SEARCH,                                             x
               LENGTH=RPLLNGTH,                                        x
               MSGAREA=MESSAGE,                                        x
               MSGLEN=128,                                             x
               OPTCD=(SKP,UPD),                                        x
               WAREA=MYRPL,                                            x
               MF=(G,(2),LEN2)
               .
               .
               .
ACCESS   ACB   MACRF=(SKP,OUT)
WORK     DS    CL125
SEARCH   DS    CL8
MESSAGE  DS    CL128
         DS    0F
MYRPL    DS    CL(RPLLNGTH)         Storage in which the RPL is to be
                                    built.
ANYNAME  DSECT Avoid generation in CSECT
RPLSTART RPL   AM=VSAM
RPLEND   DS    0F
RPLLNGTH EQU   RPLEND-RPLSTART
```

The GENCB macro's parameters are:

- BLK specifies a request parameter list is generated.
- ACB specifies that the request parameter list is associated with a data set and processing options identified by ACCESS.
- AREA and AREALEN specify a 125-byte work area used for processing records.
- ARG specifies the address of the search argument.
- LENGTH specifies that the length of the storage you provide for the RPL is the value of RPLLNGTH.
- MSGAREA and MSGLEN specify a 128-byte area used for physical-error messages.
- OPTCD specifies the subparameters that govern the request defined by the request parameter list identified by SKP and UPD.

- WAREA specifies that the storage you provide for the RPL begins at label MYRPL.
- MF specifies that the GENCB parameter list is to be built in the location specified by register 2. Also, the expansion of the GENCB macro will equate LEN2 to the length of the GENCB parameter list.

GENCB—list form: The format of the list form of GENCB follows.

The format of the list form of GENCB.

Label	Operand	Parameters
[*label*]	**GENCB**	**BLK={ACB | EXLST | RPL}** **[,AM=VSAM]** **[,COPIES=***abs expression***]** **[,***keyword***={***address* | *name* | *abs expression* | *option***},...]** **[,LENGTH=***abs expression***]** **[,LOC={BELOW | ANY}]** **[,RMODE31={ALL | BUFF | CB | NONE}]** **,MF={L | (L,***address***[,***label***])}** **[,WAREA=***address***]**

GENCB—execute form: The format of the execute form of GENCB follows.

The format of the execute form of GENCB.

Label	Operand	Parameters
[*label*]	**GENCB**	**BLK={ACB | EXLST | RPL}** **[,AM=VSAM]** **[,COPIES=***abs expression***]** **[,***keyword***={***address* | *name* | *abs expression* | *option***},...]** **[,LENGTH=***abs expression***]** **[,LOC={BELOW | ANY}]** **[,RMODE31={ALL | BUFF | CB | NONE}]** **,MF=(E,***address***)** **[,WAREA=***address***]**

GENCB—generate form: The format of the generate form of GENCB follows.

The format of the generate form of GENCB.

Label	Operand	Parameters
[*label*]	**GENCB**	**BLK={ACB | EXLST | RPL}** **[,AM=VSAM]** **[,COPIES=***abs expression***]** **[,***keyword=address* | *name* | *abs expression* | *option***},...]** **[,LENGTH=***abs expression***]** **[,LOC={BELOW | ANY}]** **[,RMODE31={ALL | BUFF | CB | NONE}]** **,MF=(G,***address***[,***label***])** **[,WAREA=***address***]**

IDALKADD—RLS record locking

The IDALKADD macro is a VSAM RLS and DFSMStvs request macro. Applications or application support packages that perform logging of changes to VSAM data sets, such as CICS file control, use the IDALKADD macro. With logging, it is necessary to create a log entry before making the corresponding change to the data set or database. The log entry must uniquely identify the

inserted, deleted, or changed record. When VSAM rejects an ADD due to a duplicate-key condition, logging an ADD to a KSDS presents a problem. Also, the record identification for an ESDS is the record RBA, and logging an ADD to an ESDS implies that the RBA of the record is known before VSAM actually adds the record. An IDALKADD request addresses these two situations.

Before IDALKADD adds a record to a KSDS, RRDS, or VRRDS, via the base or a path, the macro performs duplicate key or RRN checking. If a record with the specified key or RRN already exists in the base, the IDALKADD macro fails with the duplicate key or RRN error status. If the base data set does not contain a record with the specified key or RRN, IDALKADD obtains a record lock to ensure that no other LUWID application can add a record with this key or RRN. For an IDALKADD SEQ request to add a record to an RRDS sequentially, VSAM assigns and returns the RRN. An IDALKADD request to add a record to an ESDS returns the RBA that will be assigned to the new record, locks the record RBA, and ensures no other LUWID application can add a record that will be assigned that RBA to the ESDS. Upon successful completion of this request, the requestor writes an entry in a recovery log/journal and issues a VSAM PUT RPL OPTCD=(NUP) to add the record to the data set.

The PUT request must use the same RPL as was used by the IDALKADD. The IDALKADD and PUT NUP are a request pair in the same sense as GET UPD and PUT UPD are a request pair. Reuse of the RPL before issuing the PUT NUP cancels the IDALKADD. The length of the record specified on IDALKADD and the subsequent PUT must be the same or the PUT request is rejected with an invalid record length reason code. For a KSDS, RRDS, or VRRDS, the PUT must specify a record with the same base key/RRN as was specified by the IDALKADD request. A commit protocol application must specify the same LUWID in the PUT request as was specified in the paired IDALKADD request.

Even though an IDALKADD is successful, the corresponding PUT NUP may fail. An example of where the PUT NUP would fail is the condition where the PUT NUP would create a duplicate key in an alternate index and the alternate index requires unique keys. In this case, the PUT NUP fails.

IDALKADD is supported for both base and path access. IDALKADD is supported for both recoverable spheres and non-recoverable spheres. It is supported for KSDSs, ESDSs, RRDSs, and VRRDSs.

The record lock acquired by an IDALKADD request is released as follows:

- Recoverable Sphere

  CICS transactions and batch jobs that use DFSMStvs are allowed to add records to a recoverable sphere. An NSR batch job can add records to a recoverable sphere when the data set is not open for VSAM RLS or DFSMStvs access (for example, it has been quiesced for VSAM RLS and DFSMStvs access). The record lock is released at the end of the transaction.
- Non-Recoverable Sphere

  The following events release the record lock.
  - The paired PUT NUP is issued and the data CI containing the new record has been written to DASD and the coupling facility.
  - An ENDREQ is issued on the string.
  - The string (RPL) is re-used without issuing the paired PUT NUP.
  - The CICS transaction reaches end-of-transaction.

The keep lock option (KL/NOKL) supported by RLS and DFSMStvs for GET UPD does not apply and is ignored by an IDALKADD request.

VSAM does not support PUT NUP,SEQ in backward processing mode. This also means IDALKADD SEQ,BWD is not supported.

The format of the IDALKADD macro follows.

The format of the IDALKADD macro.

Label	Operand	Parameters
[*label*]	**IDALKADD**	**RPL=***address*

label
specifies 1 to 8 characters that provide a symbolic address for the IDALKADD macro.

RPL=*address*
specifies the address of the request parameter list that defines this IDALKADD request. You may specify the address in register notation (using a register from 1 through 12, enclosed in parentheses) or specify it with an expression that generates a valid relocatable A-type address constant.

The following RPL parameters apply to this request:

AREA
Contains a copy of the record that will be added to the data set by a PUT NUP request

When you issue IDALKADD to add a record to a KSDS, a record lock is obtained on the specified record. The record lock name is derived from the base key of the record. The base key is extracted from this copy of the record.

AREALEN
Length of the record. The subsequent PUT must specify the same length.

ARG
For an IDALKADD DIR/SKP to a RRDS or an IDALKADD DIR/SKP/SEQ request to a VRRDS, the application provides the RRN of the new record here.

LUWID
LUWID identifies the *logical unit of work ID* associated with a request to VSAM.

A commit protocol application (for example, CICS) must specify a nonzero value for this parameter. Noncommit protocol applications (for example, batch jobs that are not using DFSMStvs access) and batch applications that are using DFSMStvs access do not specify the LUWID parameter. When RLS access is used, VSAM assigns a single LUWID for all requests issued by a noncommit protocol application. When DFSMStvs access is used, DFSMStvs fills in the LUWID value using the unit of recovery ID obtained from resource recovery services (RRS).

For an ESDS, the IDALKADD request returns the RBA that will be assigned to the record by the subsequent PUT NUP request. The RBA value is returned in field RPLDDDD of the RPL for a non-extended-addressable data set, and in the lower six bytes of the field RPLRBAR for an extended-addressable data set.

For an IDALKADD SEQ to a RRDS, the RPL/string must be positioned to an empty slot. VSAM assigns the RRN of the empty slot to the new record and returns the RRN value in the area pointed to by the RPL ARG parameter.

CICS uses the LUWID parameter to pass a CICS transaction identifier to VSAM RLS. LUWID must not be specified by other (non-CICS) programs.

MODCB—modify an access method control block

The format of the MODCB macro used to modify an access method control block follows.

The format of the MODCB macro used to modify an access method control block.

Label	Operand	Parameters
[*label*]	**MODCB**	**ACB=***address* **[BSTRNO=***abs expression***]** **[,BUFND=***abs expression***]** **[,BUFNI=***abs expression***]** **[,BUFSP=***abs expression***]** **[,DDNAME=***character string***]** **[,EXLST=***address***]** **[,MACRF=([ADR][,CNV] [,KEY]** **[,CFX | NFX]** **[,DDN | DSN]** **[,DFR | NDF]** **[,DIR][,SEQ][,SKP]** **[,ICI | NCI]** **[,IN][,OUT]** **[,NIS | SIS]** **[,NRM | AIX]** **[,NRS | RST]** **[,NSR | LSR | GSR]** **[,NUB | UBF])]** **[,MAREA=***address***]** **[,MLEN=***abs expression***]** **[,PASSWD=***address***]** **[,RMODE31={ALL | BUFF | CB | NONE}]** **[,SHRPOOL=***abs expression***]** **[,STRNO=***abs expression***]**

The subparameters of the MODCB macro can be expressed as absolute numeric expressions, as character strings, as codes, as expressions that generate valid relocatable A-type address constants, in register notation, as S-type address constants, and as indirect S-type address constants. "Subparameters with GENCB, MODCB, SHOWCB, and TESTCB" on page 131, further defines these operand expressions.

label
specifies 1 to 8 characters that provide a symbolic address for the MODCB macro.

ACB=*address*
specifies the address of the access method control block to be modified. The data set identified by the access method control block must not be opened. A request to modify the access method control block of an open data set will fail.

Important: The remaining parameters represent parameters of the ACB macro that can be modified. The value specified replaces the value, if any, presently in the access method control block. *There are no defaults*. For an explanation of these parameters, see "ACB—generate an access method control block at assembly time" on page 136.

If MODCB is used to modify a MACRF subparameter, other subparameters are unaffected, except when they are mutually exclusive. For example, if you specify MACRF=ADR in the MODCB and MACRF=KEY is already indicated in the control block, both ADR and KEY are now indicated. But, if you specify MACRF=UBF in the MODCB and NUB is indicated, only UBF will now be indicated.

The RMODE31 parameter tells the VSAM OPEN routines where to obtain storage for the control blocks and I/O buffers. Therefore, the only time the values specified by the RMODE31 parameter have any effect on VSAM is on the setting just before an OPEN is issued. At other times, changing these values has no effect on the residency of the control blocks and I/O buffers. RMODE31 is ignored for RLS processing.

If MODCB RPL is used to change the address of an ACB, you must first issue an ENDREQ macro.

Restriction: If you issue a MODCB for a non-VSAM and non-VTAM ACB, the results will be unpredictable.

Example: MODCB macro (modify an access method control block): In this example, a MODCB macro is used to modify the name of the exit list in an access method control block.

```
MODCB  ACB=BLOCK,          BLOCK was generated at             x
       EXLST=EGRESS        assembly.
```

MODCB—modify an exit list

The format of the MODCB macro used to modify an exit list follows.

The format of the MODCB macro used to modify an exit list.

Label	Operand	Parameters
[*label*]	**MODCB**	**EXLST=***address* **[,EODAD=([***address***][,A | N][,L])]** **[,JRNAD=([***address***][,A | N][,L])]:** **[,LERAD=([***address***][,A | N][,L])]** **[,SYNAD=([***address***][,A | N][,L])]**

The subparameters of the MODCB macro can be expressed as absolute numeric expressions, as character strings, as codes, as expressions that generate valid relocatable A-type address constants, in register notation, as S-type address constants, and as indirect S-type address constants. “Subparameters with GENCB, MODCB, SHOWCB, and TESTCB” on page 131, further defines these operand expressions.

For information about what determines the addressing mode and the parameter list residency mode set when the exit routine gets control, see *z/OS DFSMS Using Data Sets*.

label
: specifies 1 to 8 characters that provide a symbolic address for the MODCB macro.

EXLST=*address*
: specifies the address of the exit list to be modified. You can modify an exit list at any time—that is, before or after opening the data sets for which the list indicates exit routines. You cannot, however, add an entry to the exit list if it changes the exit list's length; the exit list must already be large enough to

contain the new exit address. The order in which addresses are stored in the EXLST control block is: EODAD, SYNAD, LERAD, JRNAD, and UPAD. For example, if you generate an exit list with only the LERAD exit, you can add entries for EODAD and SYNAD later. However, you cannot add the JRNAD exit address, because doing so would increase the size of the EXLST control block. The MODCB macro does not support the UPAD user exit.

The remaining parameters represent parameters of the EXLST macro that can be modified or added to an exit list. For an explanation of these parameters, see "EXLST—generate an exit list at assembly time" on page 145.

Requirement: If the JRNAD exit is changed for an OPEN ACB, then the ACB must be closed and reopened to use the modified JRNAD exit.

For more information about user exit routines, see *z/OS DFSMS Using Data Sets*.

Example: MODCB macro (modify an exit list): In this example, a MODCB macro is used to activate an exit in an exit list.

```
         MODCB EXLST=(*,        Indirect notation is used to specify   x
               EXLSTADR),       the address of the exit list generated x
         .     EODAD=(EOD,L,A)  at execution.
         .
EOD      DC    C'ENDUP'
EXLSTADR DS    F                When the exit list was generated,      x
                                its address was saved here.
```

The MODCB macro's parameters are:

- EXLST specifies the address of the exit list being modified is located at EXLSTADR.
- EODAD specifies the entry for the end-of-data routine is marked active in the exit list that has an address at EXLSTADR. The name of the end-of-data routine (ENDUP) is at EOD.

MODCB—modify a request parameter list

The format of a MODCB macro used to modify a request parameter list follows.

The format of a MODCB macro used to modify a request parameter list.

Label	Operand	Parameters
[*label*]	**MODCB**	**RPL=***address* **[,ACB=***address***]** **[,AREA=***address***]** **[,AREALEN=***abs expression***]** **[,ARG=***address***]** **[,ECB=***address***]** **[,KEYLEN=***abs expression***]** **[,MSGAREA=***address***]** **[,MSGLEN=***abs expression***]** **[,NXTRPL=***address***]** **[,OPTCD=([ADR|CNV|KEY]** **[,DIR|SEQ|SKP]** **[,ARD|LRD]** **[,FWD|BWD]** **[,ASY|SYN]** **[,NSP|NUP|UPD]** **[,KEQ|KGE]** **[,FKS|GEN]** **[,LOC|MVE]]** **[,RBA|XRBA])]** **[,RECLEN=***abs expression***]** **[,TRANSID=***abs expression***]**

The subparameters of the MODCB macro can be expressed as absolute numeric expressions, as character strings, as codes, as expressions that generate valid relocatable A-type address constants, in register notation, as S-type address constants, and as indirect S-type address constants. "Subparameters with GENCB, MODCB, SHOWCB, and TESTCB" on page 131, further defines these operand expressions.

label
: specifies 1 to 8 characters that provide a symbolic address for the MODCB macro.

RPL=*address*
: specifies the address of the request parameter list being modified. You may not modify an active request parameter list; one that defines a request that has been issued but not completed. To modify such a request parameter list, you must first issue a CHECK or an ENDREQ macro.

Important: The remaining parameters represent parameters of the RPL macro that can be modified. The value specified replaces the value, if any, presently in the request parameter list. *There are no defaults*. For an explanation of these parameters, see "GENCB—generate a request parameter list at execution time" on page 159.

If MODCB is used to modify an OPTCD subparameter within a group of subparameters, the current subparameter for that group is changed because only one subparameter in a group is effective at a time. Only the specified OPTCD subparameter is changed.

Example: MODCB macro (modify a request parameter list): In this example, a MODCB macro is used to modify the record length field in a request parameter list.

This example shows the one exception to GENCB, MODCB, SHOWCB, and TESTCB building a parameter list and passing it to the control block manipulation module in register 1. The RPL address (in register 2) is loaded into register 1 and the RECLEN value (in register 3) is loaded into register 0. These registers are passed to the control block manipulation macro. This occurs when the LIST, EXECUTE, or GENERATE form of the MODCB macro is not used and the only parameter specified other than RPL, is RECLEN.

```
L      3,length      Load the new record length.

MODCB  RPL=(2),      Register 2 contains the address          x
                     of the request parameter list.           x
       RECLEN=(3)    Register 3 contains the record length.
```

The MODCB macro's parameters are:

- RPL specifies register 2 contains the address of the request parameter list being modified.
- RECLEN specifies the record length field is being modified. The contents of register 3 replace the current value in the RECLEN field.

MODCB—list form: The format of the list form of MODCB follows.

The format of the list form of MODCB.

Label	Operand	Parameters
[*label*]	**MODCB**	**{ACB | EXLST | RPL}=***address* **,***keyword*=**{***address* | *name* | *abs expression* | *option*}**,...** **,MF={L | (L,***address*[**,***label*]**)}**

MODCB—execute form: The format of the execute form of MODCB is:

The format of the execute form of MODCB.

Label	Operand	Parameters
[*label*]	**MODCB**	**[{ACB | EXLST | RPL}=***address*] **,***keyword*=**{***address* | *name* | *abs expression* | *option*}**,...** **,MF=(E,***address***)**

Requirement: If the execute form of MODCB is used and EXLST is used as a keyword to be processed, the block must be identified by ACB=.

MODCB—generate form: The format of the generate form of MODCB follows.

The format of the generate form of MODCB.

Label	Operand	Parameters
[*label*]	**MODCB**	**{ACB | EXLST | RPL}=***address* **,***keyword=*{*address* | *name* | *abs expression* | *option*}**,...** **,MF=(G,***address*[**,***label*]**)**

RPL—generate a request parameter list at assembly time

Use the RPL macro to generate a request parameter list. Values for RPL macro subparameters can be specified as absolute numeric expressions, character strings, codes, and expressions that generate valid relocatable A-type address constants.

RPL macro syntax

The format of the RPL macro follows.

The format of the RPL macro.

Label	Operand	Parameters
[*label*]	**RPL**	**[ACB=***address***]** **[,AM=VSAM]** **[,AREA=***address***]** **[,AREALEN=***abs expression***]** **[,ARG=***address***]** **[,ECB=***address***]** **[,KEYLEN=***abs expression***]** **[,LUWID=***luwid***]** **[,TIMEOUT=***number***]** **[,MSGAREA=***address***]** **[,MSGLEN=***abs expression***]** **[,NXTRPL=***address***]** **[,OPTCD=([ADR|CNV |KEY]** **;[,DIR|SEQ|SKP]** **;[,ARD|LRD]** **;[,FWD|BWD]** **;[,ASY|SYN]** **;[,NSP|NUP|UPD]** **;[,KEQ|KGE]** **;[,FKS|GEN]** **;[,NWAITX|WAITX]** **;[,LOC|MVE]** **;[,NRI|CR|CRE]** **;[,RBA|XRBA])]** **;[,NRI|CR|CRE] ;[,** **NOKL|KL]** **[,RECLEN=***abs expression***]** **[,TRANSID=***abs expression***]**

label
specifies 1 to 8 characters that provide a symbolic address for the generated request parameter list. You can use *label* in the request macros to give the address of the list. You can use *label* in the NXTRPL parameter of the RPL macro, when you are chaining request parameter lists, to indicate the next list.

ACB=*address*
specifies the address of the access method control block identifying the data set to which access is requested. If you used the ACB macro to generate the control block, you can specify the label of that macro for the address. If the ACB parameter is not coded, you must specify the address before issuing the request.

AM=VSAM
specifies the access method using the control block is VSAM.

AREA=*address*
specifies the address of a work area to and from which VSAM moves a data record if you request it to do so (with the RPL parameter OPTCD=MVE). If

your request is to process records in the I/O buffer (OPTCD=LOC), VSAM puts into this work area the address of a data record within the I/O buffer.

AREALEN=*abs expression*
specifies the length, in bytes, of the work area whose address is specified by the AREA parameter. Its minimum for OPTCD=MVE is the size of a data record (of the largest data record, for a data set with records of variable length). For OPTCD=LOC, the area should be 4 bytes to contain the address of a data record within the I/O buffer.

ARG=*address*
specifies the address of a field that contains the search argument for direct retrieval, skip-sequential retrieval, and positioning. For a RRDS, the ARG field must be 4 bytes long. For direct or skip-sequential processing, this field contains your search argument, a relative record number. For sequential processing (OPTCD=(KEY,SEQ)), the 4 bytes are required for VSAM to return the feedback RRN. For keyed access (OPTCD=KEY), the search argument is a full or generic key or relative record number. For addressed access (OPTCD=ADR), the search argument is an RBA. If you specify a generic key (OPTCD=GEN), you must also specify in the KEYLEN parameter how many of the bytes of the full key you are using for the generic key. ARG is also used with WRTBFR and MRKBFR. Using WRTBFR and MRKBFR to share resources is described in *z/OS DFSMS Using Data Sets*.

ECB=*address*
specifies the address of an event control block (ECB) that you can supply. VSAM indicates in the ECB whether a request is complete or not (using standard completion codes, which are described in *z/OS MVS System Codes*). You can use the ECB to determine that an asynchronous request is complete before issuing a CHECK macro. (If you issue a CHECK before a request is complete, you give up control and must wait for completion.) The ECB parameter is always optional.

KEYLEN=*abs expression*
specifies the length, in bytes, of the generic key (OPTCD=GEN) you are using for a search argument (given in the field addressed by the ARG parameter). This parameter is specified as a number from 1 through 255. It is required when the search argument is a generic key. For full-key searches, VSAM knows the key length, which is taken from the catalog definition of the data set when you open the data set. This parameter is ignored for UNIX files.

LUWID=*luwid*
For For MACRF=RLS only, LUWID identifies the *logical unit of work* associated with a request to VSAM.

A commit protocol application (for example, CICS) must specify a nonzero value for this parameter. Noncommit protocol applications (for example, batch jobs that are not using DFSMStvs access) and batch applications that are using DFSMStvs access do not specify the LUWID parameter. When DFSMStvs access is used, DFSMStvs fills in the LUWID using the URID obtained from RRS.

The LUWID together with the SUBSYSNM (specified when the ACB is opened) form the ownership for record locks. For RLS GET NUP, GET UPD, and POINT only, a commit protocol application must specify a nonzero value for this parameter. A noncommit protocol application does not specify the LUWID parameter. VSAM assigns a single LUWID for all requests issued by a non-commit protocol application.

For RLS PUT UPD/ERASE only, a commit protocol application must specify the same nonzero value in both the preceding GET UPD request and this PUT UPD/ERASE request. A noncommit protocol application does not specify the LUWID parameter. VSAM assigns a single LUWID for all requests issued by a noncommit protocol application.

For RLS PUT NUPD only, a commit protocol application must specify a nonzero value for this parameter. When the IDALKADD protocol is used, both the preceding IDALKADD request and this PUT NUP request must specify the same LUWID value. A noncommit protocol application does not specify the LUWID parameter. VSAM assigns a single LUWID for all requests issued by a noncommit protocol application.

CICS uses the LUWID parameter to pass a CICS transaction identifier to VSAM RLS. LUWID must not be specified by other (non-CICS) programs.

MSGAREA=*address*
specifies the address of an area you might, optionally, supply for VSAM to send you a message in case of a physical error. The format of a physical error message is given in "Reason code (physical errors)" on page 215.

MSGLEN=*abs expression*
specifies the size, in bytes, of the message area indicated in the MSGAREA parameter. If MSGAREA is specified, MSGLEN is required. The minimum size of a message is 128 bytes. If you provide less than 128 bytes, no message is returned to your program.

NXTRPL=*address*
specifies the address of the next request parameter list in a chain. Omit this parameter from the macro that generates the last list in the chain. When you issue a request defined by a chain of request parameter lists, indicate in the request macro the address of the first parameter list in the chain. This parameter is not supported for UNIX files and if it is specified with a non-zero value, results in an error on a subsequent GET, PUT, or POINT request.

OPTCD=([ADR|CNV|KEY]
[,DIR|SEQ|SKP]
[,ARD|LRD]
[,FWD|BWD]
[,ASY|SYN]
[,NSP|NUP|UPD]
[,KEQ|KGE]
[,FKS|GEN]
[,NWAITX|WAITX]
[,LOC|MVE]
[,CR|CRE|NRI]
[,RBA|XRBA])
specifies the subparameters governing the request defined by the request parameter list. Each group of subparameters has a default; subparameters are shown in Table 11 on page 176 with defaults underlined. Only one subparameter from each group can be specified. Some requests do not require a subparameter from all of the groups to be specified. The groups that are not required are ignored. Thus, you can use the same request parameter list for a combination of requests (GET, PUT, POINT, for example) without zeroing out the inapplicable subparameters each time you go from one request to another.

TIMEOUT=*number*
For RLS and DFSMStvs, specifies the time, in seconds, that your program is willing to wait to obtain a lock on a VSAM record when a lock on the record is

already held by another program. A nonzero value for TIMEOUT specifies the time (in seconds) this program will wait for the other program(s) to release the lock.

A value of zero specifies that TIMEOUT processing is not to be performed by VSAM for this request. That is, if the record lock required by the request is held by another program, the program waits until the other program releases the lock regardless of how long that might be.

Any value that you specify for this parameter overrides a value that was specified using RLSTMOUT in the JCL. This parameter is ignored for UNIX files.

Table 11. OPTCD options. OPTCD options

Option	Meaning
ADR	Addressed access to a key-sequenced or an entry-sequenced data set: RBAs are used as search arguments, and sequential access is done by entry sequence. VSAM RLS and DFSMStvs do not support addressed access to a KSDS.
CNV	Control interval access. Control interval access is not allowed for compressed data sets. VSAM RLS and DFSMStvs do not support CNV access. This parameter is ignored for UNIX files and if it is specified, results in an error on a subsequent GET, PUT, or POINT request.
KEY	Keyed access to an RRDS or KSDS. Keys or relative record numbers are used as search arguments and sequential access is done by key or relative record number sequence.
DIR	Direct access to a RRDS, KSDS, or ESDS.
SEQ	Sequential access to a RRDS, KSDS, or ESDS.
SKP	Skip sequential access.
ARD	User's argument determines the record to be located, retrieved, or stored.
LRD	Last record in the data set is to be located (POINT) or retrieved (GET direct); requires OPTCD=BWD.
FWD	Processing to proceed in a forward direction.
BWD	Processing to proceed in a backward direction; for keyed (KEY) or addressed (ADR) sequential (SEQ) or direct (DIR) requests; valid for POINT, GET, PUT, and ERASE operations; establish positioning by a POINT with OPTCD=BWD or by a GET direct with OPTCD=(NSP,BWD). When OPTCD=BWD is specified, the subparameters KGE and GEN are ignored and the subparameters KEQ and FKS are assumed. This parameter is ignored for UNIX files and if it is specified, results in an error on a subsequent GET, PUT, or POINT request.
ASY	Asynchronous access; VSAM returns to the processing program after scheduling a request so the program can do other processing while the request is being carried out.
SYN	Synchronous access; VSAM returns to the processing program after completing a request.
NSP	With OPTCD=DIR only, VSAM is to remember its position (for subsequent sequential access); that is, the position is not to be forgotten unless an ENDREQ macro is issued.

Table 11. OPTCD options (continued). OPTCD options

Option	Meaning
NUP	A data record being retrieved will not be updated or deleted; a record being stored is a new record; VSAM does not remember its position for direct requests into a work area.
UPD	A data record being retrieved can be updated or deleted; a record being stored or deleted was previously retrieved with OPTCD=UPD; VSAM remembers its position for sequential and direct GET requests. When PUT, ERASE or ENDREQ is issued after a DIR UPD GET request, VSAM releases exclusive control. This parameter is not supported for UNIX files and if it is specified, results in an error on a subsequent GET, PUT, or POINT request.
KEQ	For GET with OPTCD=(KEY,DIR) or (KEY,SKP) and for POINT with OPTCD=KEY, the key (full or generic) that you provide for a search argument must equal the key or relative record number of a record. For a RRDS, KEQ is assumed except for POINT.
KGE	For the same cases as KEQ, if the key (full or generic) that you provide for a search argument does not equal that of a record, the request applies to the record that has the next higher key. If using POINT with a RRDS, KGE positions to the specified relative record number whether the slot is empty or not. If the relative record number is greater than the highest existing record, EOD is returned. A subsequent PUT will insert the record at this position.
FKS	A full key is provided as a search argument.
GEN	A generic key is provided as a search argument; give the length in the KEYLEN parameter. Generic keys are not supported for a variable-length RRDS.
NWAITX	Never take the UPAD or RLSWAIT exit.
WAITX	If OPTCD=SYN and the ACB's MACRF=LSR GSR and UPAD exit routing is specified, VSAM takes the UPAD exit at points when VSAM would normally issue a WAIT. For VSAM RLS and DFSMStvs, take the RLSWAIT exit, which is active for this request.
LOC	For retrieval, VSAM leaves the data record in the I/O buffer for processing, unless the data set is compressed, in which case VSAM moves the record to a work area; not valid for PUT or ERASE; valid for GET with OPTCD=UPD. However, to update the record, you must build a new version of the record in a work area and modify the request parameter list OPTCD from LOC to MVE before issuing a PUT. For keyed-sequential retrieval, modifying key fields in the I/O buffer might cause incorrect results for subsequent GET requests until the I/O record is reread.Not valid for requests with spanned records. For UNIX files, LOC mode is supported but requires extra overhead to get storage in the user space and move the record.
MVE	For retrieval, VSAM moves the data record to a work area for processing, and for storage, VSAM moves it from the work area to the I/O buffer.

Table 11. OPTCD options (continued). OPTCD options

Option	Meaning
UPD,KL	For VSAM RLS and DFSMStvs only, UPD,KL specifies a GET for update to a recoverable sphere. UPD,KL specifies that the lock obtained as part of GET UPD processing is to be explicitly held until a request is issued by the commit protocol application (for example, CICS) to release all record locks held by the LUWID. The lock is held even if the RPL is reused or ENDREQ issued prior to the corresponding PUT UPD or ERASE. Also, the status of the lock is such that it might become a retained lock. UPD,KL is ignored for a non-recoverable sphere.
UPD,NOKL	For RLS only, UPD,NOKL specifies a GET for update to a recoverable sphere. UPD,NOKL specifies that the lock obtained as part of GET UPD processing is to be released if the RPL is reused or ENDREQ issued prior to the corresponding PUT UPD or ERASE. Also, the status of the lock is such that it will not become a retained lock. UPD,NOKL is the default.
CR	For RLS and DFSMStvs POINT and GET NUP interfaces, CR (consistent read integrity) specifies that a shared lock is to be obtained and released as part of GET processing. CR specifies that the application wants this request to be serialized with update/erase of this record by other LUWIDs and by other RPLs used by this LUWID. applications or transactions. RLS obtains a shared lock on the record. For RLS and DFSMStvs POINT, the shared lock remains held on successful completion of the POINT CR request. For RLS and DFSMStvs GET, after a copy of the record is moved to the area pointed to by the RPL AREA parameter, the shared lock is released. If neither NRI nor CR is specified, the NRI/CR option is determined in the following order: 1. RLSREAD specification in the ACB, if any 2. RLS JCL specification, if any 3. NRI is assumed. If there are multiple specifications in the RPL, CR takes precedence over NRI.

Table 11. OPTCD options (continued). OPTCD options

Option	Meaning
CRE	For RLS and DFSMStvs POINT and GET NUP interfaces, CRE (consistent read explicit) specifies that a shared lock is to be obtained. This lock is released by RLS or DFSMStvs upon completion of this transaction. This provides a repeatable read function for CICS transactions or DFSMStvs URs. CRE specifies that the application wants this request to be serialized with update/erase of this record by other LUWIDs/URs and by other RPLs used by this LUWID/UR. RLS obtains a shared lock on the record. The shared lock remains held on successful completion of the POINT CRE or GET CRE request. It will be released if positioning to the record is lost and the record is not updated. If the record requested by the POINT CR/CRE request does NOT exist, the request fails with "record not found" error status (same as for VSAM NSR/LSR access). Because the record does not exist, the CR/CRE request does not obtain a shared lock on the specified record ID. CRE is valid only when used by CICS in RLS mode or for data sets accessed in DFSMStvs mode. RLS obtains a shared lock on the record. After a copy of the record is moved to the area to which the RPL AREA parameter points, the status of the shared lock is changed to permit update/erase of the record by this LUWID/UR using this or another RPL. However, the lock remains held, inhibiting (delaying) update/erase of this record by another LUWID/UR until this LUWID/UR reaches the end of the transaction. If NRI, CR, or CRE is not specified, integrity is assumed in the following order: • RLSREAD specification on the ACB, if any, • RLS JCL specification, if any, • NRI is assumed. If there are multiple specifications in the RPL, CRE takes precedence over CR, which takes precedence over NRI.

Table 11. OPTCD options (continued). OPTCD options

Option	Meaning
NRI	For RLS and DFSMStvs POINT and GET NUP interfaces, NRI (no read integrity) specifies no locking on a GET (nonupdate). Because a lock is not obtained on the record, another application or transaction might currently hold an exclusive lock on the record. For a recoverable sphere, the returned record might be an uncommitted change that might be later backed out (this form of processing is sometimes referred to as "dirty read"). The opposite form of read processing is provided by the CR option where if another application/transaction holds an exclusive lock on the record, the reader waits for release of the exclusive lock and thus does NOT read an uncommitted change. If neither NRI or CR is specified, the NRI/CR option is determined in the following order: • RLSREAD specification on the ACB, if any, • RLS JCL specification, if any, • NRI is assumed. If there are multiple specifications in the RPL, CR takes precedence over NRI. Inserting or updating a base cluster record can result in a concurrent NRI read of the record by an alternate index path, causing you to receive a false error (return code 8, reason code 144 in Table 23 on page 204). RLS obtains a record lock and retries the request to be sure this is not a false condition.
RBA	For addressed accessing (OPTCD=ADR), the ARG field contains the address of a 4-byte RBA. RBA is the default. Extended addressing is not to be used for this request.

Table 11. OPTCD options (continued). OPTCD options

Option	Meaning
XRBA	For addressed accessing (OPTCD=ADR), the ARG field contains the address of an 8-byte RBA search argument. While you can specify RBA while using XRBA, the following considerations apply to accessing by RBA values: • For a GET extended addressing request, you must specify an OPTCD that includes DIR, ADR, and XRBA. • For a POINT extended addressingrequest, you must specify an OPTCD that includes ADR and XRBA. • For a MRKBFR extended addressingrequest, you must specify an OPTCD which includes XRBA. The ARG field has the address of a 16 byte field containing the beginning and ending 8 byte RBAs of the range. • For a SCHBFR extended addressingrequest, you must specify an OPTCD which includes XRBA. The ARG field has the address of a 16 byte field containing the beginning and ending 8 byte RBAs of the range. • For a WRTBFR TYPE=DRBA extended addressingrequest, you must specify an OPTCD which includes XRBA. The ARG field has the address of an 8 byte field containing the 8 byte RBA to be located and written. If the data being referenced by RBA for an extended addressing KSDS is less than 4GB, you do not have to code this parameter. For data with RBA greater than 4GB the RPL must specify extended addressing (XRBA) and an 8-byte RBA is required. Also, to retrieve an 8-byte RBA using SHOWCB for the RPL, XRBA must be used instead. XRBA specification can be used for any data set (whether or not it is extended addressable).

RECLEN=*abs expression*
specifies the length, in bytes, of a data record being stored. This parameter is required for a PUT request.

For GET requests, VSAM puts the length of the record retrieved in this field in the request parameter list. It will be there if you update and store the record.

TRANSID=*abs expression*
specifies a number that relates modified buffers in a buffer pool. Used in shared resource applications and described in *z/OS DFSMS Using Data Sets*. This parameter is ignored for UNIX files.

For RLS, this parameter is ignored. LUWID replaces this function.

Example: RPL macro

In this example, an RPL macro is used to generate a request parameter list named PARMLIST.

```
ACCESS   ACB   MACRF=(SKP,OUT),                                  x
               DDNAME=PAYROLL

PARMLIST RPL   ACB=ACCESS,                                       x
               AM=VSAM,                                          x
               AREA=WORK,                                        x
               AREALEN=125,                                      x
               ARG=SEARCH,                                       x
```

```
                MSGAREA=MESSAGE,                                        x
                MSGLEN=128,                                             x
                OPTCD=(SKP,UPD)   Most OPTCD defaults are appropriate   x
                                  to assumptions.
WORK      DS    CL125
SEARCH    DS    CL8
MESSAGE   DS    CL128
```

The ACB macro named ACCESS, specifies skip-sequential retrieval for update. Further details might be provided on a DD statement named PAYROLL.

The RPL macro's parameters are:

- ACB associates the request parameter list with the access method control block generated by ACCESS.
- AREA and AREALEN specify a work area, WORK, that is 125 bytes long.
- ARG specifies the search argument is defined at SEARCH. The search argument is 8 bytes long.
- MSGAREA and MSGLEN specify a message area, MESSAGE, that is 128 bytes long. The message area is provided for physical error messages.
- OPTCD specifies skip-sequential processing and specifies that a retrieved record can be updated or deleted.
- NSR is assumed.

Because KEYLEN is not coded, a full-key search is assumed.

SCHBFR—search buffer

If you are using local or global shared resources, you can use the SCHBFR macro to search a buffer.

The format of the SCHBFR macro follows.

The format of the SCHBFR macro.

Label	Operand	Parameters
[*label*]	**SCHBFR**	[**BFRNO=***abs expression*] **,RPL=***address*

label
specifies 1 to 8 characters that provide a symbolic address for the SCHBFR macro.

BFRNO=*abs expression*
specifies the number of the buffer VSAM is to search first. The buffers preceding it in the buffer pool are not searched. The default is 1; that is, the first buffer is searched first. (If the number is coded in register notation, all registers except 1 and 13 may be used.)

The meaning of BFRNO depends on the total number of buffers in the buffer pool and the number of control intervals in the RBA range given by the RPL ARG parameter. This number is the buffer number relative to the beginning of the RBA range if the total number of buffers in the buffer pool is greater than (3/4 x number of CIs in the RBA range)+3. Otherwise, it is the buffer number on the physical buffer chain.

Restriction: When a data set is in a compressed format, records might be compressed and each buffer might contain an unpredictable amount of data.

RPL=*address*
specifies the address of the request parameter list defining the SCHBFR request. These RPL parameters have meaning for SCHBFR:

ACB=*address*
AREA=*address*
If a buffer is found, the area whose address is specified contains its address (OPTCD=LOC) or a copy of its contents (OPTCD=MVE). With compressed data sets, the contents of the buffer will not be in a readable format. SCHBFR is not recommended for compressed data sets.

AREALEN=*abs expression*
At least 4 with OPTCD=LOC; at least control interval size with OPTCD=MVE.

ARG=*address*
ARG gives the address of an 8-byte field containing the beginning and ending control interval RBAs of the range to be searched on. For compressed data sets, the RBA of another record or the address of the next record in a buffer cannot be determined using the length of the current record or the length of the record provided to VSAM.

For extended addressing, the addressof a 16-byte field containing the beginning and ending 8-byte RBAs of the range.

ECB=*address*

OPTCD=({ASY|SYN},{LOC|MVE})

TRANSID=*abs expression*

All other RPL parameters are ignored. RPLs are assumed not to be chained. Control interval access is assumed.

If the ACB to which the RPL is related has MACRF=GSR, the program issuing SCHBFR must be in supervisor state with protection key 0 to 7.

SHOWCAT—display the catalog

The information shown here is provided for compatibility only.

The SHOWCAT (show, or display, the catalog) macro enables you to retrieve information from a catalog independently of an open data set defined in the catalog.

The SHOWCAT macro has three forms: standard, list, and execute. Although the integrated catalog facility catalog have different structures, the SHOWCAT macro supports integrated catalog facility catalogs. Thus, all references to catalogs in this discussion of the SHOWCAT macro apply to integrated catalog facility catalogs.

You can use the IGGSHWPL macro to generate a DSECT statement and labels for the fields in the parameter list for SHOWCAT.

The entries in a catalog are interrelated. More than one entry is required to describe an object and its associated objects; one entry points to one or more other entries, which point to yet others. Figure 5 on page 184 shows the interrelationship among entries that describe the following types of objects:

- Alternate index (G)
- Cluster (C)
- Data component (D)

- Index component (I)
- Path (R)
- Upgrade set (Y)

For example, an alternate-index entry points to the entries of its data and index components, its base cluster, and its path. SHOWCAT enables you to follow the arrows in Figure 5. You first issue SHOWCAT on the name of an object.

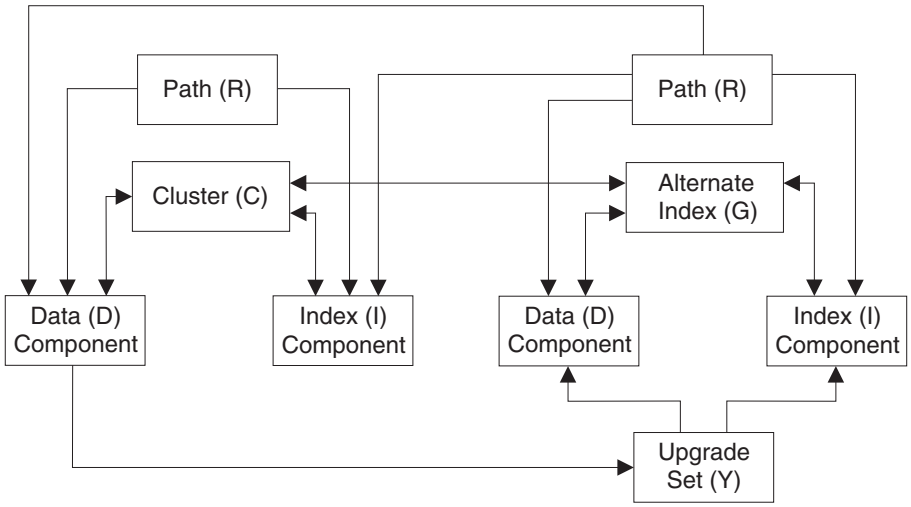


Figure 5. Interrelationships among catalog entries. An arrow indicates a pointer from one entry to another.

The information VSAM returns to you includes the control interval numbers of catalog records in entries describing associated objects. You then issue SHOWCAT on a control interval number to retrieve information from one of these other entries.

The first time you issue SHOWCAT, VSAM searches VSAM catalogs in the following order to locate the entry that describes the object you name:

1. The STEPCAT or JOBCAT user catalog or catalogs (catalogs can be concatenated under STEPCAT or JOBCAT).
2. The master catalog.
3. When the object has a qualified name, the catalog, if any, whose name or alias is the same as the first-level qualifier of the object's name.

VSAM returns the address of the access method control block that defines the catalog. In subsequent use of SHOWCAT, you can specify that address, which causes VSAM to search only that catalog.

SHOWCAT should not be used for UNIX files because UNIX files are not reflected in the catalogs. Specifying the path name in the NAME parameter is not valid and returns unpredictable results.

SHOWCAT is valid in AMODE 24 mode only.

SHOWCAT—standard form

The format of the SHOWCAT macro follows.

The format of the SHOWCAT macro.

Label	Operand	Parameters
[*label*]	**SHOWCAT**	**[ACB=***address***]** **[AREA=***address***]** **[{CI=***address* **| NAME=***address***}]**

label
specifies 1 to 8 characters that provide a symbolic address for the SHOWCAT macro.

ACB=*address*
specifies the address of the access method control block that defines the catalog containing the entry from which to display information. You issue the first SHOWCAT without ACB specified and VSAM supplies it to you for the next SHOWCAT (see the description of the work area under the AREA operand). Specifying ACB enables VSAM to go directly to the correct catalog without searching other catalogs first. You should always specify ACB when specifying CI instead of NAME.

AREA=*address*
specifies the address of the work area in which to display the catalog information. The first 2 bytes of the area must give the length of the area, including the 2 bytes. The minimum is 64. If the area is too small, VSAM returns as much information as possible.

You can use the IGGSHWPL macro to generate a DSECT statement and labels for the fields in the work area.

The format of the work area follows.

The format of the work area for the IGGSHWPL macro.

Offset	Length	Symbolic Name	Description
0(X'00')	2	SHWLEN1	Length of the area, including the length of this field (provided by you).
2(X'02')	2	SHWLEN2	Length of the area used by VSAM, including the length of this field and the preceding field.
4(X'04')	4	SHWACBP	The address of the ACB that defines the catalog that contains the entry from which information is displayed.
8(X'08')	1	SHWTYPE	Type of object about which information is returned: **C** Cluster **D** Data component **G** Alternate index **I** Index **R** Path **Y** Upgrade set

The following fields contain one set of information for C, G, R, and Y types and another set for D and I types:

The format of the work area for **C, G, R,** and **Y types** follows.

The format of the work area for C, G, R, and Y types.

Offset	Length or Bit Pattern	Symbolic Name	Description
9(X'09')	1	SHWATTR	For C and Y types: reserved.
			For G type:
	x...	SHWUP	The alternate index may (1) or may not (0) be a member of an upgrade set. One way of verifying this is to display information for the upgrade set of the base cluster and check whether it contains control interval numbers of entries that describe the components of the alternate index. Figure 5 on page 184 shows how to get from the alternate index's catalog entry to the entries that describe its components (G to C to D to Y to D and I).
	.xxx xxxx		Reserved.
			For R type:
	x...	SHWUP	The path is (1) or is not (0) defined for upgrading alternate indexes.
	.xxx xxxx		Reserved.
10(X'0A')	2	SHWASS0	The number of association pointers that follow.
		SHWACT	Each association pointer identifies another catalog entry that describes an object associated with this C, G, R, or Y object. The possible types of associated objects are: • With C: D, G, I, R. • With G: C, D, G, I. • With R: C, D, G, I. • With Y: D, I. Figure 5 on page 184 shows how the catalog entries for all these objects are interrelated.
12(X'0C')	1	SHWATYPE	Type of object the entry describes.
13(X'0D')	3	SHWAC1	The control interval number of its first record.

The format of the work area for C, G, R, and Y types.

Offset	Length or Bit Pattern	Symbolic Name	Description
16(X'10')			Next association pointer, and so on. For type Y, if the area is too small to display an association pointer for each associated object, VSAM displays as many pointers as possible and returns a code of 4 in register 15. For types C and G, if the area is too small, VSAM displays as many pointers as possible, but returns as a code of 0 in register 15 because fields for the main associated objects can always be displayed (in the smallest allowed work area). For type R, fields for all associated objects (five possible) can always be displayed. (An associated pointer occupies 4 bytes (1 byte for the associated entry type and 3 bytes for its control interval number). However, for all types except Y, 4 additional bytes are required as work space for the SHOWCAT processor. For example, if you provide 80 bytes for associated objects, as many as 10 association pointers can be displayed for type C or G and 20 for type Y.)

The format of the work area for **D** and **I** types follows.

The format of the work area for D and I types.

Offset	Length	Symbolic Name	Description
9(X'09')	1		Reserved.
10(X'0A')	2	SHWDSB	Relative position of the prime key in records in the data component.
		SHWRKP	For the data component of an ESDS, there is no prime key and this field is 0.
12(X'0C')	2	SHWKEYLN	Length of the prime key.
14(X'0E')	4	SHWCISZ	Control interval size of the data or index component.
18(X'12')	4	SHWMREC	Maximum record size of the data or index component.
22(X'16')	2	SHWASS	The number of association pointers that follow.

The format of the work area for D and I types.

Offset	Length	Symbolic Name	Description
		SHWACT	Each association pointer identifies another catalog entry that describes an object associated with this D or I object. The possible types of associated objects are: • With D: C, G, Y. • With I: C, G. Figure 5 on page 184 shows how the catalog entries for all these objects are interrelated.
24(X'18')	1	SHWATYPE	Type of object the entry describes.
25(X'19')	3	SHWACI	The control interval number of its first record.
28(X'1C')			Next association pointer, and so on. Fields for all associated objects can always be displayed.

{CI=*address***|NAME=***address***}**
specifies the address of an area that identifies the catalog entry containing the desired information.

CI=*address*
specifies the area is 3 bytes long and contains the control interval number (RBA divided by 512) of the first record in the catalog entry. You can issue the first SHOWCAT with NAME specified, and then VSAM supplies control interval numbers to you for other SHOWCATs (see the description of the work area under the AREA operand). The type of object named must be C, D, G, I, R, or Y. The 3-byte area must be separate from the work area, even though VSAM returns a control interval number in the work area.

NAME=*address*
specifies the area is 44 bytes long and contains the name of the object described by the entry. The name is left-justified and padded with blanks. The type of object named must be C, D, G, I, or R.

SHOWCAT—list form

The format of the list form of SHOWCAT follows.

The format of the list form of SHOWCAT.

Label	Operand	Parameters
[*label*]	**SHOWCAT**	**[ACB=***address***]** **[AREA=***address***]** **[{CI=***address* **| NAME=***address* **MF=L**

MF=L
specifies that this is the list form of SHOWCAT.

AREA and {CI|NAME} are optional in the list form of SHOWCAT, but, if they are not so specified, they must be specified in the execute form.

For a detailed description of ACB, AREA, and CI|NAME parameters, refer to the information contained in "SHOWCAT—standard form" on page 185.

SHOWCAT—execute form

The format of the execute form of SHOWCAT follows.

The format of the execute form of SHOWCAT.

Label	Operand	Parameters
[*label*]	**SHOWCAT**	**[ACB=***address***]** **[AREA=***address***]** **[{CI=***address***|NAME=***address***}]** **MF=({E|B},***address***)**

MF=({E|B},*address***)**
specifies this is the execute form of SHOWCAT.

E specifies the parameter list, whose address is given in *address*, is passed to VSAM for processing.

B specifies the parameter list is to be built or modified, but is not passed to VSAM. This form of the macro is similar to the list form, except that it works at execution time and can modify a parameter list, as well as build it.

To build a parameter list, first issue SHOWCAT with only MF=(B, address) specified, to zero out the area in which it will be built.

address
specifies the address of the parameter list. If you use register notation, you may use register 1, and a register from 2 through 12. Register 1 is used to pass the parameter list to VSAM (MF=E).

For a detailed description of ACB, AREA, and CI|NAME parameters, refer to the information contained in "SHOWCAT—standard form" on page 185.

Expressions that can be used for SHOWCAT

The values for an operand of SHOWCAT can be expressed as follows:

- An absolute numeric expression.
- A code or a list of codes separated by commas and enclosed in parentheses.
- A register (in parentheses) from 2 through 12 that contains an address or numeric value. In the execute form of a macro, you can use register 1 for the address of the parameter list. Equated labels can be used to designate a register; for example, BFRNO=(BFR#), where the equate statement, BFR# EQU 3, is included in the program.
- An expression valid for a relocatable A-type address constant; for example, AREA=RETURN+4.

The expressions that can be used depend on the operand. Only absolute numeric expressions, codes, registers, and relocatable A-type address constants are valid for the list form of a macro.

Table 12 on page 190 shows the expressions allowed for each operand of SHOWCAT.

Table 12. Operand expressions for the SHOWCAT macro. Operand expressions for the SHOWCAT macro

Operands	Absolute numeric	Code	Register	A-type address
SHOWCAT (STANDARD)				
ACB			X	X
AREA			X	X
CI			X	X
NAME			X	X
SHOWCAT (LIST)				
ACB				X
AREA				X
CI				X
MF		X		
NAME				X
SHOWCAT (EXECUTE)				
ACB			X	
AREA			X	
CI			X	
MF				
B		X		
E		X		
address			X	
NAME			X	

Understanding VSAM macro return and reason codes

This section describes the return codes and reason codes that are generated by the VSAM macros that are used to open and close data sets, manage VSAM control blocks, and issue record management requests.

VSAM sets the return codes in register 15. (For information on register usage conventions, see *z/OS DFSMS Macro Instructions for Data Sets*.) These return codes are paired with reason codes that are set in the access method control block (ACB) and the request parameter list (RPL). Reason codes that are set in the ACB indicate open or close errors. Reason codes that are set in the RPL indicate record management errors.

This section lists return codes and reason codes as decimal and hexadecimal values. The decimal value is shown first, followed by the hexadecimal value in parentheses. For format descriptions and examples macros, see "Coding VSAM macros" on page 131. Some VSAM reason codes that are used for diagnostic purposes are described in "VSAM diagnostic aids" on page 307 or *z/OS DFSMSdfp Diagnosis*.

OPEN return and reason codes

When your program receives control after issuing an OPEN macro, the return code in register 15 indicates whether all the data sets were opened successfully, as Table 13 shows.

Table 13. Return codes in register 15 after OPEN. Return codes in register 15 after OPEN

Return code	Meaning
0(X'0')	All data sets were opened successfully.

Table 13. Return codes in register 15 after OPEN (continued). Return codes in register 15 after OPEN

Return code	Meaning
4(X'4')	All data sets were opened successfully, but one or more attention messages were issued (reason codes less than X'80').
8(X'8')	At least one data set (VSAM or non-VSAM) was not opened successfully; the access method control block was restored to the contents it had before the OPEN was issued; or, if the data set was already open, the access method control block remains open and usable and is not changed.
12(X'C')	A non-VSAM data set was not opened successfully when a non-VSAM and a VSAM data set were being opened at the same time. The non-VSAM data control block was not restored to the contents it had before the OPEN was issued (and the data set cannot be opened without restoring the control block).
16(X'10')	One or more of the access method control blocks (ACBs) specified the RLS option but the system has not been set up for RLS (the SMSVSAM server address space is not available). For other DCBs and ACBs any condition described by other return codes is possible.

If register 15 contains a nonzero return code, use the SHOWCB macro to display the corresponding reason code. The SHOWCB macro displays the error field in each access method control block specified by the OPEN macro. (See *z/OS DFSMS Macro Instructions for Data Sets*.)

Table 14 lists the reason codes that might be in this error field.

Table 14. OPEN reason codes in the ACBERFLG field of the ACB. OPEN reason codes in the ACBERFLG field of the ACB

Reason code	Meaning
0(X'0')	One of the following conditions exists: • VSAM is processing the access method control block for some other request. • The access method control block address is invalid.
72(X'48')	One of the following errors occurred (a warning): • A non-RLS or non-DFSMStvs OPEN for input was successful against a sphere that was already in a lost locks or retained locks state. • A non-RLS or non-DFSMStvs< OPEN for output was successful against a sphere that was already in a lost locks or retained locks state because a NONRLSUPDATE was in effect.
76(X'4C')	The interrupt recognition flag (IRF) was detected for a data set opened for input processing. This indicates that DELETE processing was interrupted. The structure of the data set is unpredictable; the access method services DIAGNOSE command can be used to check the data set for structural errors. For a description of the DIAGNOSE command, see *z/OS DFSMS Access Method Services Commands*.
88(X'58')	A previous extend error has occurred during EOV processing of the data set. For MACRF=RLS, reset processing of “delete vol” has received an error.
92(X'5C')	Inconsistent use of CBUF processing. Sharing options differ between index and data components.
96(X'60')	An unusable data set was opened for input.
100(X'64')	An OPEN found an empty alternate index that is part of an upgrade set.
101(X'65')	For MACRF=RLS, the sphere that was opened is in lost locks state. The open was successful.
102(X'66')	For MACRF=RLS, the sphere is in a non-RLS update permitted state. The open was successful.

Table 14. OPEN reason codes in the ACBERFLG field of the ACB (continued). OPEN reason codes in the ACBERFLG field of the ACB

Reason code	Meaning
103(X'67')	For RLS, the sphere that was opened is in both a lost locks state and non-RLS update permitted state. The open is successful. For DFSMStvs, the open succeeded, but one of the following conditions was detected: • DFSMStvs is quiescing due to an I/O error on one of the system logs (the undo log or the shunt log). • The forward recovery log is quiescing due to an I/O error. Processing continues without the forward recovery log. • A failure occurred during an attempt to write a tie-up record to the forward recovery log. Processing continues without the forward recovery log. • The log of logs is quiescing due to an I/O error. Processing continues without the log of logs. • A failure occurred during an attempt to write a tie-up record to the log of logs. Processing continues without the log of logs.
104(X'68')	The time stamp of the volume where the data set is stored does not match the system time stamp in the data set's catalog record. This indicates extent information in the catalog record might not agree with the extents indicated in the volume's VTOC.
108(X'6C')	The time stamps of a data component and an index component do not match. This indicates that either the data or the index has been updated separately from the other.
110(X'6E')	JRNAD exit was not specified on the first ACB opened for the data set. Processing continues without journaling.
116(X'74')	The data set was not properly closed. The data set high-used RBA has not been verified. Records might be missing or duplicated. A previous VSAM program might have abnormally terminated. You should verify that all of the expected records are in the data set. If you ignore the message and try to process the data set, the results are unpredictable. The catalog will be updated when the data set has been successfully opened for output and then successfully closed. You can determine if this error occurred on opening an empty data set by using the SHOWCB macro instruction. The SHOWCB macro instruction is described in *z/OS DFSMS Macro Instructions for Data Sets*. For additional information on recovery processing, see *z/OS DFSMS Using Data Sets*.
118(X'76')	The data set was not properly closed. The data set high-used RBA has been successfully verified. Records may be missing or duplicated. A previous VSAM program may have abnormally ended. You should verify that all of the expected records are in the data set. The catalog will be updated when the data set has been successfully opened for output and then successfully closed. For additional information on recovery processing, see *z/OS DFSMS Using Data Sets*.
128(X'80')	DD statement for this access method control block is missing or invalid.
131(X'83')	An error was detected by VSAM for a media manager CONNECT.

Table 14. OPEN reason codes in the ACBERFLG field of the ACB (continued). OPEN reason codes in the ACBERFLG field of the ACB

Reason code	Meaning
132(X'84')	One of the following errors occurred: • Not enough storage was available for work areas. • The required volume could not be mounted. • A system logic error occurred while VSAM was accessing the job file control block (JFCB). • The format-1 DSCB or the catalog cluster record is invalid. • The user-supplied catalog name does not match the name on the entry. • The user is not authorized to open the catalog as a catalog. • For DFSMStvs: – Unable to connect to the forward recovery log – Unable to write tie-up record to the forward recovery log – Data set cannot be opened because it needs to be forward recovered. – DFSMStvs processing is not available. – For a data set that was previously accessed for Permit Non-RLS Update (PNRLU) processing, an error occurred attempting to write the PNRLU record to the undo log
133(X'85')	Delete Volume processing for RESET(MACRF=RST) failed during open. The DDNAME needs to be freed and reallocated to the data set.
134(X'86')	Invalid UCB address for UCB address conversion.
136(X'88')	Not enough virtual storage space is available in your program's address space for work areas, control blocks, or buffers.
138(X'8A')	A 24-bit UCB address is required for Volume Mount but a 31-bit UCB address was passed.
140(X'8C')	The catalog indicates this data set has an invalid physical record size.
144(X'90')	Uncorrectable I/O error occurred while VSAM reading or writing catalog record.
145(X'91')	An uncorrectable error occurred in the VSAM volume data set (VVDS).
148(X'94')	No record for the data set to be opened was found in the available catalogs, or an unidentified error occurred while VSAM was searching the catalog. For the catalog return code, see system message IDC3009I. For UNIX files, the requested file does not exist.
152(X'98')	Authorization checking failed for one of the following reasons: • The password specified in the access method control block for a specified level of access does not match the password in the catalog for that level of access. • RACF denied access. For the catalog return code, see system message IDC3009I in job output.

Table 14. OPEN reason codes in the ACBERFLG field of the ACB (continued). OPEN reason codes in the ACBERFLG field of the ACB

Reason code	Meaning
160(X'A0')	The operands specified in the ACB or GENCB macro are inconsistent either with each other or with the information in the catalog record. One of these conditions has been detected: • For option ACBRST – Path processing – LSR or GSR • For option ACBICI – LSR or GSR – Key-sequenced data set – Path processing – Sequence set with data – Replicated index – Block size not equal to CI size – Extended format data set • For option ACBUBF – LSR or GSR – ACBCNV not specified – ACBKEY specified – ACBADR specified • For option ACBSDS – LSR or GSR – Path processing – Upgrade processing • For option ACBCBIC – LSR or GSR – ACBICI not specified • For MACRF=RLS, an invalid option has been specified: – Linear data sets, key range data sets, cluster IMBED attribute, user buffering option, or ICI not supported. – MACRF=RLS was specified in addition to MACRF=NSR/LSR/GSR/ICI. – Attempt to open a temporary data set, catalog, VVDS, or system data set for RLS or DFSMStvs access. – Attempt to open a sphere with a specification of BWO=TYPEOTHER,TYPEIMS for RLS access. – For extended-function data sets, ICI is not supported. • For miscellaneous options – Buffer space was specified, but the amount is too small to process the data set. – Volume is not mounted. – VSAM is trying to open an empty data set for input. • For a UNIX file, an invalid option or operand has been specified. – ACBCNV or ACBKEY – ACBSKP – ACBICI – LSR, GSR, or RLS – ACBSTRNO > 1.

Table 14. OPEN reason codes in the ACBERFLG field of the ACB (continued). OPEN reason codes in the ACBERFLG field of the ACB

Reason code	Meaning
164(X'A4')	An uncorrectable I/O error occurred while VSAM was reading the volume label.
167(X'A7')	For MACRF=RLS, open or close processing received an abend while processing the request.
168(X'A8')	The data set was not available for the type of processing that you specified. Or, an attempt was made to open a reusable data set with the reset option while another user had the data set open. The data set might have the INHIBIT attribute specified. The data set cannot be opened for CBUF processing because it was already opened for non-CBUF processing. Or, the data set has conflicting CBUF attributes for the data and index components of the ACB. For MACRF=RLS, an attempt was made to access a data set with NSR/LSR/GSR and the data set is currently accessed by RLS or DFSMStvs, or vice versa. Or, an attempt was made to access the data set with NSR/LSR/GSR and the data set is in lost or retained locks state. For a UNIX file, the file type is not supported (for example, directories are not supported).
169(X'A9')	For MACRF=RLS, an attempt was made to access an ACB for RLS processing on a previous release of DFSMS that does not have RLS function.
170(X'AA')	For MACRF=RLS, an ACB specified a SUBSYSNM name that is already registered to a previous server instance.
171(X'AB')	For MACRF=RLS, required CF cache is unavailable from this system.
172(X'AC')	For MACRF=RLS, the CF cache structure failed.
173(X'AD')	For MACRF=RLS, required CF cache structure is in a quiescing or quiesced state.
174(X'AE')	One of the following errors occurred: • For MACRF=RLS, SUBSYSNM was not specified in the ACB and an attempt was made to open a data set for output to a recoverable sphere. • For DFSMStvs, this can occur if DFSMStvs is not active on the system. • The LOG parameter is ALL but LOGSTREAMID is not specified. (This error code has a different meaning on DFSMS/MVS® 1.4.0 and earlier releases.)
175(X'AF')	For MACRF=RLS, locks have been lost. This is an attempt by a new sharing SUBSYSNM to access a data set for which not all recovery has completed. The open is not successful.
177(X'B1')	For RLS or DFSMStvs, the open is rejected because the sphere is marked VSAM quiesced.
178(X'B2')	For MACRF=RLS, the open is rejected. The sphere is VSAM-quiescing and this is an attempt by a new application.
179(X'B3')	For MACRF=RLS, the open is rejected. The sphere is VSAM-quiescing in preparation for a data set copy.
180(X'B4')	A VSAM catalog specified in JCL either does not exist or is not open, and no record for the data set to be opened was found in any other catalog.
181(X'B5')	For MACRF=RLS, the DISP value specified is not consistent with the DISP value specified by another application that has opened this data set for RLS access. Either this application is requesting DISP=SHR while another application holds DISP=OLD or vice-versa.
182(X'B6')	For MACRF=RLS, the SMSVSAM server is not available.
183(X'B7')	For MACRF=RLS, open, invalid backup-while-open (BWO) flags in the catalog.
184(X'B8')	An uncorrectable I/O error occurred while VSAM was completing an I/O request.
188(X'BC')	The data set that is indicated by the access method control block is not of the type that can be specified by an access method control block. Or the access method control block (ACB) has already been opened or closed.
189(X'BC')	The Exit List (EXLST) is invalid because the length is incorrect.

Table 14. OPEN reason codes in the ACBERFLG field of the ACB (continued). OPEN reason codes in the ACBERFLG field of the ACB

Reason code	Meaning
190(X'BE')	An invalid high-allocated RBA was found in the catalog entry for this data set. The catalog entry is bad and will have to be restored.
192(X'C0')	An unusable data set was opened for output.
193(X'C1')	The interrupt recognition flag (IRF) was detected for a data set opened for output processing. This indicates that DELETE processing was interrupted. The structure of the data set is unpredictable. The access method services DIAGNOSE command may be used to check it for structural errors. For a description of the DIAGNOSE command, see *z/OS DFSMS Access Method Services Commands*.
194(X'C2')	An open of the data component of a compressed format key-sequenced data set is not allowed. For MACRF=RLS, an attempt was made to open an alternate index cluster or an individual component of a KSDS data set. KSDS components cannot be opened for RLS processing.
195(X'C3')	For MACRF=RLS, the SMS Storage Class does not specify a coupling facility CACHESET name.
196(X'C4')	Access to data was requested via an empty path. For MACRF=RLS: • Access to data was requested through an empty path. • Attempt to access a VSAM data set for RLS processing via an Alternate Index which is not part of the Upgrade Set.
197(X'C5')	Catalog indicated RLS recovery required but user's ACB did not specify recovery processing.
198(X'C6')	For MACRF=RLS, an open is rejected because a volume quiesce is in progress or a required volume is marked as “quiesced”.
200(X'C8')	The format-4 DSCB indicates that the volume is unusable.
201(X'C9')	For MACRF=RLS, the sphere is not currently assigned to a CF cache, and there are no CF caches available from this system that could be assigned to the sphere.
202(X'CA')	For MACRF=RLS, a SUBSYSNM violation occurred. The SUBSYSNM name specified is different from the subsystem name registered for this address space.
203(X'CB')	For MACRF=RLS, the JRNAD exit requested for ACB being opened for RLS processing.
204(X'CC')	The ACB MACRF specification is GSR and the caller is not operating in protect key 0 to 7. Or, the ACB MACRF specification is CBIC (Control Blocks in Common) and the caller is not operating in supervisor state with protect key 0 to 7.
205(X'CD')	The ACBCATX option or VSAM volume data set open was specified and the calling program was not authorized.
206(X'CE')	For MACRF=RLS, the LOG parameter that is associated with the base cluster is undefined.
207(X'CF')	RLS SUBSYSNM name contains invalid characters.
208(X'D0')	System logic error.
209(X'D1')	RLS or DFSMStvs open internal logic error detected.
210(X'D2')	RLS or DFSMStvs open requested for non-SMS-managed data set.
212(X'D4')	The ACB MACRF specification is GSR or LSR and the data set requires load mode processing.
213(X'D5')	For MACRF=RLS, the LOG parameter is ALL and the LOGSTREAMID either was not specified or specifies a DFSMStvs system (undo or shunt) log.
214(X'D6')	For DFSMStvs, the maximum logical record length for the data set is larger than the length that DFSMStvs supports for logging.
216(X'D8')	The ACB MACRF specification is GSR or LSR and the key length of the data set exceeds the maximum key length specified in BLDVRP.
220(X'DC')	The ACB MACRF specification is GSR or LSR and the data set's control interval size exceeds the size of the largest buffer specified in BLDVRP.

Table 14. OPEN reason codes in the ACBERFLG field of the ACB (continued). OPEN reason codes in the ACBERFLG field of the ACB

Reason code	Meaning
224(X'E0')	Improved control interval processing is specified and the data set requires load mode processing.
228(X'E4')	The ACB MACRF specification is GSR or LSR and the VSAM shared resource table (VSRT) does not exist (no buffer pool is available).
229(X'E5')	OPEN failed because a BLDVRP or DLVRP is already in progress. A retry of the OPEN is suggested.
230(X'E6')	OPEN failed because the maximum number of alternate indexes (255) has been exceeded.
231(X'E7')	OPEN failed because the maximum number of VSAM control blocks has been exceeded.
232(X'E8')	Reset was specified for a non-reusable data set and the data set is not empty.
236(X'EC')	System logic error.
240(X'F0')	Format-4 DSCB and volume timestamp verification failed during volume mount processing for output processing.
244(X'F4')	The volume containing the catalog recovery area was neither mounted nor verified for output processing.
245(X'F5')	An attempt was made to open a compressed format data set without sufficient hardware, ESCON® channels and concurrent copy capable control units, or a compressed format device was required.
246(X'F6')	The compression management services open or close function failed.
247(X'F7')	An error occurred while retrieving the dictionary token from the extended format cell.
250(X'FA')	DSAB match not found.

VSAM also writes a message to the operator console and the programmer's listing further explaining the error.

CLOSE return and reason codes

When your program receives control after it has issued a CLOSE macro, a return code in register 15 indicates whether all VSAM data sets were closed successfully, as Table 15 shows.

Table 15. Return codes in register 15 after CLOSE. Return codes in register 15 after CLOSE

Return code	Meaning
0(X'0')	All data sets were closed successfully.
4(X'4')	At least one data set (VSAM or non-VSAM) was not closed successfully.

If register 15 contains 4, use SHOWCB to display the ERROR field in each access method control block to determine if a VSAM data set was not closed successfully and the reason it was not. See *z/OS DFSMS Macro Instructions for Data Sets*. Table 16 lists the reason codes that the ERROR field might contain after close processing.

Table 16. CLOSE reason codes in the ACBERFLG field of the ACB. CLOSE reason codes in the ACBERFLG field of the ACB

Reason code	Meaning
0(X'0')	No error (set when register 15 contains 0).
4(X'4')	The data set indicated by the access method control block is already closed.

Table 16. CLOSE reason codes in the ACBERFLG field of the ACB (continued). CLOSE reason codes in the ACBERFLG field of the ACB

Reason code	Meaning
129(X'81')	CLOSE TYPE=T was issued for a VSAM data set that is not open for VSAM processing.
132(X'84')	An uncorrectable I/O error occurred while VSAM was reading the job file control block (JFCB).
136(X'88')	Not enough virtual storage was available in your program's address space for a work area for close processing.
144(X'90')	An uncorrectable I/O error occurred while VSAM was reading or writing a catalog record.
145(X'91')	An uncorrectable error occurred in the VSAM volume data set (VVDS).
148(X'94')	An unidentified error occurred while VSAM was searching the catalog. For a UNIX file, an unidentified error occurred.
167(X'A7')	For MACRF=RLS, abend occurred during open or close processing.
170(X'AA')	For MACRF=RLS, the required CF Cache is unavailable from this system.
172(X'AC')	Close was successful, but DFSMStvs was unable to write a close record to the log of logs, the forward recovery log, or both.
184(X'B8')	An uncorrectable I/O error occurred while VSAM was completing outstanding I/O requests. For a UNIX file, an error occurred while VSAM was flushing output data or disconnecting from the file.
185(X'B9')	LSR/GSR - Error in WRTBFR: I/O for data set not quiesced before WRTBFR TYPE=DS during close processing.
188(X'BC')	The data set indicated by the ACB is not the type that may be specified by an ACB. For MACRF=RLS, an invalid ACB address is specified for close processing.
236(X'EC')	System logic error because the function no longer is supported.
246(X'F6')	A call to compression management services (CMS) failed.

In addition to these reason codes, VSAM writes a message to the operator's console and the programmer's listing further explaining the error.

Control block manipulation macro return and reason codes

The GENCB, MODCB, SHOWCB, and TESTCB macros can be run (unlike the ACB, EXLST, and RPL macros). These macros cause control to be given to VSAM to perform the indicated task. VSAM indicates whether the task was completed by placing a return code in register 15. Table 17 lists the possible return codes after one of these macros runs.

Table 17. Return codes in register 15 after control block manipulation macros. Return codes in register 15 after control block manipulation macros

Return code	Meaning
0(X'0')	Task completed.
4(X'4')	Task not completed.
8(X'8')	An attempt was made to use the execute form of a macro to modify a keyword that is not in the parameter list. (See “Use of list, execute, and generate forms of VSAM macros” on page 132.)

You can cause an error if you specify the operands incorrectly.

When register 15 contains 4, register 0 contains a reason code indicating why VSAM could not perform the task. If you construct the parameter list, register 0 can contain reason codes 1, 2, 3, 10, 14, 20, and 21.

Table 18 describes each reason code returned in register 0.

Table 18. GENCB, MODCB, SHOWCB, and TESTCB reason codes returned in register 0. GENCB, MODCB, SHOWCB, and TESTCB reason codes returned in register 0

Reason code	Applicable macros[1]	Reason VSAM could not perform the task
1(X'1')	G,M,S,T	The request type (generate, modify, show, or test) is invalid.
2(X'2')	G,M,S,T	The block type (access method control block, exit list, or request parameter list) is invalid.
3(X'3')	G,M,S,T	One of the keyword codes in the parameter list is invalid.
4(X'4')	M,S,T	The block at the address indicated is not of the type you indicated (access method control block, exit list, or request parameter list).
5(X'5')	S,T	Access method control block fields were to be shown or tested, but the data set is not open or it is not a VSAM data set.
6(X'6')	S,T	Access method control block information about an index was to be shown or tested, but no index was opened with the data set.
7(X'7')	M,S	An exit list was to be modified, but the list was not large enough to contain the new entry. Or, an exit was to be modified or shown but the specified exit wasn't in the exit list. (With TESTCB, if the specified exit address is not present, you get an unequal condition when you test for it.)
8(X'8')	G	There is not enough virtual storage in your program's address space to generate the access method control blocks, exit lists, or request parameter lists and no work area outside your address space was specified.
9(X'9')	G,S	The work area specified was too small for generation or display of the indicated control block or fields.
10(X'A')	G,M	With GENCB, exit list control block type was specified and you specified an exit without giving an address. With MODCB, exit list control block type was specified and you specified an exit without giving an address. In this case, either active or inactive must be specified, but load cannot be specified.
11(X'B')	M	Either (1) a request parameter list was to be modified, but the request parameter list defines an asynchronous request that is active (that is, no CHECK or ENDREQ has been issued on the request) and thus cannot be modified; or (2) MODCB is already issued for the control block, but has not yet completed.
12(X'C')	M	An access method control block was to be modified, but the data set identified by the access method control block is open and cannot be modified.
13(X'D')	M	An exit list was to be modified, and you attempted to activate an exit without providing a new exit address. Because the indicated exit list does not contain an address for that exit, your request cannot be honored.
14(X'E')	G,M,T	One of the option codes (for MACRF, ATRB, or OPTCD) has an invalid combination of option codes specified (for example, OPTCD=(ADR,SKP)).
15(X'F')	G,S	The work area specified did not begin on a fullword boundary.
16(X'10')	G,M,S,T	A VTAM® keyword or subparameter was specified but the AM=VTAM parameter was not specified. AM=VTAM must be specified to process a VTAM version of the control block.

Table 18. GENCB, MODCB, SHOWCB, and TESTCB reason codes returned in register 0 (continued). GENCB, MODCB, SHOWCB, and TESTCB reason codes returned in register 0

Reason code	Applicable macros[1]	Reason VSAM could not perform the task
19(X'13')	M,S,T	A keyword was specified that refers to a field beyond the length of the control block located at the indicated address. (For example, a VTAM keyword is specified, but the control block it points to is a shorter, non-VTAM block.)
20(X'14')	S	Keywords were specified that apply only if MACRF includes LSR or GSR.
21(X'15')	S,T	The block to be displayed or tested does not exist because the data set is a dummy data set.
22(X'16')	S	AM=VTAM was specified and the RPL FIELDS parameter conflicts with the RPLNIB bit status. Either RPLFIELDS=NIB was specified and the RPLNIB was off, or RPL FIELDS=ARG was specified and the RPLNIB bit was on.
23(X'17')	G	The value specified in the work area length parameter exceeds the 65,535 byte limit.
24(X'18')	S,T	The SMSVSAM server is not available.
25(X'19')	S	LOKEY is not supported for MACRF=RLS.
26(X'1A')	S,T	This request was issued against an ACB open to a different instance of the SMSVSAM server. The OPEN is no longer valid.
27(X'1A')	G,M,S,T	This request was issued in AR ASC mode, home ASC mode. Or the RLS address space had to be accessed and the request was issued in secondary ASC mode.

Note:

1. G=GENCB, M=MODCB, S=SHOWCB, T=TESTCB

Record management return and reason codes

The following record management macros give return codes and reason codes in the feedback area of the RPL: GET, PUT, POINT, ERASE, VERIFY, CHECK, ENDREQ, GETIX, PUTIX, WRKBFR, SCHBFR, VERIFY, VERIFY REFRESH, and WRTBFR.

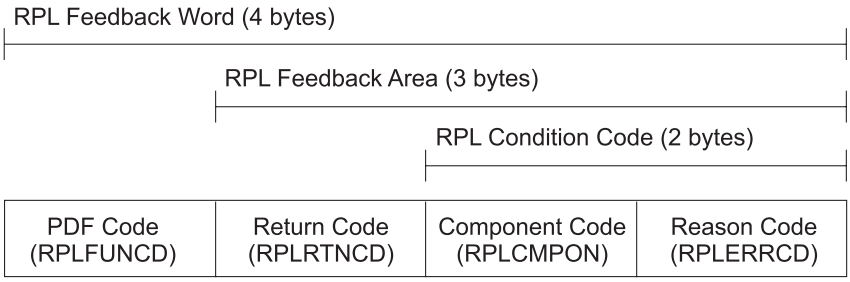
The feedback word in the RPL consists of 4 bytes:

Byte	Description
0	Problem determination function (PDF) code. This code is used to locate the point in VSAM record management where a logical error condition is recognized. A description of the returned PDF code is located in the IDARMRCD macro.
1	RPL return code. This code is returned in register 15.
2	Component code. This code specifies the component being processed when the error occurred.
3	Reason code. This code, when paired with the return code in byte 2, specifies the reason for either a successful completion or an error.

Bytes 2 through 4 make up the RPL feedback area. An explanation of the codes that appear in these three bytes follows.

Bytes 3 and 4 make up the RPL condition code. An explanation of this code also follows.

The field name of each byte appears within parentheses in the following figure.



Return codes (RPLRTNCD)

The meaning of the return code depends on whether the processing is asynchronous or synchronous.

Asynchronous request: After you issue an asynchronous request for access to a data set, VSAM sets a return code in register 15 to indicate whether the request was accepted. Table 19 describes each return code returned in register 15.

Table 19. Return code in register 15 after an asynchronous request. Return code in register 15 after an asynchronous request

Return code (RPLRTNCD)	Meaning
0(X'0')	Request was accepted.
4(X'4')	Request was not accepted because the request parameter list indicated by the request (RPL=address) was active for another request.

If the asynchronous request was accepted, issue a CHECK after doing your other processing. This way VSAM can indicate in register 15 whether the request was completed successfully, set a return code in the feedback area, and exit to any appropriate exit routine.

If the request was not accepted, you should either wait until the other request is complete (for example, by issuing a CHECK on the request parameter list) or terminate the other request (using ENDREQ). Then you can reissue the rejected request.

Synchronous request: After a synchronous request, or a CHECK or ENDREQ macro, the return code in register 15 indicates if the request completed successfully. Table 20 describes each return code returned in register 15.

Table 20. Return code in register 15 after a synchronous request. Return code in register 15 after a synchronous request

Return code (RPLRTNCD)	Meaning
0(X'0')	Request completed successfully.
4(X'4')	Request was not accepted because the request parameter list indicated by the request (RPL=address) was active for another request.

Table 20. Return code in register 15 after a synchronous request (continued). Return code in register 15 after a synchronous request

Return code (RPLRTNCD)	Meaning
8(X'8')	Logical error; specific error is indicated in the RPL feedback area.
12(X'C')	Physical error; specific error is indicated in the RPL feedback area.

Component codes (RPLCMPON)

When a logical or physical error occurs, VSAM uses the RPL component code field to identify the component being processed when the error occurred. VSAM also indicates whether the alternate index upgrade set is correct after the request that failed. You can use the SHOWCB and TESTCB macro to display and test a component code. Table 21 lists the component codes and their meanings.

Table 21. Component codes provided in the RPL. Component codes provided in the RPL

Component code (RPLCMPON)	What was being processed	Upgrade set status
0(X'0')	Base cluster	Correct
1(X'1')	Base cluster	May be incorrect
2(X'2')	Alternate index	Correct
3(X'3')	Alternate index	May be incorrect
4(X'4')	Upgrade set	Correct
5(X'5')	Upgrade set	May be incorrect

The component code (byte 3 of the RPL feedback word) and the reason code (byte 4 of the RPL feedback word) make up the two-byte RPL condition code.

Reason codes (RPLERRCD)

The 0, 8, and 12 return codes in register 15 are paired with reason codes in the RPL feedback area.

The reason codes in the RPL feedback area can be examined with the SHOWCB or TESTCB macro. Code your examination routine immediately following the request macro. Logical errors, physical errors, and reaching the end of the data set all cause VSAM to exit to the appropriate exit routine, if one is provided.

Coordinate error checking in your program with your error-analysis exit routines. If they terminate the program, for instance, you do not need to code a check for an error after a request. But, if a routine returns to VSAM to continue processing, you should check register 15 after a request to determine if there was an error. Even when an error is handled by an exit routine, you may want to modify processing because of the error.

Reason code (successful request): When the request is completed, register 15 indicates the status of the request. A reason code of 0 indicates successful completion. Table 22 on page 203 lists nonzero reason codes and their meanings.

Table 22. Successful-completion reason codes in the feedback area of the request parameter list. Successful-completion reason codes in the feedback area of the request parameter list

Reason code (RPLERRCD) when register 15=0(X'0')	Meaning
0(X'0')	Request completed successfully.
4(X'4')	Request completed successfully. For retrieval, VSAM mounted another volume to locate the record. For storage, VSAM allocated additional space or mounted another volume.
8(X'8')	For GET requests, indicates a duplicate alternate key exists (applies only when accessing a data set using an alternate index that allows non-unique keys). For PUT requests, indicates that a duplicate key was created in an alternate index with the non-unique attribute.
12(X'C')	All buffers, except for the buffer just obtained, might have been modified and might need to be written. IBM suggests that you issue the WRTBFR macro.
16(X'10')	The sequence-set record does not have enough space to allow it to address all the control intervals in the control area that should contain the record. The record was written into a new control area.
20(X'14')	Mass Storage System macros CNVTAD, MNTACQ, and ACQRANGE are no longer supported.
24(X'18')	Buffer found but not modified; no buffer writes performed.
28(X'1C')	Control interval split indicator was detected during an addressed GET NUP request.
32(X'20')	Request deferred for a resource held by the terminated RPL is asynchronous and cannot be restarted. A MRKBFR request is invalid because no candidate buffers can be found. For MACRF=RLS, there are no locks to retain because no update locks exist for this CICS address space, CICS transaction, or SPHERE.
36(X'24')	Possible data set error condition was detected.
40(X'28')	Possible data set error condition was detected.
43(X'2B')	EOV called to retrieve or update the dictionary token in the extended format cell.
44(X'2C')	EOV called to update catalog statistics.

Reason code (logical errors): If a logical error occurs and you have no LERAD routine (or the LERAD exit is inactive), VSAM returns control to your program following the last executed instruction. For information on the LERAD routine, see *z/OS DFSMS Using Data Sets*.

The return code in register 15 indicates a logical error (8), and the RPL feedback area contains a reason code identifying the error. Register 1 points to the RPL.

Some VSAM reason codes for logical errors, used for diagnosis purposes, are shown in *z/OS DFSMSdfp Diagnosis*.

Table 23 on page 204 lists the feedback area reason codes and their meanings. Some of these reason codes in the figure use the term LUWID in their meaning column. For a CICS application, LUWID is a CICS transaction identifier. For a batch job, LUWID is a unique value that RLS assigned to the address space. For DFSMStvs, LUWID is a unique value assigned to each unit of recovery (UR).

Table 23. Logical-error reason codes in the feedback area of the request parameter list. Logical-error reason codes in the feedback area of the request parameter list

Reason code (RPLERRCD) when register 15=8(X'8')	Meaning
4(X'4')	End of data set found (during sequential or skip sequential retrieval), or the search argument is greater than the high key of the data set. Either no EODAD routine is provided, or one is provided, returned to VSAM, and the processing program issued another GET. (For information on the EODAD routine, see *z/OS DFSMS Using Data Sets*.)
8(X'8')	You attempted to store a record with a duplicate key, or there is a duplicate record for an alternate index with the unique key option.
12(X'C')	An attempt was made to perform sequential or skip-sequential processing against a record whose key/record number does not follow the proper ascending/descending sequential order. The error may occur under any one of the following processing conditions: • For a key-sequenced data set – PUT sequential or skip-sequential processing – GET sequential, single-string input only – GET skip-sequential processing and the previous request is not a POINT • For a relative record data set – GET skip-sequential processing – PUT skip-sequential processing
16(X'10')	Record not found, or the RBA is not found in the buffer pool. (If multiple RPL requests are issued for alternate indexes, getting return code 16(X'10') might mean a temporary situation where processing has not been completed on either the base cluster or the associated alternate indexes.)

20(X'14')	Control interval exclusive use conflict. The address of the RPL that owns the resource is placed in the first word in the RPL error message area.
	
	For VSAM RLS and DFSMStvs, another RPL that is used by this LUWID or UR holds an exclusive lock on this record. This code means that there was an intra-LUWID exclusive control conflict. If an RPL message area of sufficient length is specified, the following information is returned.
	`Offset Length Description`
	
	`0 4 Address of RPL in exclusive control`
	`4 1 Flag Byte: - Not used For RLS`
	`X'00'--neither RPL doing a control area split`
	`X'01'--current RPL doing a control area split`
	`X'02'--other RPL doing a control area split`
	
	If this request's RPL specifies a MSGAREA of length 4 bytes or greater, the address of an RPL whose lock on this record caused this request to be rejected is returned in the first 4 bytes of MSGAREA. The application may choose to issue an ENDREQ on that RPL and then reissue this POINT, GET NUP, or GET UPD request.

Table 23. Logical-error reason codes in the feedback area of the request parameter list (continued). Logical-error reason codes in the feedback area of the request parameter list

Reason code (RPLERRCD) when register 15=8(X'8')	Meaning

21(X'15')	For VSAM RLS and DFSMStvs, another LUWID holds an exclusive lock on this record. The combination of one or more LUWIDs waiting for other record locks held by this LUWID and this LUWID waiting for this record lock produced a deadlock.
	
	If an RPL message area of sufficient length (four bytes or longer) is specified, and the requestor is a commit protocol application (for example, CICS), the following information is returned in the RPL message area:
	`Offset Length Description`
	
	`0 4 Address of problem determination area`
	`If you see this error, you are required`
	`to free this area, for example, with:`
	`?STORAGE (RELEASE) where LENGTH=VPDISIZE,`
	`SP=0,KEY=user's key.`

22(X'16')	For VSAM RLS and DFSMStvs, another LUWID holds an exclusive lock on this record. This request waited for the record lock until the timeout interval expired.
	
	If an RPL message area of sufficient length (four bytes or longer) is specified, and the requestor is a commit protocol application (for example, CICS), the following information is returned in the RPL message area:
	`Offset Length Description`
	
	`0 4 Address of problem determination area.`
	`If you see this error, you are required`
	`to free this area, for example, with:`
	`?STORAGE (RELEASE) where LENGTH=VPDISIZE,`
	`SP=0,KEY=user's key.`

24(X'18')	Record resides on a volume that cannot be mounted.
	
	For VSAM RLS and DFSMStvs, another LUWID holds a retained lock on this record. This can occur if the other UR closed the data set with the UR inflight and it was the last close of the data set on the system.
	
	If an RPL message area of sufficient length (four bytes or longer) is specified, the following information is returned in the RPL message area:
	`Offset Length Description`
	
	`0 4 Address of problem determination area.`
	`If you see this error, you are required`
	`to free this area, for example, with:`
	`?STORAGE (RELEASE) where LENGTH=VPDISIZE,`
	`SP=0,KEY=user's key.`
	
	For non-RLS, message area information is not returned.

28(X'1C')	Data set cannot be extended because VSAM cannot allocate additional direct access storage space. Either there is not enough space left to make the secondary allocation request, or you attempted to increase the size of a data set while processing with SHAREOPTIONS=4 and DISP=SHR. For VSAM RLS and DFSMStvs, the error can occur for a GET request when the same error has been issued for a preceding PUT request on the same ACB.
32(X'20')	You specified an RBA that does not give the address of any data record in the data set.

Table 23. Logical-error reason codes in the feedback area of the request parameter list (continued). Logical-error reason codes in the feedback area of the request parameter list

Reason code (RPLERRCD) when register 15=8(X'8')	Meaning
36(X'24')	Key ranges were specified for the data set when it was defined, but no range was specified that includes the record to be inserted.
40(X'28')	Insufficient virtual storage in your address space to complete the request.
44(X'2C')	Work area not large enough for the data record or for the buffer (GET with OPTCD=MVE).
48(X'30')	Invalid options, data set attributes, or processing conditions: • CNV processing • The specified RPL is asynchronous • Chained RPLs • Path processing • Shared resources (LSR/GSR) indeterminate buffer status • Load mode • Fixed-length relative record data set • Data set contains spanned records • User not in key 0 and supervisor state • End-of-volume in process (secondary allocation).
52(X'34')	Invalid options, data set attributes, or processing conditions specified by MVS/DFP™. (See X'34' for a list of the invalid options).
56(X'38')	Error from catalog update at the beginning of a CI/CA split for backup while open. For VSAM RLS and DFSMStvs, this error indicates an invalid reuse of an RLS RPL. This RPL has position established for VSAM RLS and DFSMStvs access to a data set. The application has changed the ACB or the LUWID, or both. VSAM RLS and DFSMStvs do not permit this form of RPL reuse. This error does not change or lose the string's position. Before changing the ACB or LUWID, the application must issue an ENDREQ on the RPL to release the string's position. RPL reuse violation. The RPL request had positioning information from a previous request and the ACB or LUWID specified in the RPL, or both, did not match that of the prior request.
64(X'40')	There is insufficient storage available to add another string dynamically. Or, the maximum number of place holders that can be allocated to the request has been allocated, and a place holder is not available. For VSAM RLS and DFSMStvs, the limit of 1024 outstanding requests for this ACB has been exceeded.
68(X'44')	The application attempted to use a type of processing (output or control interval processing) that was not specified when the data set was opened.
72(X'48')	The application made a keyed request for access to an entry-sequenced data set. Or, the application issued a GETIX or PUTIX to an entry-sequenced data set or fixed-length RRDS. For VSAM RLS and DFSMStvs, the application issued a GETIX or PUTIX. GETIX and PUTIX are not supported by VSAM RLS and DFSMStvs
76(X'4C')	The application issued an addressed or control interval PUT to add to a key-sequenced data set or variable-length RRDS. Or, the application issued a control interval PUT to a fixed-length RRDS.

Table 23. Logical-error reason codes in the feedback area of the request parameter list (continued). Logical-error reason codes in the feedback area of the request parameter list

Reason code (RPLERRCD) when register 15=8(X'8')	Meaning
80(X'50')	The application issued an ERASE request in one of the following situations: • For access to an entry-sequenced data set • For access to an entry-sequenced data set via a path. • With control interval access.
84(X'54')	The application specified OPTCD=LOC in one of the following situations: • For a PUT request. • In the previous request parameter list in a chain of request parameter lists. • For UBF processing.
88(X'58')	The application issued a sequential GET request without being positioned to it. Or, the application changed from addressed access to keyed access without being positioned for keyed-sequential retrieval. No positioning was established for a sequential PUT insert for an RRDS. Or, the application attempted an illegal switch between forward and backward processing.
92(X'5C')	The application issued a PUT for update, an ERASE without a previous GET for update, or a PUTIX without a previous GETIX. For DFSMStvs, this can also mean that the application issued a PUT UPD or an ERASE without the GET UPD in the same unit of recovery. This means that if the application did the GET UPD, the application did a commit or a backout before doing the PUT UPD or ERASE.
96(X'60')	The application attempted to change the prime key or key of reference while making an update. Or, for MACRF=RLS, the PUT NUP request attempted to change the key that a prior IDALKADD request specified.
100(X'64')	The application attempted to change the length of a record while making an addressed update.

Table 23. Logical-error reason codes in the feedback area of the request parameter list (continued). Logical-error reason codes in the feedback area of the request parameter list

Reason code (RPLERRCD) when register 15=8(X'8')	Meaning
104(X'68')	The RPL options are either invalid or conflicting in one of the following ways: • SKP was specified and either KEY was not specified or BWD was specified. • XRBA was not specified in the RPL OPTCD when a GET DIR or a POINT request was issued in ADR or CNV mode with LRD=OFF, and RPLARG points to a nonzero argument (RBA), while processing an extended-addressing data set. • BWD was specified for CNV processing. • FWD and LRD were specified. • Neither ADR, CNV, nor KEY was specified in the RPL. • BFRNO is invalid (less than 1 or greater than the number of buffers in the pool). • WRTBFR, MRKBFR, or SCHBFR was issued, but either TRANSID was greater than 31 or the shared resource option was not specified. • ICI processing was specified, but a request other than a GET or a PUT was issued. • MRKBFR MARK=OUT or MARK=RLS was issued but the RPL did not have a data buffer associated with it. • The RPL specified WAITX, but the ACB did not specify LSR or GSR. • CNV processing is not allowed for compressed data sets. Only VERIFY and VERIFY REFRESH are allowed. • VERIFY was specified for a UNIX file. • BWD or UPD was specified for a UNIX file. • DIR was specified for a UNIX file that is an FIFO or character special file. • Non-key access issued against extended-format, extended-addressing data set when RBA/XRBA is required for positioning.
108(X'6C')	Incorrect RECLEN. Some possible reasons are: 1. RECLEN specified was larger than the maximum allowed, equal to 0, or smaller than the sum of the length and the displacement of the key field. 2. RECLEN was not equal to record (slot) size specified for a fixed-length RRDS. 3. RECLEN was not sufficient to contain the new alternate index key pointer. With nonunique UPGRADE alternate indexes, the record is automatically increased in size each time a record is added to the base cluster and this can cause an incorrect RECLEN. Make sure the maximum RECORDSIZE on the alternate index is large enough for all base pointers it must contain.
112(X'70')	KEYLEN specified was too large or equal to 0.
116(X'74')	During initial data set loading (that is, when records are being stored in the data set the first time it is opened), GET, POINT, ERASE, direct PUT, skip-sequential PUT, or PUT with OPTCD=UPD is not allowed. For initial loading of a fixed length RRDS, the request was other than a PUT insert.
120(X'78')	Request was operating under an incorrect TCB. For example, an end-of-volume call or a GETMAIN macro was necessary to complete the request, but the request was issued from a task other than the one that opened the data set. The request can be resubmitted from the correct task if the new request reestablishes positioning.
124(X'7C')	A request was cancelled for a user JRNAD exit.
128(X'80')	A loop exists in the index horizontal pointer chain during index search processing.

Table 23. Logical-error reason codes in the feedback area of the request parameter list (continued). Logical-error reason codes in the feedback area of the request parameter list

Reason code (RPLERRCD) when register 15=8(X'8')	Meaning
132(X'84')	An attempt was made in locate mode to retrieve a spanned record.
136(X'88')	The application attempted an addressed GET of a spanned record in a key-sequenced data set.
140(X'8C')	The spanned record segment update number is inconsistent.
144(X'90')	Invalid pointer (no associated base record) in an alternate index. If multiple RPL requests are issued for alternate indexes, getting return code 144(X'90') might mean a temporary situation where processing has not been completed on either the base cluster or the associated alternate indexes. For example, you have issued multiple RPL requests including erase requests to the path or base cluster, and got a return code of X'90'. This might be a temporary situation where the base cluster has been erased, but the associated alternate index has not been erased. If you provide a message area using the MSGAREA parameter of the RPL macro, VSAM returns the address of an RPL doing the erase when the return code X'90' was set.
148(X'94')	The maximum number of pointers in the alternate index has been exceeded.
152(X'98')	Not enough buffers are available to process the application request (shared resources only).
156(X'9C')	Invalid control interval detected during keyed processing, an addressed GET UPD request failed because control interval flag was on, or an invalid control interval or index record was detected. The RPL contains the invalid control interval's RBA.
160(X'A0')	One or more candidates were found that have a modified buffer marked to be written. The buffer was left in write status with valid contents. With this condition, it is possible to have other buffers invalidated or found under exclusive control.
168(X'A8')	For MACRF=RLS, the pointer in the RPL to the record is zero.
180(X'B4')	For MACRF=RLS, an invalid request for a nonrecoverable data set.
184(X'B8')	For MACRF=RLS, the application issued an ABEND condition while VSAM was processing this request. The VSAM RLS FRR (Functional Recovery Routine) intercepted the failure and failed the VSAM request with this reason code.
185(X'B9')	For MACRF=RLS, the user task was cancelled while the request was being processed.
186(X'BA')	For MACRF=RLS, an abend occurred in an attempt to access user storage during logging. This might have happened if the application program issued FREEMAIN or STORAGE RELEASE for the buffers.
187(X'BB')	For MACRF=RLS, an error occurred with partial EOV processing.
188(X'BC')	For MACRF=RLS, the sphere is in lost locks state. A record management request was issued by this SUBSYSNM, but these requests are not allowed until the sphere is out of lost locks state.
189(X'BD')	For MACRF=RLS, a lock for the VSAM request required space in the record table, but the table was full. Installation action is needed to modify the CFRM policy and rebuild the lock structure.
190(X'BE')	Partial EOV error.
192(X'C0')	Invalid relative record number.
196(X'C4')	The application issued an addressed request to a fixed-length or variable-length RRDS.

Table 23. Logical-error reason codes in the feedback area of the request parameter list (continued). Logical-error reason codes in the feedback area of the request parameter list

Reason code (RPLERRCD) when register 15=8(X'8')	Meaning
200(X'C8')	The application attempted addressed or control interval access through a path.
204(X'CC')	PUT insert requests (or for VSAM RLS or DFSMStvs, IDALKADD requests) are not allowed in backward mode.
205(X'CD')	For DFSMStvs, indicates that the request was unable to complete because DFSMStvs restarted while the unit of recovery was inflight. To continue processing, the application must issue a commit or a backout and then begin a new unit of recovery. For LSF, indicates invalid CONTOKEN.
206(X'CE')	For DFSMStvs, indicates that the request was rejected because the data set is quiesced or quiescing for copy. Wait for the copy to complete and then retry the request. For NSR, LSR, or GSR, this reason code indicates a validity check error for shareoptions 3,4.
207(X'CF')	Indicates that DFSMStvs processing is currently unavailable because DFSMStvs is quiescing or disabling. Close all data sets to allow the quiesce/disable process to complete.
208(X'D0')	An ENDREQ was issued against an RPL that has an outstanding WAIT against its associated ACB. No ENDREQ processing was done.
209(X'D1')	For DFSMStvs, indicates that the forward recovery log is unavailable because it is disabling. For LSR, indicates cache structure failure.
210(X'D2')	For DFSMStvs, indicates that forward recovery logging failed because the record length is greater than the installation-defined maximum for the log. For shared resources, buffer being invalidated, buffer use chain changing, and so on.
211(X'D3')	For DFSMStvs, this indicates that the forward recovery log is unusable for this system as a result of either a failure by OPEN to complete connect processing to the logstream, or an error occurred while writing to this logstream. See accompanying DFSMStvs logger messages for appropriate action. For LSR, the cache request is purged.
212(X'D4')	During control area split processing, an existing condition prevents the split of the index record. Index or data control interval size, or both, might need to be increased.
213(X'D5')	For DFSMStvs, indicates that the undo log is unavailable for processing. For LSR, no connectivity to the cache structure.
214(X'D6')	For DFSMStvs, indicates that a permanent I/O error was detected in the undo log. For appropriate action, see accompanying DFSMStvs logger messages.
216(X'D8')	For MACRF=RLS, LUWID specified in the RPL does not exist for the subsystem name specified in the ACB.
217(X'D9')	DFSMStvs is unable to complete the request because the resource recovery services (RRS) instance failed and RRS has been restarted. To continue processing, the application must issue a commit or a backout and then begin a new unit of recovery.
218(X'DA')	Unrecognizable return code from SVC109.
220(X'DC')	DFSMStvs was unable to complete the request because RRS is currently unavailable.
224(X'E0')	MRKBFR OUT was issued for a buffer with invalid contents.

Table 23. Logical-error reason codes in the feedback area of the request parameter list (continued). Logical-error reason codes in the feedback area of the request parameter list

Reason code (RPLERRCD) when register 15=8(X'8')	Meaning
228(X'E4')	Caller in cross-memory mode is not in supervisor state, or RPL of caller in SRB or cross-memory mode does not specify LSR, GSR, or SYN processing. For MACRF=RLS, the caller is not in primary ASC mode, the caller is in SRB mode, the caller issued a record management request with an FRR in effect, or the task that opened the ACB is not in the caller task hierarchy.
229(X'E5')	The record length changed during decompression processing.
230(X'E6')	The processing environment was changed by the user of the UPAD exit.
232(X'E8')	UPAD error; ECB was not posted by the user in cross-memory mode.
235(X'EB')	VSAM RLS or DFSMStvs internal error.
236(X'EC')	Validity check error for SHAREOPTIONS 3 or 4.
237(X'ED')	Reserved.
238(X'EE')	Reserved.
239(X'EF')	Reserved.
240(X'F0')	For shared resources, one of the following is being performed: (1)an attempt is being made to obtain a buffer in exclusive control, (2)a buffer is being invalidated, or (3)the buffer use chain is changing. For more detailed feedback, reissue the request.
241(X'F1')	Reserved.
242(X'F2')	Reserved.
243(X'F3')	Reserved.
244(X'F4')	Register 14 stack size is not large enough.
245(X'F5')	Severe error returned by compression management services during a compress call. Additional problem determination is provided in the RPL message area.
246(X'F6')	An error occurred during an expansion of the user record for an extended-function data set. The RPL message area contains additional problem determination.
248(X'F8')	Register 14 return offset went negative.
249(X'F9')	For DFSMStvs, indicates that undo logging failed because the record length is greater than the installation-defined maximum for the log. For LSR XI, invalid vector token.
250(X'FA')	No valid dictionary token exists for the data set. VSAM is unable to decompress the data record.
252(X'FC')	Record-mode processing is not allowed for a linear data set.
253(X'FD')	VERIFY is not a valid function for a linear data set.
254(X'FD')	I/O activity on the data set was not quiesced before WRTBFR TYPE=DS was issued.

When the search argument you supply for a POINT or GET request is greater than the highest key in the data set, the reason code in the feedback area depends on the RPL's OPTCD values, as the following table shows.

This tables shows how the reason code in the feedback area depends on the RPL's OPTCD values.

Request type	RPLs OPTCD options	Reason code when register 15=8(X'8')
POINT	GEN,KEQ	16(X'10')
POINT	GEN,KGE	4(X'4')
POINT	FKS,KEQ	16(X'10')
POINT	FKS,KGE	4(X'4')
GET	GEN,KEQ,DIR	16(X'10')
GET	GEN,KGE,DIR	16(X'10')
GET	FKS,KEQ,DIR	16(X'10')
GET	FKS,KGE,DIR	16(X'10')
GET	GEN,KEQ,SKP	16(X'10')
GET	GEN,KGE,SKP	4(X'4')
GET	FKS,KEQ,SKP	16(X'10')
GET	FKS,KGE,SKP	4(X'4')

Positioning following logical errors: VSAM is unable to maintain positioning after every logical error. Whenever positioning is not maintained following an error request, you must reestablish it before processing resumes.

Positioning can be in one of four states following a POINT or a direct request that found a logical error:

Yes VSAM is positioned at the position in effect before the request in error was issued.

No VSAM is not positioned, because no positioning was established at the time the request in error was issued.

New VSAM is positioned at a new position.

U VSAM is positioned at an unpredictable position.

N/A The reason code is not applicable to the type of processing indicated.

Table 24 shows which positioning state applies to each reason code listed for sequential, direct, and skip-sequential processing. "N/A" indicates the reason code is not applicable to the type of processing indicated.

Table 24. Positioning states for reason codes listed for sequential, direct, and skip-sequential processing. Positioning states for reason codes listed for sequential, direct, and skip-sequential processing

Reason code (RPLERRCD) when register 15=8(8)	Sequential	Direct	Skip-sequential
4 (X'4')	Yes	No	Yes
8 (X'8')[1]	Yes	No	New
12 (X'C')	Yes	N/A	Yes
16 (X'10')	No	No	No
20 (X'14')	U	No[2]	No[2]

Table 24. Positioning states for reason codes listed for sequential, direct, and skip-sequential processing (continued). Positioning states for reason codes listed for sequential, direct, and skip-sequential processing

Reason code (RPLERRCD) when register 15=8(8)	Sequential	Direct	Skip-sequential
21 (X'15')	Yes[3]	New	New
22 (X'16')	Yes[3]	New	New
24 (X'18')	Yes[3]	No	No
28 (X'1C')	Yes	No	Yes
32 (X'20')	No	No	N/A
36 (X'24')	Yes	No	New
40 (X'28')	Yes	No	No
44 (X'2C')	Yes	New	Yes
48 (X'30')	U	U	U
52 (X'34')	U	U	U
56 (X'38')	Yes	Yes	Yes
64 (X'40')	No	No	No
68 (X'44')	Yes	Yes	Yes
72 (X'48')	Yes	Yes	Yes
76 (X'4C')	Yes	Yes	Yes
80 (X'50')	Yes	Yes	Yes
84 (X'54')	Yes	Yes	Yes
88 (X'58')	Yes	Yes	Yes
92 (X'5C')	Yes	Yes	Yes
96 (X'60')	Yes	Yes	Yes
100 (X'64')	Yes	Yes	Yes
104 (X'68')	Yes	New	Yes
108 (X'6C')	Yes	New	Yes
112 (X'70')	Yes	Yes	Yes
116 (X'74')	Yes	Yes	Yes
120 (X'78')	Yes	No	No
124 (X'7C')	No	No	No
128 (X'80')	Yes	No	No
132 (X'84')	Yes	New	Yes
136 (X'88')	No	No	N/A
140 (X'8C')	Yes	New	Yes
144 (X'90')	Yes	Yes	Yes
148 (X'94')	Yes	Yes	Yes
152 (X'98')	Yes	No	No
156 (X'9C')	Yes	No	No
160 (X'A0')	N/A	No	N/A
168 (X'A8')	N/A	N/A	N/A

Table 24. Positioning states for reason codes listed for sequential, direct, and skip-sequential processing (continued). Positioning states for reason codes listed for sequential, direct, and skip-sequential processing

Reason code (RPLERRCD) when register 15=8(8)	Sequential	Direct	Skip-sequential
169 (X'A9')	N/A	N/A	N/A
172 (X'AC')	N/A	N/A	N/A
176 (X'B0')	N/A	N/A	N/A
180 (X'B4')	Yes	Yes	Yes
181 (X'B5')	N/A	N/A	N/A
182 (X'B6')	N/A	N/A	N/A
184 (X'B8')	U	U	U
186 (X'BA')	Yes	Yes	Yes
189 (X'BD')	Yes	New	New
190 (X'BE')	Yes[3]	No	Yes
192 (X'C0')	Yes	Yes	Yes
196 (X'C4')	Yes	Yes	Yes
200 (X'C8')	Yes	Yes	Yes
201 (X'C9')	N/A	N/A	N/A
204 (X'CC')	Yes	Yes	Yes
205 (X'CD')	U	U	U
206 (X'CE')	Yes	Yes	Yes
208 (X'D0')	Yes	Yes	Yes
209 (X'D1')	U	U	U
210 (X'D2')	Yes	Yes	Yes
211 (X'D3')	No	No	No
212 (X'D4')	U	U	U
213 (X'D5')	U	U	U
216 (X'D8')	N/A	N/A	N/A
217 (X'D9')	Yes	Yes	Yes
224 (X'E0')	N/A	No	N/A
228 (X'E4')	No	No	No
229 (X'E5')	New	New	New
230 (X'E6')	Yes	Yes	Yes
232 (X'E8')	No	No	No
235 (X'EB')	U	U	U
236 (X'EC')	Yes	Yes	Yes
237 (X'ED')	U	U	U
238 (X'EE')	U	U	U
239 (X'EF')	U	U	U
240 (X'F0')	Yes	Yes	Yes
241 (X'F1')	No	No	No

Table 24. Positioning states for reason codes listed for sequential, direct, and skip-sequential processing (continued). Positioning states for reason codes listed for sequential, direct, and skip-sequential processing

Reason code (RPLERRCD) when register 15=8(8)	Sequential	Direct	Skip-sequential
242 (X'F2')	U	U	U
243 (X'F3')	No	No	No
244 (X'F4')	U	U	U
245 (X'F5')	New	New	New
246 (X'F6')	New	New	New
248 (X'F8')	U	U	U
249 (X'F9')	Yes	Yes	Yes
250 (X'FA')	New	New	New
251 (X'FB')	U	U	U
252 (X'FC')	No	No	No
253 (X'FD')	No	No	No

Notes:

1. A subsequent GET SEQ will retrieve the duplicate record. However, a subsequent GET SKP for the same key will get a sequence error. In a fixed- or variable-length RRDS, a subsequent PUT SEQ positions to the next slot (whether the slot is empty or not).
2. PUT UPD, DIR or UPD, SKP retains positioning. The RPL contains an RBA that could not be obtained for exclusive control.
3. For MACRF=RLS, position will advance to next record on next request.

Reason code (physical errors): If a physical error occurs and you have no SYNAD routine (or the SYNAD exit is inactive), VSAM returns control to your program following the last executed instruction. The return code in register 15 indicates a physical error (12). The RPL feedback area contains a reason code identifying the error. The RPL message area contains more details about the error. Register 1 points to the request parameter list. The RBA field in the request parameter list gives the relative byte address of the control interval in which the physical error occurred. Table 25 lists the reason codes that can be in the feedback area and explains what each code indicates.

Table 25. Physical-error reason codes in the feedback area of the request parameter list. Physical-error reason codes in the feedback area of the request parameter list

Reason code (RPLERRCD) when register 15=12(X'0C')	Meaning
4(X'4')	Read error occurred for a data set.
8(X'8')	Read error occurred for an index set.
12(X'C')	Read error occurred for a sequence set.
16(X'10')	Write error occurred for a data set.
20(X'14')	Write error occurred for an index set.
24(X'18')	Write error occurred for a sequence set.
36(X'24')	For MACRF=RLS, a CF cache structure connectivity failure occurred.

Table 25. Physical-error reason codes in the feedback area of the request parameter list (continued). Physical-error reason codes in the feedback area of the request parameter list

Reason code (RPLERRCD) when register 15=12(X'0C')	Meaning
40(X'28')	For MACRF=RLS, a CF cache structure failure occurred.
44(X'2C')	For extended format data sets, the suffix for a physical record in the CI at the RBA specified in the RPL is invalid.

Table 26 shows the format of a physical error message. The format and some of the contents of the message are purposely similar to the format and contents of the SYNADAF message, which *z/OS DFSMS Macro Instructions for Data Sets* describes.

Table 26. Physical error message format. Physical error message format

MESSAGE FORMAT FOR NON-RLS PROCESSING			
Field	**Bytes**	**Length**	**Description**
Message Length	0-1	2	Binary value of 128.
	2-3	2	Unused (0)
Message Length-4	4-5	2	Binary value of 124 (provided for compatibility with SYNADAF Message).
	6-7	2	Unused (0)
Address of I/O Buffer	8-11	4	The I/O buffer associated with the data where the error occurred.
The rest of the message is in printable format			
Date	12-16	5	YYDDD (year and day)
	17	1	Comma (,)
Time	18-25	8	HHMMSSTH (hour, minute, second, tenths and hundredths of a second.
	26	1	Comma (,)
RBA	27-38	12	Relative byte address of the record where the error occurred.
	39	1	Comma (,)
Component Type	40	1	"D"(Data) or "I"(Index)
	41	1	Comma (,)
Volume Serial Number	42-47	6	Volume serial number of the volume where the error occurred. For UNIX files, this field contains "*******".
	48	1	Comma (,)
Job Name	49-56	8	Name of the job where the error occurred.
	57	1	Comma (,)
Step Name	58-65	8	Name of the job step where the error occurred.
	66	1	Comma (,)
Unit	67-70	4	The device number where the error occurred. For UNIX files, this field contains "****".
	71	1	Comma (,)
Device Type	72-73	2	The type of device where the error occurred. (Always DA for direct access.)
	74	1	Comma (,)
ddname	75-82	8	The ddname of the DD statement defining the data set where the error occurred.
	83	1	Comma (,)

Table 26. Physical error message format (continued). Physical error message format

MESSAGE FORMAT FOR NON-RLS PROCESSING			
Field	**Bytes**	**Length**	**Description**
Channel	84-89	6	The channel command that received the error in the first two bytes, followed by "-OP" For UNIX files, this field contains the request that resulted in the error. A GET, PUT, CHECK, POINT, or ENDREQ request.

Figure 6 shows information about messages 91–105.

Message 91-105 15 Messages are divided according to ECB completion codes:

&tab;&tab;&tab;X'41' "INCORR LENGTH"
&tab;&tab;&tab; "UNIT EXCEPTION"
&tab;&tab;&tab; "PROGRAM CHECK"
&tab;&tab;&tab; "PROTECTION CHK"
&tab;&tab;&tab; "CHAN DATA CHK"
&tab;&tab;&tab; "CHAN CTRL CHK"
&tab;&tab;&tab; "INTFCE CTRL CHK"
&tab;&tab;&tab; "CHAINING CHK"
&tab;&tab;&tab; "UNIT CHECK"
&tab;&tab;&tab; "SEEK CHECK"

If the type of unit check can be determined, the "UNIT CHECK" message is replaced by one of the following messages:

&tab;&tab;&tab; "CMD REJECT"
&tab;&tab;&tab; "INT REQ"
&tab;&tab;&tab; "BUS OUT CK"
&tab;&tab;&tab; "EQP CHECK"
&tab;&tab;&tab; "DATA CHECK"
&tab;&tab;&tab; "OVER RUN"
&tab;&tab;&tab; "TRACK COND CK"
&tab;&tab;&tab; "COUNT DATA CHK"
&tab;&tab;&tab; "TRACK FORMAT"
&tab;&tab;&tab; "CYLINDER END"
&tab;&tab;&tab; "NO RECORD FOUND"
&tab;&tab;&tab; "FILE PROTECT"
&tab;&tab;&tab; "MISSING A.M."
&tab;&tab;&tab; "OVERFL INCP"
&tab;&tab;&tab;X'48' "PURGED REQUEST"
&tab;&tab;&tab;X'4A' "I/O PREVENTED"
&tab;&tab;&tab;X'4F' "R.HA.R0. ERROR"
&tab;&tab;&tab; "INVALID SUFFIX"

Figure 6. Physical error message format

Table 27 shows information about messages 106–127.

Table 27. Physical error message format. This table shows information about messages 106–127.

Field	Bytes	Length	Description
For any other ECB completion code:			
			"UNKNOWN COND".

Table 27. Physical error message format (continued). This table shows information about messages 106–127.

Field	Bytes	Length	Description
			For UNIX files, this field contains the service that encountered the error, in the form "OMVS-*nnnnnnnn*", in which *nnnnnnnn* is the name of the service.
	106	1	Comma (,)
Physical Direct Access Address	107-120	14	BBCCHHR (bin, cylinder, head, and record) For UNIX files, this field contains the return and reason code from the failing service in the form "*xxxx-yyyyyyyy*" consisting of a 2-byte hexadecimal return code and a 4-byte hexadecimal reason code.
	121	1	Comma (,)
Access Method	122-127	6	"VSAM" For UNIX files, this field contains "VSAM"

MESSAGE FORMAT FOR CF FAILURE WITH VSAM RLS OR DFSMStvs PROCESSING

Field	Bytes	Length	Description
Message Length	0-1	2	Binary value of 128
	2-3	2	Unused (0)
Message Length-4	4-5	2	Binary value of 124 (provided for compatibility with SYNADAF Message).
	6-7	2	Unused (0)
Address of I/O Buffer	8-11	4	The I/O buffer associated with the data where the error occurred.
The rest of the message is in printable format:			
Date	12-16	5	YYDDD (year and day)
	17	1	Comma (,)
Time	18-25	8	HHMMSSTH (hour, minute, second, tenths and hundredths of a second.
	26	1	Comma (,)
RBA	27-38	12	Relative byte address of the record where the error occurred.
	39	1	Comma (,)
Component Type	40	1	"D"(Data) or "I"(Index)
	41	1	Comma (,)
Volume Serial	42-47	6	For MACRF=RLS, this field does not apply and is set to asterisks.
	48	1	Comma (,)
Job Name	49-56	8	Name of the job where the error occurred.
	57	1	Comma (,)
Step Name	58-65	8	Name of the job step where the error occurred.
	66	1	Comma (,)
Unit	67-70	4	For MACRF=RLS, this field does not apply and is set to asterisks.
	71	1	Comma (,)
Device Type	72-73	2	For MACRF=RLS, this field is set to "CS" for CF cache structure.
	74	1	Comma (,)
ddname	75-82	8	The ddname of the DD statement defining the data set where the error occurred.
	83	1	Comma (,)
Channel	84-89	6	For MACRF=RLS, this field is set to "CFREAD" or "CFWRT" indicating if the CF operation is a read or write.
	90	1	Comma (,)

Table 27. Physical error message format (continued). This table shows information about messages 106–127.

Field	Bytes	Length	Description
Message	91-105	15	For MACRF=RLS, you receive either CF structure failure message or loss of connectivity message. &tab;“CF STR FAILURE” &tab;“CF CON FAILURE”
	106	1	Comma (,)
Physical Direct Access Address	107-120	14	14-character cache structure name.
	121	1	Comma (,)
Access Method	122-127	6	“VSAM”

Reason code (server errors)

If a server failure occurs in the SMSVSAM address space for VSAM RLS or DFSMStvs, the return code in register 15 indicates a server error (16). The RPL feedback area contains a reason code identifying the type of server failure. Table 28 lists the reason codes that can be in the feedback area and explains the associated failures.

Table 28. Server failure reason codes in the feedback area of the request parameter list. Server failure reason codes in the feedback area of the request parameter list

Return code (RPLERRCD) when register 15=16(X'10')	Meaning
4(X'4')	VSAM server address space is detected to be inactive, uninitialized, or at a different server instance.
8(X'8')	Server is terminating; CF connection is lost.
12(X'C')	DFSMStvs processing is unavailable, either because DFSMStvs did not initialize or because it is is disabled or quiesced.
16(X'10')	DFSMStvs processing is unavailable because RRS is down.

Return codes from macros used to share resources among data sets

VSAM has a set of macros that allow you to share I/O buffers, I/O related control blocks, and channel programs among VSAM data sets.

BLDVRP return codes

VSAM returns a code in register 15 that indicates if the BLDVRP request was successful. Table 29 describes these return codes.

Table 29. Return codes in register 15 after BLDVRP request. Return codes in register 15 after BLDVRP request

Return code	Meaning
0(X'0')	VSAM completed the request.
4(X'4')	The requested data resource pool or index resource pool already exists in the address space (LSR) or in the system protect key (GSR).
8(X'8')	Insufficient virtual storage space to satisfy request. GETMAIN or ESTAE failed.

Table 29. Return codes in register 15 after BLDVRP request (continued). Return codes in register 15 after BLDVRP request

Return code	Meaning
12(X'C')	Opens have already been issued against the shared buffer pool that BLDVRP is building. **Rule:** As a VSAM user, you are responsible for ensuring that the BLDVRP/DLVRP requests are serialized with the open or close requests. VSAM cannot completely detect the lack of such serialization.
16(X'10')	TYPE=GSR is specified but the program that issued BLDVRP is not in supervisor state with protection key 0 to 7.
20(X'14')	STRNO is less than 1 or greater than 255, or parameters are invalid.
24(X'18')	BUFFERS is specified incorrectly. A size or number is invalid.
32(X'20')	The resource pool already exists above 16 megabytes and the request was for storage below 16 megabytes. Or, the resource pool already exists below 16 megabytes and the request was for storage above 16 megabytes.
36(X'24')	BLDVRP was issued to build an index resource pool but the required corresponding data resource pool does not exist.
40(X'28')	The size for Hiperspace™™ buffers is specified incorrectly. The buffer size must be a multiple of 4K with a maximum size of 32K.
44(X'2C')	Attention: At least one request for Hiperspace buffers was rejected because of insufficient expanded storage. The specific buffer subpools rejected may be located by checking for the BLPBFNHS indicator in the Hiperspace buffer request list. The BLDVRP request was otherwise successful. This return code is also valid for jobs indicating RESTART processing.
45(X'2D')	Attention: All Hiperspace™ creates have failed because no expanded storage was installed on the system. BLDVRP processing continued as if no hiperspace buffers were requested. The BLDVRP request was otherwise successful. This return code is also valid for jobs indicating RESTART processing.
48(X'30')	A buffer size specified for a Hiperspace buffer pool is not equal to any of the buffer sizes specified for the virtual buffer pool.
52(X'34')	Another BLDVRP or DLVRP on the same shared pool is in progress.

DLVRP return codes

VSAM returns a code in register 15 that indicates if the DLVRP request was successful. Table 30 describes these return codes.

Table 30. Return codes in register 15 after a DLVRP request. Return codes in register 15 after a DLVRP request

Return code	Meaning
0(X'0')	VSAM completed the request.
4(X'4')	There is no resource pool to delete.
8(X'8')	Insufficient virtual storage space to satisfy request. GETMAIN or ESTAE failed.
12(X'C')	There is at least one open data set using the resource pool.
16(X'10')	TYPE=GSR is specified, but the program that issued DLVRP is not in supervisor state with protection key 0 to 7.
20(X'14')	Another BLDVRP or DLVRP on the same shared pool is in progress.

End-of-volume return codes

End-of-volume returns a code in register 15 that indicates if the request was successful. Table 31 describes these return codes.

Table 31. Return codes in register 15 after end of volume. Return codes in register 15 after end of volume

Return code	Meaning
0(X'0')	Successful.
4(X'4')	The requested volume could not be mounted.
8(X'8')	The requested amount of space could not be allocated.
12(X'C')	I/O operations were in progress when end-of-volume was requested.
16(X'10')	The catalog could not be updated.

SHOWCAT return codes

VSAM returns a code in register 15 that indicates whether the SHOWCAT request was successful. Table 32 describes these return codes.

Table 32. SHOWCAT return codes. SHOWCAT return codes

Return code	Meaning
0(X'00')	VSAM completed the task.
4(X'04')	The area specified in the AREA operand is too small to display all pairs of fields for the associated objects.
8(X'08')	There is insufficient virtual storage to complete the task. (A GETMAIN failed.)
12(X'0C')	Either the ACB address is invalid, or the VSAM master catalog does not exist, or it is not open.
16(X'10')	The address specified in the AREA operand is outside the partition or address space of the program that issued SHOWCAT.
20(X'14')	The named object or control interval does not exist.
24(X'18')	There was an I/O error in gaining access to the catalog.
28(X'1C')	The control interval number is invalid.
32(X'20')	The catalog record does not describe a C, D, G, I, R, or Y type of object.
36(X'24')	The interrelationship among catalog entries is in error. For example, another type.
40(X'28')	There was an unexpected error code returned from catalog management to the SHOWCAT processor.

Coding VSAM user-written exit routines

This topic covers general guidelines for coding VSAM user-written exit routines and specific information about coding the following routines.

Table of contents

General guidelines for coding exit routines

You can supply VSAM exit routines to do the following tasks:

- Analyze logical errors
- Analyze physical errors
- Perform end-of-data processing
- Record transactions made against a data set
- Perform special user processing
- Perform wait user processing
- Perform user-security verification.

VSAM user-written exit routines are identified by macro parameters in access methods services commands.

You can use the EXLST VSAM macro to create an exit list. EXLST parameters EODAD, JRNAD, LERAD, SYNAD and UPAD are used to specify the addresses of your user-written routines. Only the exits marked active are executed. For information about the EXLST macro, see *z/OS DFSMS Macro Instructions for Data Sets*.

You can use access methods services commands to specify the addresses of user-written routines to perform exception processing and user-security verification processing. For information about exits from access methods services commands, see *z/OS DFSMS Access Method Services Commands*.

Table 33 shows the exit locations available from VSAM.

Table 33. VSAM user-written exit routines. This table shows the exit locations available from VSAM.

Exit routine	When available	Where specified
Batch override	After you issue an IDCAMS SHCDS PERMITNONRLSUPDATE command while there was back out work owed	IGW8PNRU exit in LINKLIB or LPALIB
End-of-data-set	When no more sequential records or blocks are available	EODAD parameter of the EXLST macro
Exception exit	After an uncorrectable input/output error	EXCEPTIONEXIT parameter in access methods services commands
Journalize transactions against a data set	After an input/output completion or error, change to buffer contents, shared or nonshared request, program issues GET, PUT, ERASE, shift in data in a control interval	JRNAD parameter of the EXLST macro
Analyze logical errors	After an uncorrectable logical error	LERAD parameter of the EXLST macro
Wait	For non-cross-memory mode callers using RLS	RLSWAIT parameter of the EXLST macro
Error analysis	After an uncorrectable input/output error	SYNAD parameter of the EXLST macro

Table 33. VSAM user-written exit routines (continued). This table shows the exit locations available from VSAM.

Exit routine	When available	Where specified
User processing	WAIT for I/O completion or for a serially reusable request	UPAD parameter of the EXLST macro
User security verification	When opening a VSAM data set	AUTHORIZATION parameter in access methods services commands

Programming guidelines

Usually, you should observe these guidelines in coding a routine:

- Code your routine reentrant
- Save and restore registers (see individual routines for other requirements)
- Be aware of registers used by the VSAM request macros
- Be aware of the addressing mode (24 bit or 31 bit) in which your exit routine will receive control
- Determine if VSAM or your program should load the exit routine.

Information

When you code VSAM user exit routines, you should have available *z/OS DFSMS Macro Instructions for Data Sets* and *z/OS DFSMS Access Method Services Commands* and be familiar with their contents.

A user exit that is loaded by VSAM will be invoked in the addressing mode specified when the module was link edited. A user exit that is not loaded by VSAM will receive control in the same addressing mode as the issuer of the VSAM record-management, OPEN, or CLOSE request that causes the exit to be taken. It is the user's responsibility to ensure that the exit is written for the correct addressing mode.

Your exit routine can be loaded within your program or by using JOBLIB or STEPLIB with the DD statement to point to the library location of your exit routine.

Multiple request parameter lists or data sets

If the exit routine is used by a program that is doing asynchronous processing with multiple request parameter lists (RPL) or if the exit routine is used by more than one data set, you must code the exit routine so that it can handle an entry made before the previous entry's processing is completed. Saving and restoring registers in the exit routine, or by other routines called by the exit routine, is best accomplished by coding the exit routine reentrant. Another way of doing this is to develop a technique for associating a unique save area with each RPL.

If the LERAD, EODAD, or SYNAD exit routine reuses the RPL passed to it, you should be aware of these :

- The exit routine is called again if the request issuing the reused RPL results in the same exception condition that caused the exit routine to be entered originally.
- The original feedback code is replaced with the feedback code that indicates the status of the latest request issued against the RPL. If the exit routine returns to VSAM, VSAM (when it returns to the user's program) sets register 15 to also indicate the status of the latest request.

- JRNAD, UPAD, and exception exits are extensions of VSAM and, therefore, must return to VSAM in the same processing mode in which they were entered (that is, cross-memory, SRB, or task mode).

Return to a main program

Six exit routines can be entered when your main program issues a VSAM request macro (GET, PUT, POINT, and ERASE) and the macro has not completed: LERAD, SYNAD, EODAD, UPAD, RLSWAIT or the EXCEPTIONEXIT routine. Entering the LERAD, SYNAD, EODAD, or EXCEPTIONEXIT indicates that the macro failed to complete successfully. When your exit routine completes its processing, it can return to your main program in one of two ways:

- The exit routine can return to VSAM (via the return address in register 14); VSAM then returns to your program at the instruction following the VSAM request macro that failed to complete successfully. This is the easier way to return to your program.

  If your error recovery and correction process needs to reissue the failing VSAM macro against the RPL to retry the failing request or to correct it:

  – Your exit routine can correct the RPL (using MODCB), then set a switch to indicate to your main program that the RPL is now ready to retry. When your exit routine completes processing, it can return to VSAM (via register 14), which returns to your main program. Your main program can then test the switch and reissue the VSAM macro and RPL.

  – Your exit routine can issue a GENCB macro to build an RPL, and then copy the RPL (for the failing VSAM macro) into the newly built RPL. At this point, your exit routine can issue VSAM macros against the newly built RPL. When your exit routine completes processing, it can return to VSAM (via register 14), which returns to your main program.

- The exit routine can determine the appropriate return point in your program, then branch directly to that point. Note that when VSAM enters your exit routine, none of the registers contains the address of the instruction following the failing macro.

  You are required to use this method to return to your program if, during the error recovery and correction process, your exit routine issued a GET, PUT, POINT, or ERASE macro that refers to the RPL referred to by the failing VSAM macro. (That is, the RPL has been reissued by the exit routine.) In this case, VSAM has lost track of its reentry point to your main program. If the exit routine returns to VSAM, VSAM issues an error return code.

IGW8PNRU routine for batch override

To prevent damage to a data set, DFSMStvs defers the decision whether to back out a specific record to an installation exit, the batch override exit. This is an optional exit that Transaction VSAM calls when it backs out a unit of recovery (UR) that involves a data set that might have been impacted by the IDCAMS SHCDS PERMITNONRLSUPDATE command. The exit is called once for each affected undo log record for the data set.

The purpose of this exit is to return to DFSMStvs with an indication of whether or not the backout should be applied. The input is an undo log record (mapped by IGWUNLR) and a data set name. The output is a Boolean response of whether or not to do the backout, returned in register 15:

- 0 (zero) means do not back out this record.
- 4 means back out this record.

The exit is given control in the following environment:

- INTERRUPTS enabled
- STATE and KEY problem program state, key 8
- ASC Mode P=H=S, RLS address space
- AMODE, RMODE: No restrictions
- LOCKS: None held
- The exit is reentrant

Register contents

Table 34 gives the contents of the registers when VSAM exits to the IGW8PNRU routine.

Table 34. Contents of registers at entry to IGW8PNRU exit routine. Contents of registers at entry to IGW8PNRU exit routine

Register	Contents
0	Not applicable.
1	Address of IGWUNLR (in key 8 storage).
	Address of an area to be used as an autodata area (in key 8 storage).
3	Length of the autodata area.
4-13	Unpredictable. Register 13, by convention, contains the address of your processing program's 72-byte save area, which must not be used as a save area by the IGW8PNRU routine if it returns control to VSAM.
14	Return address to VSAM.
15	Entry address to the IGW8PNRU routine.

Programming considerations

The following programming considerations apply to the batch override exit:

- The name of this exit must be IGW8PNRU.
- The exit must be loadable from any system that might do peer recovery for another system.
- The IGW8PNRU module is loaded by DFSMStvs and, therefore, must reside in LINKLIB or LPALIB. If the load fails, DFSMStvs issues a message.
- If it does not find the batch override exit, DFSMStvs shunts any UR with a pending backout for a data set that was accessed through PERMITNONRLSUPDATE.
- If your installation needs to fix a code error or enhance the function of the exit, you need to restart DFSMStvs to enable the new exit.
- The exit can issue SVC instructions.

DFSMStvs establishes an ESTAE recovery environment before calling the exit to protect the RLS address space from failures in the exit. If the exit fails or an attempt to invoke it fails, the UR is shunted. A dump is taken, and the exit is disabled until the next DFSMStvs restart, but the server is not recycled. If the exit abended, it might result in a dump with a title like this:

```
DUMP  TITLE=COMPID=?????,CSECT=????????+FFFF,DATE=????????,MAINT
          ID=????????,ABND=0C4,RC=00000000,RSN=00000004
```

If this happens, investigate why the exit abended.

Recommendation: It is possible for this exit to perform other processing, but IBM strongly recommends that the exit not attempt to update any recoverable resources.

When your IGW8PNRU routine completes processing, return to your main program as described in “Return to a main program” on page 224.

EODAD exit routine to process end of data

VSAM exits to an EODAD routine when an attempt is made to sequentially retrieve or point to a record beyond the last record in the data set (one with the highest key for keyed access and the one with the highest RBA for addressed access). VSAM doesn't take the exit for direct requests that specify a record beyond the end. If the EODAD exit isn't used, the condition is considered a logical error (FDBK code X'04') and can be handled by the LERAD routine, if one is supplied. See “LERAD exit routine to analyze logical errors” on page 233.

Register contents

Table 35 gives the contents of the registers when VSAM exits to the EODAD routine.

Table 35. Contents of registers at entry to EODAD exit routine. Contents of registers at entry to EODAD exit routine

Register	Contents
0	Unpredictable.
1	Address of the RPL that defines the request that occasioned VSAM's reaching the end of the data set. The register must contain this address if you return to VSAM.
2-13	Unpredictable. Register 13, by convention, contains the address of your processing program's 72-byte save area, which must not be used as a save area by the EODAD routine if it returns control to VSAM.
14	Return address to VSAM.
15	Entry address to the EODAD routine.

Programming considerations

The typical actions of an EODAD routine are to:

- Examine RPL for information you need, for example, type of data set
- Issue completion messages
- Close the data set
- Terminate processing without returning to VSAM.

If the routine returns to VSAM and another GET request is issued for access to the data set, VSAM exits to the LERAD routine.

If a processing program retrieves records sequentially with a request defined by a chain of RPLs, the EODAD routine must determine whether the end of the data set was reached for the first RPL in the chain. If not, then one or more records have been retrieved but not yet processed by the processing program.

The type of data set whose end was reached can be determined by examining the RPL for the address of the access method control block that connects the program to the data set and testing its attribute characteristics.

If the exit routine issues GENCB, MODCB, SHOWCB, or TESTCB and returns to VSAM, it must provide a save area and restore registers 13 and 14, which are used by these macros.

When your EODAD routine completes processing, return to your main program as described in “Return to a main program” on page 224.

EXCEPTIONEXIT exit routine

You can provide an exception exit routine to monitor I/O errors associated with a data set. You specify the name of your routine via the access method services DEFINE command using the EXCEPTIONEXIT parameter to specify the name of your user-written exit routine.

Register contents

Table 36 gives the contents of the registers when VSAM exits to the EXCEPTIONEXIT routine.

Table 36. Contents of registers at entry to EXCEPTIONEXIT routine. Contents of registers at entry to EXCEPTIONEXIT routine

Register	Contents
0	Unpredictable.
1	Address of the RPL that contains a feedback return code and the address of a message area, if any.
2-13	Unpredictable. Register 13, by convention, contains the address of your processing program's 72-byte save area, which must not be used by the routine if it returns control to VSAM.
14	Return address to VSAM.
15	Entry address to the exception exit routine.

Programming considerations

The exception exit is taken for the same errors as a SYNAD exit. If you have both an active SYNAD routine and an EXCEPTIONEXIT routine, the exception exit routine is processed first.

The exception exit is associated with the attributes of the data set (specified by the DEFINE) and is loaded on every call. Your exit must reside in the LINKLIB and the exit cannot be called when VSAM is in cross-memory mode.

When your exception exit routine completes processing, return to your main program as described in “Return to a main program” on page 224.

For information about how exception exits are established, changed, or nullified, see *z/OS DFSMS Access Method Services Commands*.

JRNAD exit routine to journalize transactions

A JRNAD exit routine can be provided to record transactions against a data set, to keep track of changes in the RBAs of records, and to monitor control interval splits. It is only available for VSAM shared resource buffering. When using the JRNAD exit routine with compressed data sets, all RBA's and data length values returned represent compressed data. For shared resources, you can use a JRNAD exit routine to deny a request for a control interval split. VSAM takes the JRNAD exit each time one of the following occurs:

- The processing program issues a GET, PUT, or ERASE
- Data is shifted right or left in a control interval or is moved to another control interval to accommodate a record's being deleted, inserted, shortened, or lengthened

- An I/O error occurs
- An I/O completion occurs
- A shared or nonshared request is received
- The buffer contents are to be changed.

Restriction: The JRNAD exit is not supported by RLS.

Register contents

Table 37 gives the contents of the registers when VSAM exits to the JRNAD routine.

Table 37. Contents of registers at entry to JRNAD exit routine. Contents of registers at entry to JRNAD exit routine

Register	Contents
0	Byte 0—the subpool-id token created by a BLDVRP request. Bytes 2 - 3—the relative buffer number, that is, the buffer array index within a buffer pool.
1	Address of a parameter list built by VSAM.
2-3	Unpredictable.
4	Address of buffer control block (BUFC).
5-13	Unpredictable.
14	Return address to VSAM.
15	Entry address to the JRNAD routine.

Programming considerations

If the JRNAD is taken for I/O errors, a journal exit can zero out, or otherwise alter, the physical-error return code, so that a series of operations can continue to completion, even though one or more of the operations failed.

The contents of the parameter list built by VSAM, pointed to by register 1, can be examined by the JRNAD exit routine, which is described in Table 38 on page 231.

If the exit routine issues GENCB, MODCB, SHOWCB, or TESTCB, it must restore register 14, which is used by these macros, before it returns to VSAM.

If the exit routine uses register 1, it must restore it with the parameter list address before returning to VSAM. (The routine must return for completion of the request that caused VSAM to exit.)

The JRNAD exit must be indicated as active before the data set for which the exit is to be used is opened, and the exit must not be made inactive during processing. If you define more than one access method control block for a data set and want to have a JRNAD routine, the first ACB you open for the data set must specify the exit list that identifies the routine.

When the data set being processed is extended addressable, the JRNAD exits dealing with RBAs are not taken or are restricted due to the increase in the size of the field required to provide addressability to RBAs which may be greater than 4 GB. The restrictions are for the entire data set without regard to the specific RBA value.

Journalizing transactions: For journalizing transactions (when VSAM exits because of a GET, PUT, or ERASE), you can use the SHOWCB macro to display

information in the request parameter list about the record that was retrieved, stored, or deleted (FIELDS=(AREA,KEYLEN,RBA,RECLEN), for example). You can also use the TESTCB macro to find out whether a GET or a PUT was for update (OPTCD=UPD).

If your JRNAD routine only journals transactions, it should ignore reason X'0C' and return to VSAM; conversely, it should ignore reasons X'00', X'04', and X'08' if it records only RBA changes.

RBA changes: For recording RBA changes, you must calculate how many records there are in the data being shifted or moved, so you can keep track of the new RBA for each. If all the records are the same length, you calculate the number by dividing the record length into the number of bytes of data being shifted. If record length varies, you can calculate the number by using a table that not only identifies the records (by associating a record's key with its RBA), but also gives their length.

You should provide a routine to keep track of RBA changes caused by control interval and control area splits. RBA changes that occur through keyed access to a key-sequenced data set must also be recorded if you intend to process the data set later by direct-addressed access.

Control interval splits: Some control interval splits involve data being moved to two new control intervals, and control area splits normally involve many control intervals' contents being moved. In these cases, VSAM exits to the JRNAD routine for each separate movement of data to a new control interval.

You might also want to use the JRNAD exit to maintain shared or exclusive control over certain data or index control intervals; and in some cases, in your exit routine you can reject the request for certain processing of the control intervals. For example, if you used this exit to maintain information about a data set in a shared environment, you might reject a request for a control interval or control area split because the split might adversely affect other users of the data set.

Figure 7 on page 230 is a skeleton program USERPROG with a user exit routine USEREXIT. It demonstrates the use of the JRNAD exit routine to cancel a request for a control interval or control area split.

```
USERPROG  CSECT
          SAVE(R14,R12)       Standard entry code
          .
          .
          .
          BLDVRP BUFFERS=(512(3)),  Build resource pool              X
                KEYLEN=4,                                            X
                STRNO=4,                                             X
                TYPE=LSR,                                            X
                SHRPOOL=1,                                           X
                RMODE31=ALL
          OPEN (DIRACB)       Logically connect KSDS1
          .
          .
          .
          PUT RPL=DIRRPL      This PUT causes the exit routine USEREXIT
                              to be taken with an exit code X'50' if
                              there is a CI or CA split
          LTR R15,R15         Check return code from PUT
          BZ NOCANCEL         Retcode = 0 if USEREXIT did not cancel
                                        CI/CA split
                                    = 8 if cancel was issued, if
                                        we know a CI or CA split
                                        occurred
          .
          .                   Process the cancel situation
          .
NOCANCEL  .                   Process the noncancel situation
          .
          .
          CLOSE (DIRACB)      Disconnect KSDS1
          DLVRP TYPE=LSR,SHRPOOL=1  Delete the resource pool
          .
          .
          .
          RETURN              Return to caller.
          .
          .
          .
DIRACB    ACB AM=VSAM,                                               X
          DDNAME=KSDS1,                                              X
          BUFND=3,                                                   X
          BUFNI=2,                                                   X
          MACRF=(KEY,DDN,SEQ,DIR,OUT,LSR),                          X
          SHRPOOL=1,                                                 X
          EXLST=EXITLST
*
DIRRPL    RPL AM=VSAM,                                               X
          ACB=DIRACB,                                                X
          AREA=DATAREC,                                              X
          AREALEN=128,                                               X
          ARG=KEYNO,                                                 X
          KEYLEN=4,                                                  X
          OPTCD=(KEY,DIR,FWD,SYN,NUP,WAITX),                         X
          RECLEN=128
*
DATAREC   DC  CL128'DATA RECORD TO BE PUT TO KSDS1'
KEYNO     DC  F'0'            Search key argument for RPL
EXITLST   EXLST AM=VSAM,JRNAD=(JRNADDR,A,L)
JRNADDR   DC  CL8'USEREXIT'   Name of user exit routine
          END                 End of USERPROG


USEREXIT  CSECT               On entry to this exit routine, R1 points
                              to the JRNAD parameter list and R14 points
                              back to VSAM.
          .
          .                   Nonstandard entry code -- need not save
                              the registers at caller's save area and,
          .                   since user exit routines are reentrant for
          .                   most applications, save R1 and R14 at some
          .                   registers only if R1 and R14 are to be
                              destroyed.
```

Parameter list: The parameter list built by VSAM contains reason codes to indicate why the exit was taken, and also locations where you can specify return codes for VSAM to take or not take an action on returning from your routine. The information provided in the parameter list varies depending on the reason the exit was taken. Table 38 shows the contents of the parameter list.

The parameter list will reside in the same area as the VSAM control blocks, either above or below the 16 MB line. For example, if the VSAM data set was opened and the ACB stated RMODE31=CB, the exit parameter list will reside above the 16 MB line. To access a parameter list that resides above the 16 MB line, you will need to use 31-bit addressing.

Table 38. Contents of parameter list built by VSAM for the JRNAD exit. Contents of parameter list built by VSAM for the JRNAD exit

Offset	Bytes	Description
0(X'0')	4	Address of the RPL that defines the request that caused VSAM to exit to the routine.
4(X'4')	4	Address of a 5-byte field that identifies the data set being processed. This field has the format: **4 bytes** Address of the access method control block specified by the RPL that defines the request occasioned by the JRNAD exit. **1 byte** Indication of whether the data set is the data (X'01') or the index (X'02') component.
8(X'8')	4	Variable, depends on the reason indicator at offset 20: **Offset 20** **Contents at offset 8** **X'0C'** The RBA of the first byte of data that is being shifted or moved. **X'20'** The RBA of the beginning of the control area about to be split. **X'24'** The address of the I/O buffer into which data was going to be read. **X'28'** The address of the I/O buffer from which data was going to be written. **X'2C'** The address of the I/O buffer that contains the control interval contents that are about to be written. **X'30'** Address of the buffer control block (BUFC) that points to the buffer into which data is about to be read under exclusive control. **X'34'** Address of BUFC that points to the buffer into which data is about to be read under shared control. **X'38'** Address of BUFC that points to the buffer which is to be acquired in exclusive control. The buffer is already in the buffer pool. **X'3C'** Address of the BUFC that points to the buffer which is to be built in the buffer pool in exclusive control. **X'40'** Address of BUFC which points to the buffer whose exclusive control has just been released. **X'44'** Address of BUFC which points to the buffer whose contents have been made invalid. **X'48'** Address of the BUFC which points to the buffer into which the READ operation has just been completed. **X'4C'** Address of the BUFC which points to the buffer from which the WRITE operation has just been completed.

Table 38. Contents of parameter list built by VSAM for the JRNAD exit (continued). Contents of parameter list built by VSAM for the JRNAD exit

Offset	Bytes	Description
12(X'C')	4	Variable, depends on the reason indicator at offset 20: **Offset 20** **Contents at offset 12** **X'0C'** The number of bytes of data that is being shifted or moved (this number does not include free space, if any, or control information, except for a control area split, when the entire contents of a control interval are moved to a new control interval.) **X'20'** Unpredictable. **X'24'** Unpredictable. **X'28'** Bits 0-31 correspond with transaction IDs 0-31. Bits set to 1 indicate that the buffer that was being written when the error occurred was modified by the corresponding transactions. You can set additional bits to 1 to tell VSAM to keep the contents of the buffer until the corresponding transactions have modified the buffer. **X'2C'** The size of the control interval whose contents are about to be written. **X'30'** Zero. **X'34'** Zero. **X'38'** Zero. **X'3C'** Size of the buffer which is to be built in the buffer pool in exclusive control. **X'48'** Size of the buffer into which the READ operation has just been completed. **X'4C'** Size of the buffer from which the WRITE operation has just been completed.
16(X'10')	4	Variable, depends on the reason indicator at offset 20: **Offset 20** **Contents at offset 16** **X'0C'** The RBA of the first byte to which data is being shifted or moved. **X'20'** The RBA of the last byte in the control area about to be split. **X'24'** The fourth byte contains the physical error code from the RPL FDBK field. You use this fullword to communicate with VSAM. Setting it to 0 indicates that VSAM is to ignore the error, bypass error processing, and let the processing program continue. Leaving it nonzero indicates that VSAM is to continue as usual: terminate the request that occasioned the error and proceed with error processing, including exiting to a physical error analysis routine. **X'28'** Same as for X'24'. **X'2C'** The RBA of the control interval whose contents are about to be written. **X'48'** Unpredictable. **X'4C'** Unpredictable.

Table 38. Contents of parameter list built by VSAM for the JRNAD exit (continued). Contents of parameter list built by VSAM for the JRNAD exit

Offset	Bytes	Description
20(X'14')	1	Indication of the reason VSAM exited to the JRNAD routine: **X'00'** GET request. **X'04'** PUT request. **X'08'** ERASE request. **X'0C'** RBA change. **X'10'** Read spanned record segment. **X'14'** Write spanned record segment. **X'18'** Reserved. **X'1C'** Reserved. The following codes are for shared resources only: **X'20'** Control area split. **X'24'** Input error. **X'28'** Output error. **X'2C'** Buffer write. **X'30'** A data or index control interval is about to be read in exclusive control. **X'34'** A data or index control interval is about to be read in shared status. **X'38'** Acquire exclusive control of a control interval already in the buffer pool. **X'3C'** Build a new control interval for the data set and hold it in exclusive control. **X'40'** Exclusive control of the indicated control interval already has been released. **X'44'** Contents of the indicated control interval have been made invalid. **X'48'** Read completed. **X'4C'** Write completed. **X'50'** Control interval or control area split. **X'54'–X'FF'** Reserved.
21(X'15')	1	JRNAD exit code set by the JRNAD exit routine. Indication of action to be taken by VSAM after resuming control from JRNAD (for shared resources only): **X'80'** Do not write control interval. **X'84'** Treat I/O error as no error. **X'88'** Do not read control interval. **X'8C'** Cancel the request for control interval or control area split.

LERAD exit routine to analyze logical errors

A LERAD exit routine should examine the feedback field in the request parameter list to determine what logical error occurred. What the routine does after determining the error depends on your knowledge of the kinds of things in the processing program that can cause the error.

VSAM does not call the LERAD exit if the RPL feedback code is 64.

Register contents

Table 39 gives the contents of the registers when VSAM exits to the LERAD exit routine.

Table 39. Contents of registers at entry to LERAD exit routine. Contents of registers at entry to LERAD exit routine

Register	Contents
0	Unpredictable.
1	Address of the RPL that contains the feedback field the routine should examine. The register must contain this address if you return to VSAM.
2-13	Unpredictable. Register 13, by convention, contains the address of your processing program's 72-byte save area, which must not be used as a save area by the LERAD routine if the routine returns control to VSAM.
14	Return address to VSAM.
15	Entry address to the LERAD routine. The register does not contain the logical-error indicator.

Programming considerations

The typical actions of a LERAD routine are:

1. Examine the feedback field in the RPL to determine what error occurred
2. Determine what action to take based on error
3. Close the data set
4. Issue completion messages
5. Terminate processing and exit VSAM or return to VSAM.

If the LERAD exit routine issues GENCB, MODCB, SHOWCB, or TESTCB and returns to VSAM, it must restore registers 1, 13, and 14, which are used by these macros. It must also provide two save areas; one, whose address should be loaded into register 13 before the GENCB, MODCB, SHOWCB, or TESTCB is issued, and the second, to separately store registers 1, 13, and 14.

If the error cannot be corrected, close the data set and either terminate processing or return to VSAM.

If a logical error occurs and no LERAD exit routine is provided (or the LERAD exit is inactive), VSAM returns codes in register 15 and in the feedback field of the RPL to identify the error.

When your LERAD exit routine completes processing, return to your main program as described in “Return to a main program” on page 224.

RLSWAIT exit routine

The RLSWAIT exit is entered at the start of the record management request and the request is processed asynchronously under a separate VSAM execution unit. If a UPAD is specified, RLS ignores it.

The exit can do its own wait processing associated with the record management request that is being asynchronously executed. When the record management request is complete, VSAM will post the ECB that the user specified in the RPL.

For RLS, the RLSWAIT exit is entered only for a request wait, never for a resource- or I/O- wait or post as with non-RLS VSAM UPAD.

The RLSWAIT exit is optional. It is used by applications that cannot tolerate VSAM suspending the execution unit that issued the original record management request. The RLSWAIT exit is required for record management request issued in cross memory mode.

RLSWAIT should be specified on each ACB which requires the exit. If the exit is not specified on the ACB via the EXLST, there is no RLSWAIT exit processing for record management requests associated with that ACB. This differs from non-RLS VSAM where the UPAD exit is associated with the control block structure so that all ACBs connected to that structure inherit the exit of the first connector.

To activate RLSWAIT exit processing for a particular record management request, the RPL must specify OPTCD=(SYN,WAITX). RLSWAIT is ignored if the request is asynchronous.

Register contents

Table 40 gives the contents of the registers when RLSWAIT is entered in 31–bit mode.

Table 40. Contents of registers for RLSWAIT exit routine. Contents of registers for RLSWAIT exit routine

Register	Contents
1	Address of the user RPL. If a chain of RPLs was passed on the original record management request, this is the first RPL in the chain.
12	Reserved and must be the same on exit as on entry.
13	Reserved and must be the same on exit as on entry.
14	Return address. The exit must return to VSAM using this register.
15	Address of the RLSWAIT exit.

The RLSWAIT exit must conform to the following restictions:

- The exit must return to VSAM using register 14 and it must return with the same entry environment. That is, under the same execution unit as on entry and with the same cross-memory environment as on entry.
- The exit must not issue any request using the RPL passed in register 1.
- The exit must be reentrant if multiple record management request that use the exit can be concurrently outstanding.

Request environment

VSAM RLS record management requests must be issued in PRIMARY ASC mode and cannot be issued in home, secondary, or AR ASC mode. The user RPL, EXLST, ACB, must be addressable from primary. Open must have been issued from the same primary address space. VSAM RLS record management request task must be the same as the task that opened the ACB, or the task that opened the ACB must be in the task hierarchy (i.e., the record management task was attached by that task that opened the ACB, or by a task that was attached by the task that opened that ACB). VSAM RLS record management requests must not be issued in SRB mode, and must not have functional recovery routine (FRR) in effect.

If the record management request is issued in cross memory mode, then the caller must be in supervisor state and must specify that an RLSWAIT exit is associated with the request (RPLWAITX = ON). The request must be synchronous.

The RLSWAIT exit is optional for non-cross memory mode callers.

The RLSWAIT exit, if specified, is entered at the beginning of the request and VSAM processes the request asynchronously under a separate execution unit. VSAM RLS does not enter the RLSWAIT exit for post processing.

VSAM assumes that the ECB supplied with the request is addressable form both home and primary, and that the key of the ECB is the same as the key of the record management caller.

SYNAD exit routine to analyze physical errors

VSAM exits to a SYNAD routine if a physical error occurs when you request access to data. It also exits to a SYNAD routine when you close a data set if a physical error occurs while VSAM is writing the contents of a buffer out to direct-access storage.

Register contents

Table 41 gives the contents of the registers when VSAM exits to the SYNAD routine.

Table 41. Contents of registers at entry to SYNAD exit routine. Contents of registers at entry to SYNAD exit routine

Register	Contents
0	Unpredictable.
1	Address of the RPL that contains a feedback return code and the address of a message area, if any. If you issued a request macro, the RPL is the one pointed to by the macro. If you issued an OPEN, CLOSE, or cause an end-of-volume to be done, the RPL was built by VSAM to process an internal request. Register 1 must contain this address if the SYNAD routine returns to VSAM.
2-13	Unpredictable. Register 13, by convention, contains the address of your processing program's 72-byte save area, which must not be used by the SYNAD routine if it returns control to VSAM.
14	Return address to VSAM.
15	Entry address to the SYNAD routine.

Programming considerations

A SYNAD routine should typically:

- Examine the feedback field in the request parameter list to identify the type of physical error that occurred.
- Get the address of the message area, if any, from the request parameter list, to examine the message for detailed information about the error
- Recover data if possible
- Print error messages if uncorrectable error
- Close the data set
- Terminate processing.

The main problem with a physical error is the possible loss of data. You should try to recover your data before continuing to process. Input operation (ACB MACRF=IN) errors are generally less serious than output or update operation (MACRF=OUT) errors, because your request was not attempting to alter the contents of the data set.

If the routine cannot correct an error, it might print the physical-error message, close the data set, and terminate the program. If the error occurred while VSAM was closing the data set, and if another error occurs after the exit routine issues a CLOSE macro, VSAM doesn't exit to the routine a second time.

If the SYNAD routine returns to VSAM, whether the error was corrected or not, VSAM drops the request and returns to your processing program at the instruction following the last executed instruction. Register 15 is reset to indicate that there was an error, and the feedback field in the RPL identifies it.

Physical errors affect positioning. If a GET was issued that would have positioned VSAM for a subsequent sequential GET and an error occurs, VSAM is positioned at the control interval next in key (RPL OPTCD=KEY) or in entry (OPTCD=ADR) sequence after the control interval involved in the error. The processing program can therefore ignore the error and proceed with sequential processing. With direct processing, the likelihood of reencountering the control interval involved in the error depends on your application.

If the exit routine issues GENCB, MODCB, SHOWCB, or TESTCB and returns to VSAM, it must provide a save area and restore registers 13 and 14, which these macros use.

See “Example of a SYNAD user-written exit routine” for the format of a physical-error message that can be written by the SYNAD routine.

When your SYNAD exit routine completes processing, return to your main program as described in “Return to a main program” on page 224.

If a physical error occurs and no SYNAD routine is provided (or the SYNAD exit is inactive), VSAM returns codes in register 15 and in the feedback field of the RPL to identify the error. For a description of these return codes, see “Understanding VSAM macro return and reason codes” on page 190.

Example of a SYNAD user-written exit routine

The example in Figure 8 on page 238 demonstrates a user-written exit routine. It is a SYNAD exit routine that examines the FDBK field of the RPL checking for the type of physical error that caused the exit. After the checking, special processing can be performed as necessary. The routine returns to VSAM after printing an appropriate error message on SYSPRINT.

```
ACB1     ACB    EXLST=EXITS

EXITS    EXLST  SYNAD=PHYERR
RPL1     RPL    ACB=ACB1,
                MSGAREA=PERRMSG,
                MSGLEN=128

PHYERR   USING  *,15            This routine is nonreentrant.
*                               Register 15 is entry address.
         .
         .                      Save caller's register
                                (1, 13, 14).
         LA     13,SAVE         Point to routine's save area.
         .
         .                      If register 1=address of RPL1,
         .                      then error did not occur for a
                                CLOSE.
         SHOWCB RPL=RPL1,
                FIELDS=FDBK,
                AREA=ERRCODE,
                LENGTH=4
*                               Show type of physical error.
         .
         .                      Examine error, perform special
         .                      processing.
         PUT    PRTDCB,ERRMSG   Print physical error message.
         .
         .                      Restore caller's registers
         .                      (1, 13, 14).

         BR     14              Return to VSAM.
         .
         .
         .
ERRCODE  DC     F'0'            RPL reason code from SHOWCB.
PERRMSG  DS     0XL128          Physical error message.

         DS     XL12            Pad for unprintable part.

ERRMSG   DS     XL116           Printable format part of message.
         .
         .
         .
PRTDCB   DCB    .....           QSAM DCB.

SAVE     DS     18F             SYNAD routine's save area.

SAVREG   DS     3F              Save registers 1, 13, 14.
```

Figure 8. Example of a SYNAD exit routine

UPAD exit routine for user processing

VSAM calls the UPAD routine only when the request's RPL specifies OPTCD=(SYN, WAITX) and the ACB specifies MACRF=LSR or MACRF=GSR, or MACRF=ICI. VSAM CLOSE can also cause a UPAD exit to be taken to post a record-management request deferred for VSAM internal resource. VSAM takes the UPAD exit to wait for I/O completion or for a serially reusable resource and the UPAD can also be taken to do the corresponding post processing subject to conditions listed in Table 42 on page 239.

If you are executing in cross-memory mode, you must have a UPAD routine and RPL must specify WAITX. *z/OS DFSMS Using Data Sets* describes cross-memory mode. The UPAD routine is optional for non-cross-memory mode.

Table 42 describes the conditions in which VSAM calls the UPAD routine for synchronous requests with shared resources. UPAD routine exits are taken only for synchronous requests with shared resources or improved control interval processing (ICI).

Table 42. Conditions when exits to UPAD routines are taken. Conditions when exits to UPAD routines are taken

XMM	Sup. state	UPAD needed	I/O wait	I/O post	Resource wait	Resource post
Yes	Yes	Yes	UPAD taken	UPAD taken	UPAD taken	UPAD taken
No	Yes	No	UPAD taken if requested	UPAD not taken even if requested	UPAD taken if requested	UPAD taken if either resource owner or the deferred request runs in XM mode
No	No	No	UPAD taken if requested	UPAD not taken even if requested	UPAD taken if requested	UPAD taken if either resource owner or the deferred request runs in XM mode

Notes:

- You must be in supervisor state when you are in cross-memory mode or SRB mode.
- RPL WAITX is required if UPAD is required. A UPAD routine can be taken only if RPL specifies WAITX.
- VSAM gives control to the UPAD exit in the same primary address space of the VSAM record management request. However, VSAM can give control to UPAD with home and secondary ASIDs different from those of the VSAM record management request because the exit was set up during OPEN.
- When a UPAD exit is taken to do post processing, make sure the ECB is marked posted before returning to VSAM. VSAM does not check the UPAD return code and does not do post after UPAD has been taken. For non-cross-memory task mode only, if the UPAD exit taken for wait returns with ECB not posted, VSAM issues a WAIT SVC.
- The UPAD exit must return to VSAM in the same address space, mode, state, and addressing mode, and under the same TCB or SRB from which the UPAD exit was called. Registers 1, 13, and 14 must be restored before the UPAD exit returns to VSAM.
- ICI does not require UPAD for any mode. Resource wait and post processings do not apply to ICI.

RLS ignores the UPAD exit.

Register contents

Table 43 shows the register contents passed by VSAM when the UPAD exit routine is entered.

Table 43. Contents of registers at entry to UPAD exit routine. Contents of registers at entry to UPAD exit routine

Register	Contents
0	Unpredictable.
1	Address of a parameter list built by VSAM.
2-12	Unpredictable.
13	Reserved.
14	Return address to VSAM.
15	Entry address of the UPAD routine.

Programming considerations

The UPAD exit routine must be active before the data set is opened. The exit must not be made inactive during processing. If the UPAD exit is desired and multiple ACBs are used for processing the data set, the first ACB that is opened must specify the exit list that identifies the UPAD exit routine.

You can use the UPAD exit to examine the contents of the parameter list built by VSAM, pointed to by register 1. Table 44 describes this parameter list.

Table 44. Parameter list passed to UPAD routine. Parameter list passed to UPAD routine

Offset	Bytes	Description
0(X'0')	4	Address of user's RPL; address of system-generated RPL if UPAD is taken for CLOSE processing or for an alternate index through a path.
4(X'4')	4	Address of a 5-byte data set identifier. The first four bytes of the identifier are the ACB address. The last byte identifies the component; data (X'01'), or index (X'02').
8(X'8')	4	Address of the request's ECB.
12(X'0C')	4	Reserved.
12(X'10')	1	UPAD flags: Bit 0 = ON: Wait for resource Bit 0 = OFF: Wait for I/O (Bit 0 is only applicable to UPAD taken for wait processing.) Lower 7 bits are reserved.
16(X'11')	4	Reserved.
20(X'14')	1	Reason code: X'00' VSAM to do wait processing X'04' UPAD to do post processing X'08'–X'FC' Reserved

If the UPAD exit routine modifies register 14 (for example, by issuing a TESTCB), the routine must restore register 14 before returning to VSAM. If register 1 is used, the UPAD exit routine must restore it with the parameter list address before returning to VSAM.

The UPAD routine must return to VSAM under the same TCB from which it was called for completion of the request that caused VSAM to exit. The UPAD exit routine cannot use register 13 as a save area pointer without first obtaining its own save area.

The UPAD exit routine, when taken before a WAIT during LSR or GSR processing, might issue other VSAM requests to obtain better processing overlap (similar to asynchronous processing). However, the UPAD routine must not issue any synchronous VSAM requests that do not specify WAITX, because a started request might issue a WAIT for a resource owned by a starting request.

If the UPAD routine starts requests that specify WAITX, the UPAD routine must be reentrant. After multiple requests have been started, they should be synchronized by waiting for one ECB out of a group of ECBs to be posted complete rather than

waiting for a specific ECB or for many ECBs to be posted complete. (Posting of some ECBs in the list might be dependent on the resumption of some of the other requests that entered the UPAD routine.)

If you are executing in cross-memory mode, you must have a UPAD routine and RPL must specify WAITX. When waiting or posting of an event is required, the UPAD routine is given control to do wait or post processing (reason code 0 or 4 in the UPAD parameter list).

User-security-verification routine

If you use VSAM password protection, you can also have your own routine to check a requester's authority. Your routine is invoked from OPEN, rather than via an exit list. VSAM transfers control to your routine, which must reside in SYS1.LINKLIB, when a requester gives a correct password other than the master password.

Recommendation: Do not use VSAM password protection. Instead, use RACF or an equivalent product.

Through the access method services DEFINE command with the AUTHORIZATION parameter you can identify your user-security-verification routine (USVR) and associate as many as 256 bytes of your own security information with each data set to be protected. The user-security-authorization record (USAR) is made available to the USVR when the routine gets control. You can restrict access to the data set as you choose. For example, you can require that the owner of a data set give ID when defining the data set and then permit only the owner to gain access to the data set.

If the USVR is being used by more than one task at a time, you must code the USVR reentrant or develop another method for handling simultaneous entries.

When your USVR completes processing, it must return (in register 15) to VSAM with a return code of 0 for authority granted or not 0 for authority withheld in register 15. Table 45 gives the contents of the registers when VSAM gives control to the USVR.

Table 45. Communication with user-security-verification routine. Communication with user-security-verification routine

Register	Contents
0	Unpredictable.
1	Address of a parameter list with the following format: 44 bytes Name of the data set for which authority to process is to be verified (the name you specified when you defined it with access method services) 8 bytes Prompting code (or 0's). 8 bytes Owner identification (or 0's). 8 bytes The password that the requester gave (it has been verified by VSAM). 2 bytes Length of the user-security-authorization routine (in binary). – The user-security-authorization.

Table 45. Communication with user-security-verification routine (continued). Communication with user-security-verification routine

Register	Contents
2-13	Unpredictable.
14	Return address to VSAM.
15	Entry address to the USVR. When the routine returns to VSAM, it indicates by the following codes in register 15 if the requester has been authorized to gain access to the data set: 0 Authority granted. not 0 Authority withheld.

Chapter 4. Programming applications to use DVSMStvs

This topic contains Programming Interface information.

Application programming tasks for DFSMStvs include modifying existing applications to use, designing and coding new applications, and handling DFSMStvs error codes.

Modifying applications to use DFSMStvs

You can modify an application to use DFSMStvs by specifying RLS in the JCL or the access control block (ACB) and having the application access a recoverable data set using either open for input with CRE or open for output from a batch job. Before modifying your existing applications, you need to determine which applications can use DFSMStvs effectively and how to change an application that currently uses nonshared resources to use DFSMStvs. For information about which applications to modify and how to modify them, see *z/OS DFSMStvs Planning and Operating Guide*.

Designing and coding applications to use DFSMStvs

For information about designing and coding applications to use DFSMStvs, see *z/OS DFSMStvs Planning and Operating Guide*.

Handling DFSMStvs error codes

This topic lists DFSMStvs error codes in return code and reason code tables. All errors listed are failures unless otherwise indicated. Some reason codes descriptions might appear more than once because each reason code also uniquely identifies the module that issued it.

Reason codes use a standard format:

- *ccmmrrrr cc*: two-digit hexadecimal component identifier. For DFSMStvs, this is either 6F for Recoverable File Services or 70 for DFSMStvs Logging Services.
- *mm*: two-digit hexadecimal module identifier.
- *rrrr*: four-digit unique reason code

Module identifiers

Each reason code uniquely identifies the module that issued it. Table 46 lists the two-digit hexadecimal module identifiers.

Table 46. Reason code module identifiers. This table lists the two-digit hexadecimal module identifiers.

Module	Identifier	Area
IGW8IIN1	01	Initialization
IGW8RRES	03	Restart
IGW8OODS	04	Open
IGW8OCDS	05	Close module identifier area
IGW8FCRB	06	Front end processing
IGW8FEXP	07	Front end processing

Table 46. Reason code module identifiers (continued). This table lists the two-digit hexadecimal module identifiers.

Module	Identifier	Area
IGW8FCLN	08	Front-end processing clean-up routine
IGW8RECE	09	End context exit
IGW8REOM	0A	End-of-memory exit
IGW8RCHK	0B	State check
IGW8RPRP	0C	Prepare
IGW8RPR1	0D	Prepare
IGW8RCMT	0E	Commit
IGW8RCMS	0F	Commit
IGW8RBOU	10	Backout
IGW8RIOM	11	Backout
IGW8RBOS	12	Backout
IGW8ROP2	15	SMSVSAM server open
IGW8RCLS	16	SMSVSAM server close
IGW8RMSG	17	Message processing
IGW8RCSE	18	Context switch exit
IGW8RNFY	19	Notification exit
IGW8RQUI	1A	Quiesce exit
IGW8RCEF	1B	Context services exit failed exit
IGW8IIN2	1C	Initialization
IGW8RVTV	1D	Sync point processing
IGW8RDSN	1E	Shunt processing
IGW8PCLN	1F	Peer recovery processing
IGW8QPOP	20	Quiesce processing or shunt processing
IGW8FLBI	21	Front end processing
IGW8FLAI	22	Front end processing
IGW8RPC1	23	PC router
IGW8RPC2	24	PC router
IGW8RPC3	25	PC router
IGW8FLPA	26	Front end processing
IGW8RCHN	27	RPL chain processing
IGW8FMSG	28	Message processing
IGW8XMSG	29	Message processing
IGW8ROAE	2A	Only agent exit
IGW8SLUD	2B	Shunt processing
IGW8FEPL	2C	Front end processing
IGW8QEXR	2D	Quiesce exit
IGW8QEXP	2E	Quiesce exit
IGW8QEXT	2F	Quiesce exit
IGW8QEXR	30	Quiesce exit
IGW8CDLG	31	Command processing

Table 46. Reason code module identifiers (continued). This table lists the two-digit hexadecimal module identifiers.

Module	Identifier	Area
IGW8CDTV	32	Command processing
IGW8CDJB	33	Command processing
IGW8CDDS	35	Command processing
IGW8CVLG	36	Command processing
IGW8CVTV	37	Command processing
IGW8CSQT	38	Command processing
IGW8CSAK	39	Command processing
IGW8CDUR	3B	Command processing
IGW8RREF	3C	Exit failed exit
IGW8OLOG	3F	Open
IGW8RRTS	40	Restart
IGW8IIN6	41	Initialization
IGW8RRMC	42	Backout
IGW8SRBC	43	Shunt processing
IGW8SRUR	44	Shunt processing
IGW8SPBC	45	Shunt processing - base cluster
IGW8SPUR	46	Shunt processing - unit of recovery (UR)
IGW8SLBC	47	Shunt processing - base cluster
IGW8SLUR	48	Shunt processing - unit of recovery (UR)
IGW8SLUC	49	Shunt processing
IGW8IIN7	4A	Initialization
IGW8CVT1	4B	Command processing
IGW8FTSK	4C	Front end processing
IGW8RCID	4D	Commit
IGW8RBID	4E	Backout
IGW8RAKP	4F	Activity keypointing
IGW8RCSE	50	Context switch exit
IGW8QSHN	51	Redrive shunt processor
IGW8IRLB	52	Initialization
IGW8RSRB	53	SRB scheduling
IGW8RSR1	54	SRB scheduling
IGW8RFRR	55	SRB Recovery routine
IGW8RSCH	56	Schedule SRB
IGW8RCLN	57	Sync point processing clean-up routine
IGW8ISTX	58	Read undo log
IGW8IRS1	59	Read undo log
IGW8IRS2	5A	Read undo log
IGW8PIN1	5B	Read undo log
IGW8PIN2	5C	Read undo log
IGW8OPNR	5D	Open data set exit

Table 46. Reason code module identifiers (continued). This table lists the two-digit hexadecimal module identifiers.

Module	Identifier	Area
IGW8REOX	5E	End of task, job step, or memory
IGW8RCS1	5F	Context switch
IGW8RSR2	60	SRB processing
IGW8RSR3	61	SRB processing
IGW8RSR4	62	SRB processing
IGW8RSHN	63	Shunt processing

Initialization reason codes

Initialization reason codes

Reason Code	Description
6F021000	IGW8IIN1: Initialization was unable to get main storage for the DFSMStvs master information block (TMIB).
6F021001	IGW8IIN1: Generate message failure
6F021002	IGW8IIN1: Instance name failure
6F021003	IGW8IIN1: Initialization invoked the service queue manager (SQM) to submit a task, and SQM returned an error.
6F021004	IGW8IIN1: Initialization was unable to create a storage pool for DFSMStvs context unit-of-recovery blocks (CURBs).
6F02100F	IGW8IIN1: Initialization attempted to set the token for the DFSMStvs master information block (TMIB) and received an error.
6F021010	IGW8IIN1: VRGB extract token failure
6F021011	IGW8IIN1: Initialization attempted to extract the token for the VSAM record-level sharing (RLS) master information block (VMIB) and received an error.
6F021012	IGW8IIN1: Initialization was unable to create a storage pool for DFSMStvs open block extensions (TOBEs)
6F021013	IGW8IIN1: Initialization was unable to create a storage pool for DFSMStvs open block extension headers (TOBHs)
6F021014	IGW8IIN1: DFSMStvs did not initialize because no DFSMStvs instance name was found for this system
6F021016	IGW8IIN1: Initialization was unable to create a storage pool for DFSMStvs data set lists (TDSLs)
6F021017	IGW8IIN1: Initialization was unable to initialize the DFSMStvs latch in the TMIB
6F021018	IGW8IIN1: ACB pool failure
6F021019	IGW8IIN1: Initialization detected an error while converting from ESTAE to FRR
6F02101A	IGW8IIN1: Initialization detected an error while converting from FRR to ESTAE
6F02101B	IGW8IIN1: Initialization received an error while attempting to get main arguments
6F021029	IGW8IIN1: MDTA pool failure
6F021040	IGW8IIN1: TMIB DFSMStvs restart latch failure
6F021047	IGW8IIN1: QFET pool failure
6F021048	IGW8IIN1: QEXT pool failure
6F021049	IGW8IIN1: TMIB set ATOKEN failure
6F021051	IGW8IIN1: QFET SQM submit failure
6F021054	IGW8IIN1: IFGQ pool failure
6F021055	IGW8IIN1: IGWLOGS pool failure
6F02105E	IGW8IIN1: QDSN pool failure
6F021073	IGW8IIN1: Initialize shunt latch failure

Initialization reason codes

Reason Code	Description
6F021077	IGW8IIN1: Initialize indoubt latch failure
6F021086	IGW8IIN1: CSV query failure
6F02108B	IGW8IIN1: Initialize allocation/open latch failure
6F021091	IGW8IIN1: CSV query failure
6F021098	IGW8IIN1: Initialize open/close latch failure
6F02109B	IGW8IIN1: TMIB command latch failure
6F0210F7	IGW8IIN1: Initialization invoked the service queue manager (SQM) to create a task, and SQM returned an error.
6F1C1000	IGW8IIN2: Initialization was unable to get main storage for the DFSMStvs master information block (TMIB)
6F1C1001	IGW8IIN2: Generate message failure
6F1C1005	IGW8IIN2: Initialization attempted to connect to the SHCDS and received an unexpected error
6F1C1006	IGW8IIN2: Initialization attempted to disconnect from the SHCDS and received an unexpected error
6F1C1007	IGW8IIN2: Initialization attempted to add a system line to the SHCDS and received an unexpected error
6F1C1008	IGW8IIN2: Initialization invoked storage management locking services (SMLS) to identify itself and received an unexpected error
6F1C1009	IGW8IIN2: Initialization detected an error while trying to FREEMAIN the DFSMStvs master information block (TMIB)
6F1C100A	IGW8IIN2: Initialization invoked the context services CRGGRM function to register as a resource manager and received an error
6F1C100B	IGW8IIN2: Initialization invoked the context services CRGSEIF function to register its exit information and received an error
6F1C100C	IGW8IIN2: CRGSEIFCNT failure
6F1C100D	IGW8IIN2: Initialization invoked the RRS (resource recovery services) ATRIRLN function to retrieve log names and received an error
6F1C100E	IGW8IIN2: Initialization invoked the RRS (resource recovery services) ATRISLN function to set its log name and received an error
6F1C100F	IGW8IIN2: TMIB set GTOKEN failure
6F1C1010	IGW8IIN2: Initialization attempted to extract the token for the VSAM RLS global block (VRGB) and received an error
6F1C1015	IGW8IIN2: Initialization detected an error during DFSMStvs logger initialization
6F1C1019	IGW8IIN2: Convert to FRR failure
6F1C101A	IGW8IIN2: Convert to ESTAE failure
6F1C101C	IGW8IIN2: Initialization received an error while attempting FREEMAIN arguments
6F1C101D	IGW8IIN2: Initialization detected an error while trying to obtain the DFSMStvs latch in the TMIB
6F1C101E	IGW8IIN2: Initialization detected an error while trying to release the DFSMStvs latch in the TMIB
6F1C101F	IGW8IIN2: Context services function CRGSEIF context failure
6F1C1020	IGW8IIN2: Initialization invoked the RRS (resource recovery services) ATRBRS function to begin restart and received an error
6F1C1021	IGW8IIN2: Initialization invoked the RRS (resource recovery services) ATRIRNI function to retrieve a unit of recovery interest and received an error
6F1C1022	IGW8IIN2: ATRIERS failure
6F1C1023	IGW8IIN2: TMIB extract token failure
6F1C1024	IGW8IIN2: An error occurred while initialization was attempting to build a CURB
6F1C1023	IGW8IIN2: Initialization attempted to extract the token for the TMIB received an error

Initialization reason codes

Reason Code	Description
6F1C1025	IGW8IIN2: Initialization attempted to extract the token for the TMIB received an error
6F1C1026	IGW8IIN2: Initialization attempted to add a line to the SHCDS for RRS (resource recovery services) and received an error
6F1C1027	IGW8IIN2: Initialization attempted to retrieve system instance data from the SHCDS and received an error
6F1C1028	IGW8IIN2: Initialization attempted to find an item in the SHCDS and received an error
6F021029	IGW8IIN2: Initialization was unable to create a storage pool for MDTA blocks
6F1C1038	IGW8IIN2: Initialization received an error while trying to register result
6F1C1039	IGW8IIN2: Initialization invoked the SMLS query subsystem function and received an error
6F021040	IGW8IIN2: Initialization was unable to initialize the restart latch in the TMIB
6F1C1041	IGW8IIN2: Initialization detected an error while trying to obtain the restart latch in the TMIB
6F1C1042	IGW8IIN2: Initialization detected an error while trying to release the restart latch in the TMIB
6F1C1043	IGW8IIN2: Initialization detected an error while trying to start a browse of the undo log to perform restart processing
6F1C1044	IGW8IIN2: Initialization detected an error while trying to end a browse of the undo log during restart processing
6F1C1045	IGW8IIN2: Undo get next failure
6F1C1045	IGW8IIN2: Restore chain failure
6F1C1046	IGW8IIN2: Get TDSL block failure
6F1C1049	IGW8IIN2: TMIB set ATOKEN failure
6F1C1053	IGW8IIN2: QEXT SQM submit failure
6F1C1057	IGW8IIN2: DFSMStvs logger log of logs connect failure
6F1C105A	IGW8IIN2: Initialize connection to undo log submit failure
6F1C105B	IGW8IIN2: Initialize connection to log of logs submit failure
6F1C105C	IGW8IIN2: Initialize connection to undo log failure
6F1C105D	IGW8IIN2: Initialize connection to log of logs failure
6F1C1062	IGW8IIN2: Sharing control delete RRS (resource recovery services) list failure
6F1C1064	IGW8IIN2: SGIB extract token failure
6F1C1065	IGW8IIN2: SMIB extract token failure
6F1C1066	IGW8IIN2: DFSMStvs logger general log disconnect failure
6F1C1067	IGW8IIN2: Free TDSL block failure
6F1C106D	IGW8IIN2: FREEMAIN close list failure
6F1C106E	IGW8IIN2: GETMAIN close list failure
6F1C106F	IGW8IIN2: Front end task SQM submit failure
6F1C1071	IGW8IIN2: Obtain shunt latch failure
6F1C1072	IGW8IIN2: Release shunt latch failure
6F1C1074	IGW8IIN2: Excess sharing control lines failure
6F1C1075	IGW8IIN2: Obtain indoubt latch failure
6F1C1076	IGW8IIN2: Release indoubt latch failure
6F1C1080	IGW8IIN2: Browse chains start failure
6F1C1081	IGW8IIN2: Browse chains end failure
6F1C1082	IGW8IIN2: Get next chain failure
6F1C1083	IGW8IIN2: Chain browse get next failure
6F1C1084	IGW8IIN2: The 'about to release locks' record is present and should not be
6F1C1085	IGW8IIN2: The 'about to release locks' record is not first on chain

Initialization reason codes

Reason Code	Description
6F1C1087	IGW8IIN2: SMLS lost locks failure
6F1C1088	IGW8IIN2: Undo log quiesce failure
6F1C1089	IGW8IIN2: Shunt log quiesce failure
6F1C108A	IGW8IIN2: DFSMStvs quiesce failure
6F1C109C	IGW8IIN2: Obtain command latch failure
6F1C109D	IGW8IIN2: Release command latch failure
6F1C109E	IGW8IIN2: Log of Logs Quiesce Failure
6F1C10AC	IGW8IIN2: Log of Logs Quiesce Failure
6F411019	IGW8IIN6: Convert to FRR failure
6F41101A	IGW8IIN6: Convert to ESTAE failure
6F41101D	IGW8IIN6: Obtain TMIB latch failure
6F41101E	IGW8IIN6: Release TMIB latch failure
6F411023	IGW8IIN6: TMIB extract token failure
6F411039	IGW8IIN6: SMLS Query subsystem failure
6F41105F	IGW8IIN6: Read undo log failure
6F411060	IGW8IIN6: Start browse failure
6F411061	IGW8IIN6: End browse failure
6F4A1005	IGW8IIN7: Sharing control connect failure
6F4A1006	IGW8IIN7: Sharing control disconnect failure
6F4A1019	IGW8IIN7: Convert to FRR failure
6F4A101A	IGW8IIN7: Convert to ESTAE failure
6F4A1068	IGW8IIN7: Sharing control find item failure
6F4A1069	IGW8IIN7: Sharing control get system instance failure
6F4A106A	IGW8IIN7: GETMAIN sharing control failure
6F4A106B	IGW8IIN7: FREEMAIN sharing control failure
6F4A106C	IGW8IIN7: Sharing control delete list failure
6F521005	IGW8IRLB: Sharing control connect failure
6F521006	IGW8IRLB: Sharing control disconnect failure
6F521019	IGW8IRLB: Convert to FRR failure
6F52101A	IGW8IRLB: Convert to ESTAE failure
6F52101D	IGW8IRLB: Obtain TMIB latch failure
6F52101E	IGW8IRLB: Release TMIB latch failure
6F521024	IGW8IRLB: Build a CURB failure
6F521038	IGW8IRLB: Register result failure
6F521043	IGW8IRLB: Start browse undo log failure
6F521044	IGW8IRLB: Undo end browse failure
6F521045	IGW8IRLB: Undo get next failure
6F521045	IGW8IRLB: Restore chain failure
6F521046	IGW8IRLB: Get TDSL block failure
6F521067	IGW8IRLB: Free TDSL block failure
6F521068	IGW8IRLB: Sharing control find item failure
6F521069	IGW8IRLB: Sharing control get system instance failure
6F52106A	IGW8IRLB: GETMAIN sharing control failure
6F52106B	IGW8IRLB: FREEMAIN sharing control failure
6F52106C	IGW8IRLB: Sharing control delete list failure
6F52106D	IGW8IRLB: FREEMAIN close list failure
6F52106E	IGW8IRLB: GETMAIN close list failure
6F521080	IGW8IRLB: Browse chains start failure
6F521081	IGW8IRLB: Browse chains end failure
6F521082	IGW8IRLB: Get next chain failure
6F521083	IGW8IRLB: Chain browse get next failure
6F521085	IGW8IRLB: The "about to release locks" record is not first on chain
6F52108C	IGW8IRLB: Bad undo chain failure
6F52108D	IGW8IRLB: Free log record storage failure

Initialization reason codes

Reason Code	Description
6F52108E	IGW8IRLB: Get log record storage failure
6F52108F	IGW8IRLB: Undo log record check failure
6F521090	IGW8IRLB: Connect to redo log failure
6F521095	IGW8IRLB: Invalid owed file close record on the unit of recovery (UR) chain
6F521096	IGW8IRLB: Invalid start AKP record on UR chain
6F521097	IGW8IRLB: Invalid end AKP record on UR chain
6F521099	IGW8IRLB: No shunt status in DSN failure
6F52109A	IGW8IRLB: Bad data name failure
6F52109F	IGW8IRLB: Disconnect redo log failure
6F5210A0	IGW8IRLB: Failure on call to SCM to disconnect redo log
6F5210AE	IGW8IRLB: Curb latch failure
6F5210B3	IGW8IRLB: Disconnect redo log failure
6F59100E	IGW8IRS1: ATRISLN failure
6F591020	IGW8IRS1: ATRIBRS failure
6F591021	IGW8IRS1: ATRIRNI failure
6F591024	IGW8IRS1: Curb build failure
6F59101A	IGW8IRS1: Convert to ESTAE failure
6F591019	IGW8IRS1: Convert to Frr failure
6F5910BB	IGW8IRS1: Curb latch failure
6F591047	IGW8IRS1: Get TDSL block failure
6F59105C	IGW8IRS1: Initialization connection undo log failure
6F5910D3	IGW8IRS1: Obtain indoubt chain latch failure
6F5910CC	IGW8IRS1: Obtain restart chain latch failure
6F59101D	IGW8IRS1: Obtain TMIB latch failure
6F591038	IGW8IRS1: Register result failure
6F5910D2	IGW8IRS1: ReleaseIndoubtChnLatch failure
6F5910CB	IGW8IRS1: ReleaseRestartChnLatch failure
6F59101E	IGW8IRS1: ReleaseTmibLatch failure
6F591026	IGW8IRS1: shcds_AddRRSLine failure
6F591005	IGW8IRS1: shcds_Connect failure
6F591006	IGW8IRS1: shcds_DisConnect failure
6F591039	IGW8IRS1: SMLS_QuerySubsystem failure
6F591015	IGW8IRS1: TLSLogInit failure
6F591057	IGW8IRS1: TLSLogofLogsCon failure
6F591010	IGW8IRS1: VRGB_ExtractToken failure
6F5A101A	IGW8IRS2: ConvertToEstae failure
6F5A1092	IGW8IRS2: FreeCurbBlock failure
6F5A1067	IGW8IRS2: FreeTdslBlock failure
6F5A10CC	IGW8IRS2: ObtainRestartChnLatch failure
6F5A10CB	IGW8IRS2: ReleaseRestartChnLatch failure
6F5A10ED	IGW8IRS2: SMLSEndTransaction failure
6F5A1087	IGW8IRS2: SMLSLostLocks failure
6F582000	IGW8ISTX: ATRIRLN failure
6F582001	IGW8ISTX: ATRISLN failure
6F582002	IGW8ISTX: ConvertToEstae failure
6F582003	IGW8ISTX: ConvertToFrr failure
6F58200E	IGW8ISTX: CRGGRM failure
6F582004	IGW8ISTX: CRGSEIF failure
6F582005	IGW8ISTX: CRGSEIFRRS failure
6F582006	IGW8ISTX: DeleteSHCLine failure
6F58200D	IGW8ISTX: ExcessSHCLines failure
6F58203F	IGW8ISTX: Indeterminate failure
6F58200F	IGW8ISTX: ObtainTVSLatch failure

Initialization reason codes

Reason Code	Description
6F582007	IGW8ISTX: ShcDeleteRRSList failure
6F582008	IGW8ISTX: ShcdsAddRRSLine failure
6F582009	IGW8ISTX: ShcdsConnect failure
6F58200A	IGW8ISTX: ShcdsDisConnect failure
6F58200B	IGW8ISTX: ShcFindItem failure
6F58200C	IGW8ISTX: ShcGetSysInstance failure
6F582010	IGW8ISTX: ReleaseTVSLatch failure

Open and close reason codes

Open and close reason codes

Reason Code	Description
6F041500	IGW8OODS: Open failed because DFSMStvs is unavailable.
6F041503	IGW8OODS: The data set is quiesced.
6F041504	IGW8OODS: Write to undo log failure
6F041505	IGW8OODS: Write to forward recovery log failure
6F041506	IGW8OODS: Write to log of logs failure
6F041508	IGW8OODS: Log stream ID is required.
6F041509	IGW8OODS: Log stream ID specifies a DFSMStvs undo or shunt log.
6F04150A	IGW8OODS: Open failed because the LRECL for data set is greater than 62 KB.
6F04150B	IGW8OODS: Open failed because the maximum LRECL for data set is not supported by forward recovery log.
6F04150C	IGW8OODS: Open failed because the maximum LRECL for the data set is not supported by undo log.
6F04150D	IGW8OODS: Open failed because a prior open is still active and the log stream ID or log parameter has been changed for this open.
6F0415E6	IGW8OODS: Initialize FR latch failure
6F0415E7	IGW8OODS: Release FR latch failure
6F0415E8	IGW8OODS: Get FR latch failure
6F0415EA	IGW8OODS: SHCDS disconnect failure
6F0415EB	IGW8OODS: TOBE chain failure
6F0415EC	IGW8OODS: TOBH chain failure
6F0415ED	IGW8OODS: SHCDS connect failure
6F0415EE	IGW8OODS: Open failed to obtain the VMIB token
6F0415EF	IGW8OODS: Get IGWLOGS pool failure
6F0415F0	IGW8OODS: SQM submit failure
6F0415F1	IGW8OODS: Open detected an error while converting from ESTAE to FRR
6F0415F2	IGW8OODS: Open detected an error while converting from FRR to ESTAE
6F0415F4	IGW8OODS: Open attempted to extract the token for the TMIB received an error
6F0415F5	IGW8OODS: Open attempted to obtain a storage pool for DFSMStvs open block extension header (TOBH) and received an error
6F0415F6	IGW8OODS: Open received an error while trying to generate a message
6F0415F7	IGW8OODS: Open attempted to release the TOBH latch and received an error
6F0415F8	IGW8OODS: Open was unable to initialize the latch in the TOBH
6F0415F9	IGW8OODS: Open detected an error while trying to release the DFSMStvs latch in the TMIB
6F0415FA	IGW8OODS: Open attempted to obtain the TOBH latch and received an error.

Open and close reason codes

Reason Code	Description
6F0415FB	IGW8OODS: Open attempted to obtain the DFSMStvs latch in the TMIB and received an error
6F0415FC	IGW8OODS: Get TOBE pool failure
6F0415FD	IGW8OODS: Open attempted to a DFSMStvs open block extension header (TOBH) from the pool and received an error
6F051601	IGW8OCDS: Disconnect redo log failure
6F0516E9	Close found that the DFSMStvs state or a DFSMStvs log state had previously transitioned to quiesced or disabled while still in use.
6F0516EC	IGW8OCDS: Release FR latch failure
6F0516ED	IGW8OCDS: Get FR latch failure
6F0516EE	IGW8OCDS: Vary SQM submit failure
6F0516EF	IGW8OCDS: Close failed to GETMAIN a parameter list to change the DFSMStvs and system logs state to quiesced or disabled.
6F0516F1	IGW8OCDS: The data set being closed has an I/O error on its forward recovery log. Close detected an error while attempting to obtain a DFSMStvs quiesce request (QEXT) from the free pool.
6F0516F4	IGW8OCDS: The data set being closed has an I/O error on its forward recovery log. Close detected an error while trying to quiesce the data set.
6F0516F5	IGW8OCDS: Close detected an error while converting from ESTAE to FRR
6F0516F6	IGW8OCDS: Close detected an error while converting from FRR to ESTAE
6F0516F7	IGW8OCDS: Generate message failure. Close received an error while trying to generate a message (module IGW8OCDS).
6F0516F8	IGW8OCDS: Close could not find the DFSMStvs open block extension (TOBE) for the data set
6F0516FA	IGW8OCDS: Close attempted to release the TOBH latch and received an error
6F0516FB	IGW8OCDS: Close attempted to obtain the TOBH latch and received an error
6F0516FC	IGW8OCDS: Close could not find the DFSMStvs open block extension header (TOBH) for the data set
6F0516FD	IGW8OCDS: Close failed while attempting to release the DFSMStvs latch in the TMIB
6F0516FD	IGW8OCDS: Release TMIB latch failure
6F0516FE	IGW8OCDS: Close failed while attempting to obtain the DFSMStvs latch in the TMIB
6F0516FF	IGW8OCDS: Close attempted to extract the token for the TMIB received an error
6F5D1500	IGW8OPNR: DFSMStvs is unavailable
6F5D1504	IGW8OPNR: Connect to undo log failed
6F5D15DE	IGW8OPNR: Release peer chain latch failure
6F5D15DF	IGW8OPNR: Release indoubt chain latch failure
6F5D15E0	IGW8OPNR: Release shunt chain latch failure
6F5D15E1	IGW8OPNR: Release curb latch failure
6F5D15E2	IGW8OPNR: Get peer chain latch failure
6F5D15E3	IGW8OPNR: Get indoubt chain latch failure
6F5D15E4	IGW8OPNR: Get shunt chain latch failure
6F5D15E5	IGW8OPNR: Get curb latch failure
6F5D15E5	IGW8OPNR: Issue IDARECOV failure
6F5D15F1	IGW8OPNR: Convert to Frr failure
6F5D15F2	IGW8OPNR: Convert to Estae failure
6F5D15F4	IGW8OPNR: Get TMIB failure
6F5D15F6	IGW8OPNR: Generate message failure

Open and close reason codes

Reason Code	Description
6F5D15F9	IGW8OPNR: Release TMIB latch failure
6F5D15FB	IGW8OPNR: Get TMIB latch failure
6F2B1600	IGW8OCHK: Active transaction failure
6F2B16F5	IGW8OCHK: Convert to FRR failure
6F2B16F6	IGW8OCHK: Convert to ESTAE failure
6F2B16F7	IGW8OCHK: Generate message failure
6F2B16FD	IGW8OCHK: Release TMIB latch failure
6F2B16FE	IGW8OCHK: Get TMIB latch failure
6F2B16FF	IGW8OCHK: Get TMIB failure
6F3F1500	IGW8OLOG: Open failed because DFSMStvs is unavailable
6F3F1503	IGW8OLOG: Open failed because the data set is quiesced
6F3F1504	IGW8OLOG: Connect redo log failure
6F3F1504	IGW8OLOG: The forward recovery log is unavailable
6F3F1505	IGW8OLOG: Write to the forward recovery log failed
6F3F1506	IGW8OLOG: Write log of logs failure
6F3F150E	IGW8OLOG: Create undo log owed file close chain failure
6F3F150F	IGW8OLOG: Write owed file close chain to undo log failed
6F3F1510	IGW8OLOG: There was a previous I/O error on the forward recovery log
6F3F15E9	IGW8OLOG: DFSMStvs open passed a parameter list which was not valid
6F3F15ED	IGW8OLOG: SHCDS connect failure
6F3F15EE	IGW8OLOG: Get VMIB failure
6F3F15F1	IGW8OLOG: Open failed while converting from ESTAE to FRR
6F3F15F2	IGW8OLOG: Open failed while converting from FRR to ESTAE
6F3F15F4	IGW8OLOG: Open attempted to extract the token for the TMIB and received an error
6F3F15F6	IGW8OLOG: Open detected an error while attempting to issue a message
6F3F15F7	IGW8OLOG: Open failed while attempting to release the TOBH latch
6F3F15F9	IGW8OLOG: Open failed while attempting to release the TMIB latch
6F3F15FA	IGW8OLOG: Open failed while attempting to obtain the TOBH latch
6F3F15FB	IGW8OLOG: Open failed while attempting to obtain the TMIB latch
6F3F1601	IGW8OLOG: Disconnect redo log failure
6F3F1602	IGW8OLOG: Close detected an error attempting to write a close record to the forward recovery log
6F3F16EA	IGW8OLOG: Close was unable to release the DFSMStvs open/close latch
6F3F16EB	IGW8OLOG: Close was unable to obtain the DFSMStvs open/close latch
6F3F16EC	IGW8OLOG: Release FR latch failure
6F3F16ED	IGW8OLOG: Get FR latch failure
6F3F16F0	IGW8OLOG: SHCDS disconnect failure
6F3F16F2	IGW8OLOG: Close detected an error while attempting to change a log status to quiesced or disabled for the log of logs
6F3F16F3	IGW8OLOG: Quiesce forward recovery log failure

Command processor reason codes

Command processor reason codes

Reason Code	Description
6F311701	IGW8CDLG: Generate message failure
6F311710	IGW8CDLG: Extract TMIB failure

Command processor reason codes

Reason Code	Description
6F311719	IGW8CDLG: Convert to FRR failure
6F31171A	IGW8CDLG: Convert to ESTAE failure
6F31171B	IGW8CDLG: GETMAIN arguments failure
6F31171C	IGW8CDLG: FREEMAIN arguments failure
6F31171D	IGW8CDLG: Obtain TMIB latch failure
6F31171E	IGW8CDLG: Release TMIB latch failure
6F311733	IGW8CDLG: Obtain command latch failure
6F311734	IGW8CDLG: Release command latch failure
6F321701	IGW8CDTV: Generate message failure
6F321710	IGW8CDTV: Extract TMIB failure
6F321719	IGW8CDTV: Convert to FRR failure
6F32171A	IGW8CDTV: Convert to ESTAE failure
6F32171B	IGW8CDTV: GETMAIN arguments failure
6F32171C	IGW8CDTV: FREEMAIN arguments failure
6F32171D	IGW8CDTV: Obtain TMIB latch failure
6F32171E	IGW8CDTV: Release TMIB latch failure
6F321733	IGW8CDTV: Obtain command latch failure
6F321734	IGW8CDTV: Release command latch failure
6F331701	IGW8CDJB: Generate message failure
6F33170D	IGW8CDJB: ATRRID failure
6F33170E	IGW8CDJB: ATRRURD failure
6F331710	IGW8CDJB: Extract TMIB failure
6F331719	IGW8CDJB: Convert to FRR failure
6F33171A	IGW8CDJB: Convert to ESTAE failure
6F33171B	IGW8CDJB: GETMAIN arguments failure
6F33171C	IGW8CDJB: FREEMAIN arguments failure
6F33171D	IGW8CDJB: Obtain TMIB latch failure
6F33171E	IGW8CDJB: Release TMIB latch failure
6F331733	IGW8CDJB: Obtain command latch failure
6F331734	IGW8CDJB: Release command latch failure
6F351701	IGW8CDDS: Generate message failure
6F351710	IGW8CDDS: Extract TMIB failure
6F351718	IGW8CDDS: Find failure
6F351719	IGW8CDDS: Convert to FRR failure
6F35171A	IGW8CDDS: Convert to ESTAE failure
6F35171B	IGW8CDDS: GETMAIN arguments failure
6F35171C	IGW8CDDS: FREEMAIN arguments failure
6F35171D	IGW8CDDS: Obtain TMIB latch failure
6F35171E	IGW8CDDS: Release TMIB latch failure
6F351733	IGW8CDDS: Obtain command latch failure
6F351734	IGW8CDDS: Release command latch failure
6F361701	IGW8CVLG: Generate message failure
6F361710	IGW8CVLG: Extract TMIB failure
6F361715	IGW8CVLG: DFSMStvs logger initialization failure
6F361719	IGW8CVLG: Convert to FRR failure
6F36171A	IGW8CVLG: Convert to ESTAE failure
6F36171B	IGW8CVLG: GETMAIN arguments failure
6F36171C	IGW8CVLG: FREEMAIN arguments failure
6F36171D	IGW8CVLG: Obtain TMIB latch failure
6F36171E	IGW8CVLG: Release TMIB latch failure
6F36171F	IGW8CVLG: CRGDRM failure
6F361720	IGW8CVLG: SMLS unidentify subsystem failure
6F361721	IGW8CVLG: Init SQM submit failure
6F361722	IGW8CVLG: DFSMStvs logger initialization failure

Command processor reason codes

Reason Code	Description
6F361723	IGW8CVLG: Log disconnect failure
6F361726	IGW8CVLG: Obtain shunt latch failure
6F361727	IGW8CVLG: Release shunt latch failure
6F36172A	IGW8CVLG: Vary SQM submit failure
6F361731	IGW8CVLG: Obtain open/close latch failure
6F361732	IGW8CVLG: Release open/close latch failure
6F361733	IGW8CVLG: Obtain command latch failure
6F361734	IGW8CVLG: Release command latch failure
6F361735	IGW8CVLG: Log of logs disconnect failure
6F371701	IGW8CVTV: Generate message failure
6F371710	IGW8CVTV: Extract TMIB failure
6F371719	IGW8CVTV: Convert to FRR failure
6F37171A	IGW8CVTV: Convert to ESTAE failure
6F37171B	IGW8CVTV: GETMAIN arguments failure
6F37171C	IGW8CVTV: FREEMAIN arguments failure
6F37171D	IGW8CVTV: Obtain TMIB latch failure
6F37171E	IGW8CVTV: Release TMIB latch failure
6F37171F	IGW8CVTV: CRGDRM failure
6F371720	IGW8CVTV: SMLS unidentify subsystem failure
6F371721	IGW8CVTV: Init SQM submit failure
6F371723	IGW8CVTV: Log disconnect failure
6F37172A	IGW8CVTV: Vary SQM submit failure
6F371733	IGW8CVTV: Obtain command latch failure
6F371734	IGW8CVTV: Release command latch failure
6F381719	IGW8CSAK: Convert to FRR failure
6F38171A	IGW8CSAK: Convert to ESTAE failure
6F381733	IGW8CSAK: Obtain command latch failure
6F381734	IGW8CSAK: Release command latch failure
6F391710	IGW8CSAK: Extract TMIB failure
6F391719	IGW8CSAK: Convert to FRR failure
6F39171A	IGW8CSAK: Convert to ESTAE failure
6F391733	IGW8CSAK: Obtain command latch failure
6F391734	IGW8CSAK: Release command latch failure
6F3A1710	IGW8CVDS: Extract TMIB failure
6F3A1719	IGW8CVDS: Convert to FRR failure
6F3A171A	IGW8CVDS: Convert to ESTAE failure
6F3A1733	IGW8CVDS: Obtain command latch failure
6F3A1734	IGW8CVDS: Release command latch failure
6F3B1701	IGW8CDUR: Generate message failure
6F3B170D	IGW8CDUR: ATRRID failure
6F3B170E	IGW8CDUR: ATRRURD failure
6F3B1710	IGW8CDUR: Extract TMIB failure
6F3B1719	IGW8CDUR: Convert to FRR failure
6F3B171A	IGW8CDUR: Convert to ESTAE failure
6F3B171B	IGW8CDUR: GETMAIN arguments failure
6F3B171C	IGW8CDUR: FREEMAIN arguments failure
6F3B171D	IGW8CDUR: Obtain TMIB latch failure
6F3B171E	IGW8CDUR: Release TMIB latch failure
6F3B1733	IGW8CDUR: Obtain command latch failure
6F3B1734	IGW8CDUR: Release command latch failure
6F4B1701	IGW8CVT1: Generate message failure
6F4B1710	IGW8CVT1: Extract TMIB failure
6F4B1719	IGW8CVT1: Convert to FRR failure
6F4B171A	IGW8CVT1: Convert to ESTAE failure

Command processor reason codes

Reason Code	Description
6F4B171B	IGW8CVT1: GETMAIN arguments failure
6F4B171C	IGW8CVT1: FREEMAIN arguments failure
6F4B171D	IGW8CVT1: Obtain TMIB latch failure
6F4B171E	IGW8CVT1: Release TMIB latch failure
6F4B171F	IGW8CVT1: CRGDRM failure
6F4B1720	IGW8CVT1: SMLS unidentify subsystem failure
6F4B1723	IGW8CVT1: Log disconnect failure
6F4B1726	IGW8CVT1: Obtain shunt latch failure
6F4B1727	IGW8CVT1: Release shunt latch failure
6F4B1728	IGW8CVT1: Obtain indoubt latch failure
6F4B1729	IGW8CVT1: Release indoubt latch failure
6F4B172B	IGW8CVT1: Undo log quiesce failure
6F4B172C	IGW8CVT1: Shunt log quiesce failure
6F4B172D	IGW8CVT1: Log of logs quiesce failure
6F4B172E	IGW8CVT1: Redo log quiesce failure
6F4B1730	IGW8CVT1: Undo log disconnect failure
6F4B1733	IGW8CVT1: Obtain command latch failure
6F4B1734	IGW8CVT1: Release command latch failure
6F4B1735	IGW8CVT1: Log of logs disconnect failure

Front end (VSAM record management) reason codes

Front end (VSAM record management) reason codes

Reason Code	Description
6F061100	IGW8FCRB: An attempt to extract the token for the CURB was unsuccessful.
6F061101	IGW8FCRB: An attempt to set the token for the CURB was unsuccessful
6F061104	IGW8FCRB: The TMIB token is zero
6F06112C	IGW8FCRB: DFSMStvs is quiesced and there is an existing unit of recovery
6F071106	IGW8FEXP: An invocation of CTXEINT to express interest in a context failed
6F071107	IGW8FEXP: An invocation of ATREINT to express interest in a unit of recovery failed
6F071108	IGW8FEXP: An invocation of ATRSIT to change interest type from unprotected to protected failed
6F071121	IGW8FEXP: DFSMStvs invoked ATREINT to express interest in a UR, but RRS (resource recovery services) is unavailable
6F071122	IGW8FEXP: DFSMStvs invoked ATRSIT to change interest type from unprotected to protected, but RRS (resource recovery services) is unavailable
6F071123	IGW8FEXP: DFSMStvs invoked ATREINT to express interest in a UR, but RRS (resource recovery services) restarted while the UR was in-flight; the UR must issue a sync point before issuing further requests
6F07112D	IGW8FEXP: The undo log status is not enabled
6F07112F	IGW8FEXP: STOKEN extract failure
6F071134	IGW8FEXP: DFSMStvs tried to express interest in a UR and found that DFSMStvs had restarted
6F07113F	IGW8FEXP: DFSMStvs restarted
6F071140	IGW8FEXP: An invocation of CTXSCID to set context interest data failed
6F071143	IGW8FEXP: TTOKEN extract failure

Front end (VSAM record management) reason codes

Reason Code	Description
6F08110C	IGW8FCLN: An attempt to obtain the DFSMStvs latch in the TMIB failure
6F08110D	IGW8FCLN: An attempt to release the DFSMStvs latch in the TMIB failure
6F08111A	IGW8FCLN: Front end processing received an error while attempting to generate a message
6F08114E	IGW8FCLN: Later abend failure
6F211100	IGW8FLBI: An attempt to extract the token for the CURB was unsuccessful
6F211103	IGW8FLBI: An attempt to extract the token for the TMIB was unsuccessful
6F211105	IGW8FLBI: Get block failure
6F211109	IGW8FLBI: Register result failure
6F21110C	IGW8FLBI: An attempt to obtain the DFSMStvs latch in the TMIB failed
6F21110D	IGW8FLBI: An attempt to release the DFSMStvs latch in the TMIB failed
6F21110F	IGW8FLBI: Request rejected because DFSMStvs is in an invalid state
6F211112	IGW8FLBI: DFSMStvs detected that the RPL passed by the caller is not on the RPL chain from a previous GET
6F211115	IGW8FLBI: An attempt to force records to the undo log failed
6F21111A	IGW8FLBI: Front end processing received an error while attempting to generate a message
6F21111B	IGW8FLBI: An attempt to obtain the DFSMStvs latch in the TMIB for SMLS register failed
6F21111C	IGW8FLBI: An attempt to obtain the DFSMStvs latch in the TMIB to chain a TDSL failed
6F21111D	IGW8FLBI: An attempt to obtain the DFSMStvs latch in the TMIB to chain a CURB failed
6F21111E	IGW8FLBI: An attempt to obtain the DFSMStvs latch in the TMIB to chain an RPL failed
6F21111F	IGW8FLBI: An attempt to obtain the DFSMStvs latch in the TMIB to chain an RPL failed
6F211120	IGW8FLBI: DFSMStvs instance count mismatch
6F21112B	IGW8FLBI: The TMIB is not initialized and RRS (resource recovery services) is unavailable
6F21112C	IGW8FLBI: DFSMStvs is quiesced and there is an existing unit of recovery
6F21112D	IGW8FLBI: The undo log status is not enabled
6F211131	IGW8FLBI: The data set is quiesced
6F211132	IGW8FLBI: The data set is quiesced
6F211133	IGW8FLBI: The data set is quiesced
6F21115E	IGW8FLBI: The data set is quiesced
6F21115F	IGW8FLBI: The data set is quiesced
6F211142	IGW8FLBI: The URID in the CURB is zero
6F211148	IGW8FLBI: DFSMStvs is disabling
6F211149	IGW8FLBI: Unexpected error
6F221100	IGW8FLAI: An attempt to extract the token for the CURB was unsuccessful
6F221103	IGW8FLAI: An attempt to extract the token for the TMIB was unsuccessful
6F22110A	IGW8FLAI: An attempt to create a chain for undo log records failed
6F22110B	IGW8FLAI: An attempt to write a record to the undo log failed
6F22110C	IGW8FLAI: An attempt to obtain the DFSMStvs latch in the TMIB failure

Front end (VSAM record management) reason codes

Reason Code	Description
6F22110D	IGW8FLAI: An attempt to release the DFSMStvs latch in the TMIB failure
6F22110F	IGW8FLAI: Request rejected because DFSMStvs is in an invalid state
6F221112	IGW8FLAI: DFSMStvs detected that the RPL passed by the caller is not on the RPL chain
6F221116	IGW8FLAI: Front end processing received an error while attempting to generate message IGW10071I
6F221117	IGW8FLAI: Front end processing received an error while attempting to generate message IGW10071I
6F221118	IGW8FLAI: Front end processing received an error while attempting to generate message IGW10071I
6F22111A	IGW8FLAI: Front end processing received an error while attempting to generate a message
6F221120	IGW8FLAI: DFSMStvs instance count mismatch
6F221124	IGW8FLAI: A buffer length error was detected on a write force request
6F221125	IGW8FLAI: Front end processing received an error while attempting to generate VPDI common message
6F221126	IGW8FLAI: An attempt to write a record to the redo log failed
6F221127	IGW8FLAI: General log buffer length error
6F221128	IGW8FLAI: A permanent log error occurred trying to create an undo log chain
6F221129	IGW8FLAI: A system logger error occurred writing to a general log
6F22112B	IGW8FLAI: The TMIB is not initialized and RRS (resource recovery services) is unavailable
6F22112C	IGW8FLAI: DFSMStvs is quiesced and there is an existing unit of recovery
6F22112D	IGW8FLAI: The undo log status is not enabled
6F22112E	IGW8FLAI: The undo log status is not enabled
6F221130	IGW8FLAI: The redo log status is not enabled
6F221135	IGW8FLAI: Front end processing received an error while attempting to generate message IGW10075I
6F221136	IGW8FLAI: Front end processing received an error while attempting to generate message IGW10075I
6F221137	IGW8FLAI: Front end processing received an error while attempting to generate message IGW10075I
6F221138	IGW8FLAI: Front end processing received an error while attempting to generate message IGW10073I
6F221139	IGW8FLAI: Front end processing received an error while attempting to generate message IGW10073I
6F22113A	IGW8FLAI: Front end processing received an error while attempting to generate message IGW10073I
6F22113B	IGW8FLAI: Front end processing received an error while attempting to generate message IGW10073I
6F221148	IGW8FLAI: DFSMStvs is disabling
6F221149	IGW8FLAI: Unexpected error
6F261100	IGW8FLPA: An attempt to extract the token for the CURB was unsuccessful
6F261103	IGW8FLPA: An attempt to extract the token for the TMIB was unsuccessful
6F26110A	IGW8FLPA: An attempt to create a chain for undo log records failed
6F26110B	IGW8FLPA: An attempt to write a record to the undo log failed
6F26110C	IGW8FLPA: An attempt to obtain the DFSMStvs latch in the TMIB failure

Front end (VSAM record management) reason codes

Reason Code	Description
6F26110D	IGW8FLPA: An attempt to release the DFSMStvs latch in the TMIB failure
6F26110F	IGW8FLPA: Request rejected because DFSMStvs is in an invalid state
6F26111A	IGW8FLPA: Front end processing received an error while attempting to generate a message
6F261120	IGW8FLPA: DFSMStvs instance count mismatch
6F261124	IGW8FLPA: A buffer length error was detected on a write force request
6F261128	IGW8FLPA: A permanent log error occurred trying to create an undo log chain
6F26112B	IGW8FLPA: The TMIB is not initialized and RRS (resource recovery services) is unavailable
6F26112C	IGW8FLPA: DFSMStvs is quiesced and there is an existing unit of recovery
6F26112D	IGW8FLPA: The undo log status is not enabled
6F26112E	IGW8FLPA: The undo log status is not enabled
6F261148	IGW8FLPA: DFSMStvs is disabling
6F261149	IGW8FLPA: Unexpected error
6F281119	IGW8FMSG: An attempt was made to generate a message for which a message definition could not be found
6F28113C	IGW8FMSG: An error occurred attempting to generate a variant of message IGW10071I
6F28113D	IGW8FMSG: An error occurred attempting to generate a variant of message IGW10073I
6F28113E	IGW8FMSG: An error occurred attempting to generate a variant of message IGW10075I
6F281149	IGW8FMSG: Unexpected error
6F4C1103	IGW8FTSK: An attempt to extract the token for the TMIB was unsuccessful
6F4C110C	IGW8FTSK: An attempt to obtain the DFSMStvs latch in the TMIB failure
6F4C110D	IGW8FTSK: An attempt to release the DFSMStvs latch in the TMIB failure
6F4C1113	IGW8FTSK: Convert to ESTAE failure
6F4C1114	IGW8FTSK: Convert to FRR failure
6F4C111A	IGW8FTSK: Front end processing received an error while attempting to generate a message
6F4C1141	IGW8FTSK: Front end processing received an error while attempting to generate message IGW10074I
6F4C1149	IGW8FTSK: Unexpected error

Message processing reason codes

Message processing reason codes

Reason Code	Description
6F291030	IGW8XMSG: Free MDTA failure
6F291031	IGW8XMSG: Convert to ESTAE failure
6F291032	IGW8XMSG: Convert to FRR failure
6F291035	IGW8XMSG: Invalid function requested
6F291036	IGW8XMSG: Invalid function requested
6F291078	IGW8XMSG: Console message GETMAIN failure
6F291079	IGW8XMSG: Job log message GETMAIN failure
6F29107A	IGW8XMSG: Console message segment GETMAIN failure
6F29107B	IGW8XMSG: Job log message segment GETMAIN failure

Message processing reason codes

Reason Code	Description
6F29107C	IGW8XMSG: Console message FREEMAIN failure
6F29107D	IGW8XMSG: Job log message FREEMAIN failure
6F29107E	IGW8XMSG: Console message segment FREEMAIN failure
6F29107F	IGW8XMSG: Job log message segment FREEMAIN failure
6F291091	IGW8XMSG: Hard copy message FREEMAIN failure
6F291092	IGW8XMSG: Hard copy message GETMAIN failure
6F291093	IGW8XMSG: Hard copy message segment FREEMAIN failure
6F291094	IGW8XMSG: Hard copy message segment GETMAIN failure

Quiesce reason codes

Quiesce reason codes

Reason Code	Description
6F2D16F6	Generate message failure (module IGW8QPOP)
6F2D16F7	Convert to Frr failure (module IGW8QPOP)
6F2D16F8	Convert to ESTAE failure (module IGW8QPOP)
6F2D16FF	Get TMIB failure (module IGW8QPOP)
6F2E16E6	Quiesce received an incorrect IFGQUIES parameter block from RLS (module IGW8QEXP)
6F2E16F4	Quiesce received a data set quiesce request from RLS and the data set name was not specified (module IGW8QEXP)
6F2E16F5	Quiesce was unable to obtain an QEXT from the QEXT storage pool (module IGW8QEXP)
6F2E16F6	Quiesce received an error while attempting to issue a message (module IGW8QEXP)
6F2E16F7	Quiesce detected an error while converting from ESTAE to FRR (module IGW8QEXP)
6F2E16F8	Quiesce detected an error while converting from FRR to ESTAE (module IGW8QEXP)
6F2E16F9	Quiesce attempted to release the TOBH latch and received an error (module IGW8QEXP)
6F2E16FA	Quiesce attempted to obtain the TOBH latch and received an error (module IGW8QEXP)
6F2E16FB	Quiesce attempted to release the TMIB latch and received an error (module IGW8QEXP)
6F2E16FC	Quiesce attempted to obtain the TMIB latch and received an error (module IGW8QEXP)
6F2E16FF	Quiesce attempted to extract the token for the TMIB and received an error (module IGW8QEXP)
6F2F16E8	Quiesce was unable to return a QEXT to the QEXT storage pool (module IGW8QEXT)
6F2F16EE	Quiesce was unable to write a tie up record to the forward recovery log (module IGW8QEXT)
6F2F16EF	Quiesce attempted to issue an SQM submit of a task and detected an error (module IGW8QEXT)
6F2F16F5	Quiesce was unable to obtain a QEXT from the QEXT storage pool (module IGW8QEXT)
6F2F16F6	Quiesce detected an error while attempting to issue a message (module IGW8QEXT)
6F2F16F7	Quiesce detected an error while converting from ESTAE to FRR (module IGW8QEXT)
6F2F16F8	Quiesce detected an error while converting from FRR to ESTAE (module IGW8QEXT)

Quiesce reason codes

Reason Code	Description
6F2F16F9	Quiesce attempted to release the TOBH latch and received an error (module IGW8QEXT)
6F2F16FA	Quiesce attempted to obtain the TOBH latch and received an error (module IGW8QEXT)
6F2F16FB	Quiesce attempted to release the TMIB latch and received an error (module IGW8QEXT)
6F2F16FC	Quiesce attempted to obtain the TMIB latch and received an error (module IGW8QEXT)
6F2F16FF	Quiesce attempted to extract the token for the TMIB and received an error (module IGW8QEXT)
6F3016DC	Release TOBHX chain latch failure (module IGW8QEXR)
6F3016DD	Get TOBHX chain latch failure (module IGW8QEXR)
6F3016DE	Release QEXT chain latch failure (module IGW8QEXR)
6F3016DF	Get QEXT chain latch failure (module IGW8QEXR)
6F3016E7	Quiesce attempted to obtain an IFGQUIES from the IFGQUIES storage pool and received an error (module IGW8QEXR)
6F3016E8	Quiesce attempted to return a QEXT to the QEXT storage pool and received an error (module IGW8QEXR)
6F3016E9	Quiesce attempted to return an IFGQUIES to the IFGQUIES storage pool and received an error (module IGW8QEXR)
6F3016EA	Quiesce attempted to issue an IDAQUIES and received an error (module IGW8QEXR)
6F3016F6	Quiesce received an error while attempting to issue a message (module IGW8QEXR)
6F3016F7	Quiesce detected an error while converting from ESTAE to FRR (module IGW8QEXR)
6F3016F8	Quiesce detected an error while converting from FRR to ESTAE (module IGW8QEXR)
6F3016FF	Quiesce attempted to extract the token for the TMIB and received an error (module IGW8QEXR)
6F5116E2	Call retry shunt failure (module IGW8QSHN)
6F5116E3	Release shunt chain latch failure (module IGW8QSHN)
6F5116E4	Get shunt chain latch failure (module IGW8QSHN)
6F5116E5	Get a IGWQWRK from the free pool failure (module IGW8QSHN)
6F5116F6	Issue a message failure (module IGW8QSHN)
6F5116F7	Convert to FRR failure (module IGW8QSHN)
6F5116F8	Convert to ESTAE failure (module IGW8QSHN)
6F5116FB	Release the TMIB latch failure (module IGW8QSHN)
6F5116FC	Get the TMIB latch failure (module IGW8QSHN)
6F5116FF	Obtain a TMIB failure (module IGW8QSHN)

Shunt processing reason codes

Shunt processing reason codes

Reason Code	Description
5F491701	IGW8SLUC: Generate message failure
5F491710	IGW8SLUC: Extract TMIB failure
5F491719	IGW8SLUC: Convert to FRR failure
5F49171A	IGW8SLUC: Convert to ESTAE failure
5F49171D	IGW8SLUC: Obtain TMIB latch failure
5F49171E	IGW8SLUC: Release TMIB latch failure
5F491725	IGW8SLUC: Invalid BTREE token failure
5F491726	IGW8SLUC: Obtain shunt latch failure
5F491727	IGW8SLUC: Release shunt latch failure

Shunt processing reason codes

Reason Code	Description
5F491733	IGW8SLUC: Obtain command latch failure
5F491734	IGW8SLUC: Release command latch failure
6F2B1205	IGW8SLUD: Obtain curb latch failure
6F2B1206	IGW8SLUD: Release curb latch failure
6F2B1289	IGW8SLUD: Obtain curb chain latch failure
6F2B128A	IGW8SLUD: Release curb chain latch failure
6F2B1823	IGW8SLUD: Invalid B-tree token failure
6F2B182C	IGW8SLUD: Obtain in-doubt latch failure
6F2B182D	IGW8SLUD: Release in-doubt latch failure
6F2B182F	IGW8SLUD: Tree add failure
6F2B1833	IGW8SLUD: Obtain shunt latch failure
6F2B1834	IGW8SLUD: Release shunt latch failure
6F2B1836	IGW8SLUD: End browse failure
6F2B183B	IGW8SLUD: Obtain restart latch failure
6F2B183C	IGW8SLUD: Release restart latch failure
6F2B183F	IGW8SLUD: Indeterminate failure
6F431800	IGW8SRBC: Convert to ESTAE failure
6F431801	IGW8SRBC: Convert to FRR failure
6F431802	IGW8SRBC: Find failure
6F431803	IGW8SRBC: Invalid BTREE token failure
6F431804	IGW8SRBC: Overflow of local TDSL failure
6F431805	IGW8SRBC: Read undo log failure
6F431806	IGW8SRBC: SMLS end transaction failure
6F431807	IGW8SRBC: SMLS mark keep failure
6F431808	IGW8SRBC: Start browse failure
6F431809	IGW8SRBC: Write ATRLR failure
6F43180A	IGW8SRBC: Write DSN failure
6F43180B	IGW8SRBC: Zero log token failure
6F43180C	IGW8SRBC: Obtain TVS latch failure
6F43180D	IGW8SRBC: Release TVS latch failure
6F43180E	IGW8SRBC: Free block failure
6F43180F	IGW8SRBC: Tree add failure
6F431810	IGW8SRBC: Locate volser CCHH failure
6F431811	IGW8SRBC: ATRL present failure
6F431812	IGW8SRBC: Obtain shunt latch failure
6F431813	IGW8SRBC: Release shunt latch failure
6F431814	IGW8SRBC: SQM submit failure
6F431815	IGW8SRBC: End browse failure
6F431816	IGW8SRBC: Nullify curb latch failure
6F431817	IGW8SRBC: Invalid log record failure
6F431818	IGW8SRBC: Obtain in-doubt latch failure
6F431819	IGW8SRBC: Release in-doubt latch failure
6F43181F	IGW8SRBC: Indeterminate failure
6F441820	IGW8SRUR: Convert to ESTAE failure
6F441821	IGW8SRUR: Convert to FRR failure
6F441822	IGW8SRUR: Find failure
6F441823	IGW8SRUR: Invalid BTREE token failure
6F441824	IGW8SRUR: Overflow of local TDSL failure
6F441825	IGW8SRUR: Read undo log failure
6F441826	IGW8SRUR: SMLS end transaction failure
6F441827	IGW8SRUR: SMLS mark keep failure
6F441828	IGW8SRUR: Start browse failure
6F441829	IGW8SRUR: Write ATRLR failure
6F44182A	IGW8SRUR: Write DSN failure

Shunt processing reason codes

Reason Code	Description
6F44182B	IGW8SRUR: Zero log token failure
6F44182C	IGW8SRUR: Obtain TVS latch failure
6F44182D	IGW8SRUR: Release TVS latch failure
6F44182E	IGW8SRUR: Free block failure
6F44182F	IGW8SRUR: Tree add failure
6F441830	IGW8SRUR: Locate volser CCHH failure
6F441831	IGW8SRUR: No active data sets failure
6F441832	IGW8SRUR: ATRL present failure
6F441833	IGW8SRUR: Obtain shunt latch failure
6F441834	IGW8SRUR: Release shunt latch failure
6F441835	IGW8SRUR: SQM submit fFailure
6F441836	IGW8SRUR: End browse Ffailure
6F441837	IGW8SRUR: Nullify curb latch failure
6F441838	IGW8SRUR: Obtain in-doubt latch failure
6F441839	IGW8SRUR: Release in-doubt latch failure
6F44183A	IGW8SRUR: Invalid log record failure
6F44183F	IGW8SRUR: Indeterminate failure
6F451860	IGW8SPBC: Convert to ESTAE failure
6F451861	IGW8SPBC: Convert to FRR failure
6F451862	IGW8SPBC: Invalid B tree token failure
6F451863	IGW8SPBC: Overflow of local TDSL failure
6F451864	IGW8SPBC: Read undo log failure
6F451865	IGW8SPBC: SMLS end transaction failure
6F451866	IGW8SPBC: SMLS mark keep failure
6F451867	IGW8SPBC: Write ATRLR failure
6F451868	IGW8SPBC: Write DSN failure
6F451869	IGW8SPBC: ATRLR present failure
6F45186A	IGW8SPBC: Obtain TVS latch failure
6F45186B	IGW8SPBC: Release TVS latch failure
6F45186C	IGW8SPBC: Free block failure
6F45186D	IGW8SPBC: Tree add failure
6F45186E	IGW8SPBC: Locate volser CCHH failure
6F45186F	IGW8SPBC: Obtain shunt latch failure
6F451870	IGW8SPBC: Release shunt latch failure
6F451871	IGW8SPBC: SQM submit failure
6F451872	IGW8SPBC: Obtain in-doubt latch failure
6F451873	IGW8SPBC: Release in-doubt latch failure
6F451874	IGW8SPBC: Nullify curb latch failure
6F45187F	IGW8SPBC: Indeterminate failure
6F461840	IGW8SPUR: Convert to ESTAE failure
6F461841	IGW8SPUR: Convert to FRR failure
6F461842	IGW8SPUR: B tree token not valid failure
6F461843	IGW8SPUR: Overflow of local TDSL failure
6F461844	IGW8SPUR: SMLS end transaction failure
6F461845	IGW8SPUR: Write ATRLR failure
6F461846	IGW8SPUR: Obtain TVS latch failure
6F461847	IGW8SPUR: Release TVS latch failure
6F461848	IGW8SPUR: Free block failure
6F461849	IGW8SPUR: Tree add failure
6F46184A	IGW8SPUR: Locate volser CCHH failure
6F46184B	IGW8SPUR: SMLS reset subsystem failure
6F46184C	IGW8SPUR: Obtain shunt latch failure
6F46184D	IGW8SPUR: Release shunt latch failure
6F46184E	IGW8SPUR: SQM submit failure

Shunt processing reason codes

Reason Code	Description
6F46184F	IGW8SPUR: Obtain in-doubt latch failure
6F461850	IGW8SPUR: Release in-doubt latch failure
6F461851	IGW8SPUR: Nullify curb latch failure
6F46185F	IGW8SPUR: Indeterminate failure
6F471701	IGW8SLBC: Generate message failure
6F471710	IGW8SLBC: Extract TMIB failure
6F471719	IGW8SLBC: Convert to FRR failure
6F47171A	IGW8SLBC: Convert to ESTAE failure
6F47171D	IGW8SLBC: Obtain TMIB latch failure
6F47171E	IGW8SLBC: Release TMIB latch failure
6F471724	IGW8SLBC: Tree add failure
6F471725	IGW8SLBC: Invalid BTREE token failure
6F471726	IGW8SLBC: Obtain shunt latch failure
6F471727	IGW8SLBC: Release shunt latch failure
6F471733	IGW8SLBC: Obtain command latch failure
6F471734	IGW8SLBC: Release command latch failure
6F481701	IGW8SLUR: Generate message failure
6F481710	IGW8SLUR: Extract TMIB failure
6F481719	IGW8SLUR: Convert to FRR failure
6F48171A	IGW8SLUR: Convert to ESTAE failure
6F48171D	IGW8SLUR: Obtain TMIB latch failure
6F48171E	IGW8SLUR: Release TMIB latch failure
6F481724	IGW8SLUR: Tree add failure
6F481725	IGW8SLUR: Invalid BTREE token failure
6F481726	IGW8SLUR: Obtain shunt latch failure
6F481727	IGW8SLUR: Release shunt latch failure
6F481733	IGW8SLUR: Obtain command latch failure
6F481734	IGW8SLUR: Release command latch failure
6F631289	IGW8RDSN: Obtain chain latch failure
6F63128A	IGW8RDSN: Release chain latch failure
6F63133B	IGW8RDSN: Build a curb failure
6F631340	IGW8RDSN: Obtain shunt latch failure
6F631341	IGW8RDSN: Release shunt latch failure
6F631342	IGW8RDSN: Obtain curb latch failure
6F631343	IGW8RDSN: Release curb latch failure
6F63134F	IGW8RDSN: Indeterminate failure
6F63141D	IGW8RDSN: Get block failure
6F631422	IGW8RDSN: Shunt curb integrity failure
6F631426	IGW8RDSN: Initialize curb latch failure
6F63142C	IGW8RDSN: Initialize endreq close latch
6F631446	IGW8RDSN: FREEMAIN arguments failure
6F631464	IGW8RDSN: Suspend failure
6F631465	IGW8RDSN: PurgeDq failure

Restart reason codes

Restart reason codes

Reason Code	Description
6F031220	IGW8RRES: Restart chain failure
6F031221	IGW8RRES: Shunt chain failure
6F031222	IGW8RRES: Indoubt chain failure
6F031223	IGW8RRES: Shunt curb integrity failure
6F031224	IGW8RRES: Obtain tobhx latch failure

Restart reason codes

Reason Code	Description
6F031225	IGW8RRES: Release tobhx latch failure
6F031226	IGW8RRES: Obtain restart curb latch failure
6F031227	IGW8RRES: Release restart curb latch failure
6F031228	IGW8RRES: Obtain TVS latch failure
6F031229	IGW8RRES: Release TVS latch failure
6F03122A	IGW8RRES: Obtain shunt latch failure
6F03122B	IGW8RRES: Release shunt latch failure
6F03122C	IGW8RRES: Obtain in-doubt latch failure
6F03122D	IGW8RRES: Release in-doubt latch failure
6F03122E	IGW8RRES: Obtain curb latch failure
6F03122F	IGW8RRES: Release curb latch failure
6F031230	IGW8RRES: SMLS end transaction failure
6F031231	IGW8RRES: Destroy chain failure
6F031232	IGW8RRES: Call ATRIRRI failure
6F031233	IGW8RRES: Write ATRL failure
6F031234	IGW8RRES: Convert to ESTAE failure
6F031235	IGW8RRES: Convert to FRR failure
6F031236	IGW8RRES: GETMAIN argument list failure
6F031237	IGW8RRES: Free block failure
6F031238	IGW8RRES: SQM create instance failure
6F031239	IGW8RRES: SQM submit failure
6F03123A	IGW8RRES: Quiesce failure
6F03123B	IGW8RRES: Nullify curb latch failure
6F03123C	IGW8RRES: VP GETMAIN arguments failure
6F03123D	IGW8RRES: Vary SQM submit failure
6F031244	IGW8RRES: Indeterminate failure
6F40125A	IGW8RRTS: SMLS end transaction failure
6F40125B	IGW8RRTS: Destroy chain failure
6F40125C	IGW8RRTS: Call ATRIRRI failure
6F40125D	IGW8RRTS: Write ATRLR failure
6F40125E	IGW8RRTS: FREEMAIN arguments failure
6F40125F	IGW8RRTS: Convert to ESTAE failure
6F401260	IGW8RRTS: Convert to FRR failure
6F401261	IGW8RRTS: Release TVS latch failure
6F401262	IGW8RRTS: Obtain TVS latch failure
6F401263	IGW8RRTS: Release restart latch failure
6F401264	IGW8RRTS: Free block failure
6F401265	IGW8RRTS: Data set name match failure
6F401266	IGW8RRTS: Release shunt latch failure
6F401267	IGW8RRTS: Obtain shunt latch failure
6F401268	IGW8RRTS: IGW8RCHN failure
6F401269	IGW8RRTS: Force redo log failure
6F40126A	IGW8RRTS: system error failure
6F40126B	IGW8RRTS: Invalid tdsl failure
6F40126C	IGW8RRTS: Release curb latch failure
6F40126D	IGW8RRTS: Obtain curb latch failure
6F40126E	IGW8RRTS: Write force failure
6F40126F	IGW8RRTS: Shunt curb integrity failure
6F401274	IGW8RRTS: Indeterminate failure

Peer recovery reason codes

Peer recovery reason codes

Reason Code	Description
6F1F18C0	IGW8PCLN: LaterAbendFailure
6F5B1018	IGW8PIN1: ACB_Pool_failure
6F5B101A	IGW8PIN1: ConvertToEstae failure
6F5B1019	IGW8PIN1: ConvertToFrr failure
6F5B1004	IGW8PIN1: CURB_Pool failure
6F5B101B	IGW8PIN1: GetmainArgs failure
6F5B1000	IGW8PIN1: GetmainPeerTmib failure
6F5B108E	IGW8PIN1: GetmainPeerTmibY failure
6F5B10C8	IGW8PIN1: Init_RestartChainLatch failure
6F5B10D4	IGW8PIN1: Initial_CurrentLatch failure
6F5B1098	IGW8PIN1: Initial_IgwLogsLatch failure
6F5B1077	IGW8PIN1: Initial_InDoubtLatch failure
6F5B1073	IGW8PIN1: Initial_SHUNTLatch failure
6F5B1002	IGW8PIN1: InitSQMCreate failure
6F5B1003	IGW8PIN1: InitSQMSubmit failure
6F5B1055	IGW8PIN1: Logs_Pool failure
6F5B1029	IGW8PIN1: Mdta_Pool_failure
6F5B10EC	IGW8PIN1: NCRB_Pool failure
6F5B10B0	IGW8PIN1: ObtainPeerChainLatch failure
6F5B101D	IGW8PIN1: ObtainTmibLatch failure
6F5B10B1	IGW8PIN1: ReleasePeerChainLatch failure
6F5B101E	IGW8PIN1: ReleaseTmibLatch failure
6F5B1016	IGW8PIN1: Tdsl_Pool failure
6F5B1023	IGW8PIN1: Tmib_ExtractToken failure
6F5B1012	IGW8PIN1: Tobe_Pool failure
6F5B1013	IGW8PIN1: Tobh_Pool failure
6F5C101A	IGW8PIN2: ConvertToEstae_Failure failure
6F5C1019	IGW8PIN2: ConvertToFrr failure
6F5C10B7	IGW8PIN2: DiscLogofLogs failure
6F5C10B6	IGW8PIN2: FreeACBPool failure
6F5C1092	IGW8PIN2: FreeCurbBlock failure
6F5C10A1	IGW8PIN2: FreeCurbPool failure
6F5C10A8	IGW8PIN2: FreeLogsPool failure
6F5C108F	IGW8PIN2: FreemainPeerTmib failure
6F5C1090	IGW8PIN2: FreemainPeerTmibY failure
6F5C10A2	IGW8PIN2: FreeMdtaPool failure
6F5C1067	IGW8PIN2: FreeTdslBlock failure
6F5C10A0	IGW8PIN2: FreeTdslPool failure
6F5C10A3	IGW8PIN2: FreeTobePool failure
6F5C10A4	IGW8PIN2: FreeTobhPool failure
6F5C10AC	IGW8PIN2: LogofLogsQuiesce failure
6F5C10DC	IGW8PIN2: NULL_RestartChainLatch failure
6F5C10E7	IGW8PIN2: Nullify_CECLatch failure
6F5C10E8	IGW8PIN2: Nullify_CurbLatch failure
6F5C10DA	IGW8PIN2: NULLIFY_CurrentLatch failure
6F5C10E2	IGW8PIN2: NULLIFY_IgwLogsLatch failure
6F5C10D8	IGW8PIN2: NULLIFY_InDoubtLatch failure
6F5C10DE	IGW8PIN2: NULLIFY_SHUNTLatch failure
6F5C10C2	IGW8PIN2: ObtainPeerChainLatch failure
6F5C10F3	IGW8PIN2: ObtainSharedLatch failure
6F5C101D	IGW8PIN2: ObtainTmibLatch failure
6F5C10D9	IGW8PIN2: OBTC_CurrentLatch failure

Peer recovery reason codes

Reason Code	Description
6F5C10E1	IGW8PIN2: OBTC_IgwLogsLatch failure
6F5C10D7	IGW8PIN2: OBTC_InDoubtLatch failure
6F5C10DB	IGW8PIN2: OBTC_RestartChainLatch failure
6F5C10DD	IGW8PIN2: OBTC_SHUNTLatch failure
6F5C10F5	IGW8PIN2: PrepareSharedLatch failure
6F5C10B1	IGW8PIN2: ReleasePeerChainLatch failure
6F5C10F4	IGW8PIN2: ReleaseSharedLatch failure
6F5C101E	IGW8PIN2: ReleaseTmibLatch failure
6F5C1005	IGW8PIN2: shcds_Connect failure
6F5C1089	IGW8PIN2: ShuntLogQuiesce_failure
6F5C1008	IGW8PIN2: SMLS_IdSubsys failure
6F5C10BA	IGW8PIN2: SMLSUnidentify failure
6F5C10D1	IGW8PIN2: TLSPeerTerminate failure
6F5C108A	IGW8PIN2: TVSQuiesce failure
6F5C1088	IGW8PIN2: UndoLogQuiesce failure

Syncpoint reason codes

Syncpoint reason codes

Reason Code	Description
6F551700	IGW8RFRR: Kill user address space
6F2A137F	IGW8ROAE: Indeterminate failure
6F421470	IGW8RRMC: Convert to ESTAE failure
6F421471	IGW8RRMC: Convert to FRR failure
6F421476	IGW8RRMC: FREEMAIN failure
6F421472	IGW8RRMC: GETMAIN LR failure
6F421475	IGW8RRMC: VRM failure
6F5613F2	IGW8RSCH: Extract TMIB failure
6F5613F0	IGW8RSCH: Extract VRGB failure
6F5613F1	IGW8RSCH: Get block failure
6F5612F5	IGW8RSCH: LRR curb not valid
6F5612F3	IGW8RSCH: Suspend failure
6F5612FF	IGW8RSCH: Unexpected error
6F5418E2	IGW8RSR1: Curb not valid
6F5418E6	IGW8RSR1: Extract VRGB failure
6F5418E0	IGW8RSR1: IDAVRARR abnormally ending
6F5418E7	IGW8RSR1: Caller not valid
6F5418E1	IGW8RSR1: No curb pointer
6F5418E4	IGW8RSR1: Resume failure
6F5418E5	IGW8RSR1: Server recycled
6F5418EF	IGW8RSR1: Unexpected error
6F5418E3	IGW8RSR1: Zero suspend
6F6013E9	IGW8RSR2: Curb not valid
6F6013E6	IGW8RSR2: Extract VRGB failure
6F6013EA	IGW8RSR2: GETMAIN failed
6F6013E4	IGW8RSR2: IGW8RFRR end server
6F6013E5	IGW8RSR2: No curb pointer from IDAVRFRR
6F6013E1	IGW8RSR2: No curb pointer from IGW8RSRB
6F6013E7	IGW8RSR2: Server recycled
6F6013E0	IGW8RSR2: TCB token failure
6F6013EE	IGW8RSR2: Unexpected error
6F6013E8	IGW8RSR2: Zero suspend
6F6113DF	IGW8RSR3: Resume failure

Syncpoint reason codes

Reason Code	Description
6F6113E3	IGW8RSR3: Resume SR3 failure
6F6113DE	IGW8RSR3: Suspend failure
6F6113EF	IGW8RSR3: Unexpected error
6F6218F1	IGW8RSR4: FREEMAIN failure
6F6218F0	IGW8RSR4: Resume failure
6F6218FF	IGW8RSR4: Unexpected error
6F1D146A	IGW8RVTV: GETMAIN arguments failure
6F1D146F	IGW8RVTV: Indeterminate failure
6F1D146B	IGW8RVTV: Vary SQM submit failure
6F0C1200	IGW8RPRP: GETMAIN argument list failure
6F0C1201	IGW8RPRP: SQM submit failure
6F0C1202	IGW8RPRP: Convert to ESTAE failure
6F0C1203	IGW8RPRP: Convert to FRR failure
6F0C1204	IGW8RPRP: Generate message failure
6F0C1205	IGW8RPRP: Obtain TVS latch failure
6F0C1206	IGW8RPRP: Release TVS latch failure
6F0C1207	IGW8RPRP: FREEMAIN argument list failure
6F0C1208	IGW8RPRP: Indeterminate failure (0C4, and so on)
6F0C1209	IGW8RPRP: S token extract failure
6F0C120A	IGW8RPRP: Bad VTLB eyecatcher failure
6F0D1211	IGW8RPR1: RPL chain failure
6F0D1212	IGW8RPR1: Force redo log failure
6F0D1213	IGW8RPR1: Obtain TVS latch failure
6F0D1214	IGW8RPR1: Release TVS latch failure
6F0D1215	IGW8RPR1: Convert to ESTAE failure
6F0D1216	IGW8RPR1: Convert to FRR failure
6F0D1217	IGW8RPR1: Generate message failure
6F0D1218	IGW8RPR1: FREEMAIN argument list failure
6F0D1219	IGW8RPR1: ATR failure
6F0D121A	IGW8RPR1: Indeterminate failure (0C4, and so on)
6F0E1300	IGW8RCMT: GETMAIN arguments failure
6F0E1301	IGW8RCMT: SQM submit failure
6F0E1302	IGW8RCMT: Convert to ESTAE failure
6F0E1303	IGW8RCMT: Convert to FRR failure
6F0E1304	IGW8RCMT: Generate message failure
6F0E1305	IGW8RCMT: Obtain TVS latch failure
6F0E1306	IGW8RCMT: Release TVS latch failure
6F0E1310	IGW8RCMT: FREEMAIN arguments failure
6F0E1311	IGW8RCMT: Indeterminate failure
6F0F1320	IGW8RCMS: Convert to FRR failure
6F0F1321	IGW8RCMS: Convert to ESTAE failure
6F0F1322	IGW8RCMS: Generate message failure
6F0F1323	IGW8RCMS: Free block failure
6F0F1324	IGW8RCMS: FREEMAIN arguments failure
6F0F1325	IGW8RCMS: SMLS mark keep failure
6F0F1326	IGW8RCMS: SMLS end transaction failure
6F0F1327	IGW8RCMS: VRM failure
6F0F1328	IGW8RCMS: Read undo log failure
6F0F1329	IGW8RCMS: VRM logical failure
6F0F1330	IGW8RCMS: Write ATRLR failure
6F0F1331	IGW8RCMS: Obtain TVS latch failure
6F0F1332	IGW8RCMS: Release TVS latch failure
6F0F1335	IGW8RCMS: SHCDS disconnect Failure
6F0F1336	IGW8RCMS: Resource recovery services (RRS) environment failure

Syncpoint reason codes

Reason Code	Description
6F0F1337	IGW8RCMS: SQM submit failure
6F0F1338	IGW8RCMS: TVS quiesce failure
6F0F133A	IGW8RCMS: Shunt log quiesce failure
6F0F133B	IGW8RCMS: Build a curb failure
6F0F133C	IGW8RCMS: CRGDRM failure
6F0F133D	IGW8RCMS: SMLS unidentify subsystem failure
6F0F133E	IGW8RCMS: Undolog disconnect failure
6F0F133F	IGW8RCMS: Quiesce failure
6F0F1340	IGW8RCMS: Obtain shunt latch failure
6F0F1341	IGW8RCMS: Release shunt latch failure
6F0F1342	IGW8RCMS: Obtain curb latch failure
6F0F1343	IGW8RCMS: Release curb latch failure
6F0F134F	IGW8RCMS: Indeterminate failure
6F4D1880	IGW8RCID: SMLS mark keep failure
6F4D1881	IGW8RCID: SMLS end transaction failure
6F4D1882	IGW8RCID: Convert to ESTAE failure
6F4D1883	IGW8RCID: Convert to FRR failure
6F4D1884	IGW8RCID: Resource recovery services (RRS) environment failure
6F4D1885	IGW8RCID: FREEMAIN arguments failure
6F4D1886	IGW8RCID: Obtain TVS latch failure
6F4D1887	IGW8RCID: Release TVS latch failure
6F4D1888	IGW8RCID: Obtain in-doubt latch failure
6F4D1889	IGW8RCID: Release in-doubt latch failure
6F4D188A	IGW8RCID: Obtain shunt latch failure
6F4D188B	IGW8RCID: Release shunt latch failure
6F4D188C	IGW8RCID: SQM submit failure
6F4D188D	IGW8RCID: SHCDS connect failure
6F4D188E	IGW8RCID: SHCDS disconnect failure
6F4D188F	IGW8RCID: SHCDS add line failure
6F4D1890	IGW8RCID: Free block failure
6F4D1891	IGW8RCID: Obtain curb latch failure
6F4D1892	IGW8RCID: Release curb latch failure
6F4D1893	IGW8RCID: Nullify curb latch failure
6F4D189F	IGW8RCID: Indeterminate failure
6F101441	IGW8RBOU: Convert to ESTAE failure
6F101442	IGW8RBOU: Convert to FRR failure
6F101443	IGW8RBOU: SQM submit failure
6F101444	IGW8RBOU: Obtain TVS latch failure
6F101445	IGW8RBOU: Release TVS latch failure
6F101446	IGW8RBOU: FREEMAIN arguments failure
6F101447	IGW8RBOU: GETMAIN arguments failure
6F101448	IGW8RBOU: Indeterminate failure
6F101449	IGW8RBOU: Stoken extract failure
6F10144A	IGW8RBOU: Bad VTLB eye-catcher failure
6F121450	IGW8RBOS: RPL chain failure
6F121451	IGW8RBOS: Force redolog failure
6F121452	IGW8RBOS: Generate message failure
6F121453	IGW8RBOS: FREEMAIN arguments failure
6F121454	IGW8RBOS: Convert to ESTAE failure
6F121455	IGW8RBOS: Convert to FRR failure
6F121456	IGW8RBOS: Free block failure
6F121457	IGW8RBOS: ATR failure
6F121458	IGW8RBOS: Release TVS latch failure
6F121459	IGW8RBOS: Obtain TVS latch failure

Syncpoint reason codes

Reason Code	Description
6F12145A	IGW8RBOS: SQM submit failure
6F12145B	IGW8RBOS: TVS quiesce failure
6F12145C	IGW8RBOS: Undolog quiesce failure
6F12145D	IGW8RBOS: Shunt log quiesce failure
6F12145E	IGW8RBOS: CRGDRM failure
6F12145F	IGW8RBOS: SMLS unidentify subsystem failure
6F121460	IGW8RBOS: Undolog disconnect failure
6F121461	IGW8RBOS: Quiesce failure
6F121462	IGW8RBOS: Obtain curb latch failure
6F121463	IGW8RBOS: Release curb latch failure
6F121464	IGW8RBOS: Suspend failure
6F121465	IGW8RBOS: Purge deq failure
6F12146F	IGW8RBOS: Indeterminate failure
6F111400	IGW8RIOM: SMLS mark keep failure
6F111401	IGW8RIOM: SMLS end transaction failure
6F111402	IGW8RIOM: VRM failure
6F111403	IGW8RIOM: Read undolog failure
6F111404	IGW8RIOM: VRM logical failure
6F111405	IGW8RIOM: Write ATRLR failure
6F111407	IGW8RIOM: entry not found failure
6F111408	IGW8RIOM: Generate message failure
6F111409	IGW8RIOM: GETMAIN failure
6F11140A	IGW8RIOM: FREEMAIN failure
6F111411	IGW8RIOM: Start browse failure
6F111412	IGW8RIOM: Entry not found failure
6F111413	IGW8RIOM: SMLS mark keep all failure
6F111414	IGW8RIOM: Write shunt failure
6F111415	IGW8RIOM: Get new curb failure
6F111416	IGW8RIOM: SHCDS connect failure
6F111417	IGW8RIOM: SHCDS disconnect failure
6F111418	IGW8RIOM: SHCDS add line failure
6F111419	IGW8RIOM: Convert to ESTAE failure
6F11141A	IGW8RIOM: Convert to FRR failure
6F11141B	IGW8RIOM: Obtain TVS latch failure
6F11141C	IGW8RIOM: Release TVS latch failure
6F11141D	IGW8RIOM: Get block failure
6F11141E	IGW8RIOM: Zero data set ACB failure
6F11141F	IGW8RIOM: Obtain shunt latch failure
6F111420	IGW8RIOM: Release shunt latch failure
6F111421	IGW8RIOM: Free block failure
6F111422	IGW8RIOM: Shunt Curb integrity failure
6F111423	IGW8RIOM: Obtain in-doubt latch failure
6F111424	IGW8RIOM: Release in-doubt latch failure
6F111425	IGW8RIOM: End browse failure
6F111426	IGW8RIOM: Initialize curb latch failure
6F111427	IGW8RIOM: Obtain curb latch failure
6F111428	IGW8RIOM: Release curb latch failure
6F11142F	IGW8RIOM: Indeterminate failure
6F271430	IGW8RCHN: Convert to FRR failure
6F271431	IGW8RCHN: Convert to ESTAE failure
6F271432	IGW8RCHN: VRM failure
6F271433	IGW8RCHN: RPL not unchained failure
6F271434	IGW8RCHN: Indeterminate failure
6F4E18A0	IGW8RBID: SMLS mark keep failure

Syncpoint reason codes

Reason Code	Description
6F4E18A1	IGW8RBID: SMLS end transaction failure
6F4E18A2	IGW8RBID: Convert to ESTAE failure
6F4E18A3	IGW8RBID: Convert to FRR failure
6F4E18A4	IGW8RBID: Resource recovery services (RRS) environment failure
6F4E18A5	IGW8RBID: FREEMAIN arguments failure
6F4E18A6	IGW8RBID: Obtain TVS latch failure
6F4E18A7	IGW8RBID: Release TVS latch failure
6F4E18A8	IGW8RBID: Obtain in-doubt latch failure
6F4E18A9	IGW8RBID: Release in-doubt latch failure
6F4E18AA	IGW8RBID: Obtain shunt latch failure
6F4E18AB	IGW8RBID: Release shunt latch failure
6F4E18AC	IGW8RBID: SQM submit failure
6F4E18B0	IGW8RBID: Free block failure
6F4E18B1	IGW8RBID: Obtain curb latch failure
6F4E18B2	IGW8RBID: Release curb latch failure
6F4E18B3	IGW8RBID: Nullify curb latch failure
6F4E18BF	IGW8RBID: Indeterminate failure
6F151245	IGW8ROP2: ACB pool failure
6F151246	IGW8ROP2: Convert to ESTAE failure
6F151247	IGW8ROP2: Convert to FRR failure
6F151248	IGW8ROP2: Generate message failure
6F151249	IGW8ROP2: Free block failure
6F15124F	IGW8ROP2: Indeterminate failure
6F161250	IGW8RCLS: Convert to ESTAE failure
6F161251	IGW8RCLS: Convert to FRR failure
6F161252	IGW8RCLS: Generate message failure
6F161253	IGW8RCLS: Free block failure
6F161259	IGW8RCLS: Indeterminate failure

Miscellaneous reason codes

Miscellaneous reason codes

Reason Code	Description
6F191275	IGW8RNFY: Obtain TVS latch failure
6F191276	IGW8RNFY: Release TVS latch failure
6F191277	IGW8RNFY: Convert to ESTAE failure
6F191278	IGW8RNFY: Convert to FRR failure
6F191279	IGW8RNFY: Invalid exit manager failure
6F19127A	IGW8RNFY: GETMAIN arguments failure
6F19127B	IGW8RNFY: CTXEU requested failure
6F19127C	IGW8RNFY: CTXEU bad return code failure
6F19127D	IGW8RNFY: CTXEU exit failed failure
6F19127E	IGW8RNFY: Resource recovery services (RRS) EU requested failure
6F19127F	IGW8RNFY: Resource recovery services (RRS) EU bad return code failure
6F191280	IGW8RNFY: Resource recovery services (RRS) EU exit failed exit failed failure
6F191281	IGW8RNFY: CTX invalid pval1 failure
6F191282	IGW8RNFY: Resource recovery services (RRS) invalid pval1 failure
6F191283	IGW8RNFY: CTX invalid pval2 failure
6F191284	IGW8RNFY: RRS (resource recovery services) invalid pval2 failure
6F191285	IGW8RNFY: Invalid TMIB state failure
6F191286	IGW8RNFY: Init SQM submit failure

Miscellaneous reason codes

Reason Code	Description
6F191287	IGW8RNFY: Free block failure
6F191288	IGW8RNFY: SMLS retain all locks failure
6F1912FF	IGW8RNFY: Indeterminate failure
6F091350	IGW8RECE: Obtain TVS latch failure
6F091351	IGW8RECE: Release TVS latch failure
6F091352	IGW8RECE: End transaction failure
6F091353	IGW8RECE: Free tdsl block failure
6F091355	IGW8RECE: Mark keep failure
6F091356	IGW8RECE: Curb set token failure
6F091357	IGW8RECE: Free Curb block failure
6F091358	IGW8RECE: Obtain Curb latch failure
6F091359	IGW8RECE: Nullify Curb latch failure
6F091360	IGW8RECE: Release Curb latch failure
6F091361	IGW8RECE: Destroy chain failure
6F091362	IGW8RECE: Nonzero urid failure
6F09136F	IGW8RECE: Indeterminate failure
6F4F1501	IGW8RAKP: AKP_CS_bad_1
6F4F1502	IGW8RAKP: AKP_CS_bad_2
6F4F1506	IGW8RAKP: chaintoken_zero_1
6F4F1507	IGW8RAKP: chaintoken_zero_2
6F4F150A	IGW8RAKP: CreateChainFailure
6F4F1505	IGW8RAKP: DeleteHistoryFailure
6F4F1504	IGW8RAKP: DestroyChainFailure
6F4F150B	IGW8RAKP: GetCurrentCurbChainLatchFailure
6F4F150B	IGW8RAKP: GetrestartCurbChainLatchFailure
6F4F1509	IGW8RAKP: MoveChainFailure
6F4F1510	IGW8RAKP: ObtainCurbLatchFailure
6F4F1516	IGW8RAKP: ObtainIndoubtLatchFailure
6F4F1514	IGW8RAKP: ObtainShuntLatchFailure
6F4F150C	IGW8RAKP: RelCurrentCurbChainLatchFailure
6F4F1511	IGW8RAKP: ReleaseCurbLatchFailure
6F4F1517	IGW8RAKP: ReleaseIndoubtLatchFailure
6F4F1515	IGW8RAKP: ReleaseShuntLatchFailure
6F4F150C	IGW8RAKP: RelrestartCurbChainLatchFailure
6F4F1503	IGW8RAKP: SetHistoryFailure
6F4F1500	IGW8RAKP: TmibExtractFailure
6F4F151A	IGW8RAKP: unexpected_error
6F4F1508	IGW8RAKP: WriteUNDOLogFailure
6F1B1383	IGW8RCEF: CTXEndCtxtExit_Failed
6F1B1384	IGW8RCEF: CTXEOMCtxtExit_Failed
6F1B1382	IGW8RCEF: CTXSwitchExit_Failed
6F1B138F	IGW8RCEF: Indeterminate failure
6F1B1380	IGW8RCEF: Non-DFSMStvs exit value
6F1B1381	IGW8RCEF: Non-DFSMStvs reason value
6F501600	IGW8RCSE: Curb extract token failure
6F501602	IGW8RCSE: Curb set token failure 1
6F501603	IGW8RCSE: Curb set token failure 2
6F50160C	IGW8RCSE: Curb set token failure 3
6F501609	IGW8RCSE: Delete context interest failure
6F50160A	IGW8RCSE: Obtain restart latch failure
6F50160B	IGW8RCSE: Release restart latch failure
6F501605	IGW8RCSE: S token extract failure
6F50160D	IGW8RCSE: TMIB extract token failure 1
6F50160E	IGW8RCSE: TMIB extract token failure 2

Miscellaneous reason codes

Reason Code	Description
6F501601	IGW8RCSE: Token already set
6F501604	IGW8RCSE: Tokenc mismatch
6F501606	IGW8RCSE: T token extract failure
6F50160F	IGW8RCSE: Unexpected error
6F5F16A3	IGW8RCS1: Curb set token failure 1
6F5F16A4	IGW8RCS1: Curb set token failure 2
6F5F16A2	IGW8RCS1: Unexpected error
6F0A1720	IGW8REOM: Resume failure
6F0A1721	IGW8REOM: Unexpected error
6F5E13D0	IGW8REOX: Get current curb chain latch failure
6F5E13D4	IGW8REOX: Obtain curb latch failure
6F5E13D1	IGW8REOX: Release current curb chain latch failure
6F5E13D5	IGW8REOX: Release curb latch failure
6F5E13D6	IGW8REOX: Resume failure
6F5E13CF	IGW8REOX: Unexpected error
6F5E13CE	IGW8REOX: Unknown caller
6F2318D0	IGW8RPC1: Indeterminate failure
6F2418D4	IGW8RPC2: Indeterminate failure
6F2518DA	IGW8RPC3: Indeterminate failure
6F2518DB	IGW8RPC3: Instance count mismatch
6F3C139F	IGW8RREF: Indeterminate failure
6F3C139A	IGW8RREF: Invalid Pval1 failure
6F3C139C	IGW8RREF: Invalid Pval2 failure
6F3C139B	IGW8RREF: Invalid RC failure

Logging reason code prefixes

Logging reason code prefixes

Module	Reason code prefix	Brief description
IGW9LING	7001	LOGGER INITIALIZATION
IGW9LWFG	7002	WRITE AND FORCE UNDO LOG
IGW9LCBG	7003	CHAIN BROWSE
IGW9LCCG	7004	CHAIN CONTROL
IGW9LME1	7005	EXECUTE MESSAGE
IGW9LGLG	7006	GENERAL LOG GATE
IGW9LMVG	7007	MOVE CHAIN GATE
IGW9LBAG	7008	BROWSE ALL GATE
IGW9LDS1	7009	SRB STIMER MODULE
IGW9LREC	700A	RECOVERY (CLEANUP) ROUTINE
DFHL2OFC	7065	DFHL2OFC MACRO
DFHL2VPC	7066	DFHL2VPC MACRO
DFHL2SYC	7067	DFHL2SYC MACRO
HCCKERN	7068	HCCKERN MACRO
IGWL2LMC	7069	LOCK MANAGER
IGW9LDSC	706A	DISPATCHER CLASS
IGW9LBL1	70C8	BLOCK

Logging reason code prefixes

Module	Reason code prefix	Brief description
IGW9LBL2	70C9	BLOCK
IGW9LBS1	70CA	BROWSABLE STREAM
IGW9LBS2	70CB	BROWSABLE STREAM
IGW9LBS3	70CC	BROWSABLE STREAM
IGW9LBS4	70CD	BROWSABLE STREAM
IGW9LCHA	70CE	CHAIN
IGW9LCHE	70CF	CHAIN
IGW9LCHG	70D0	CHAIN
IGW9LCHH	70D1	CHAIN
IGW9LCHI	70D2	CHAIN
IGW9LCHL	70D3	CHAIN
IGW9LCHM	70D4	CHAIN
IGW9LCHN	70D5	CHAIN
IGW9LCHO	70D6	CHAIN
IGW9LCHP	70D7	CHAIN
IGW9LCHR	70D8	CHAIN
IGW9LCHS	70D9	CHAIN
IGW9LCH1	70DA	CHAIN
IGW9LCH2	70DB	CHAIN
IGW9LCH3	70DC	CHAIN
IGW9LCH4	70DD	CHAIN
IGW9LCH5	70DE	CHAIN
IGW9LHSF	70DF	HARD STREAM
IGW9LHSG	70E0	HARD STREAM
IGW9LHSJ	70E1	HARD STREAM
IGW9LHS2	70E2	HARD STREAM
IGW9LHS3	70E3	HARD STREAM
IGW9LHS4	70E4	HARD STREAM
IGW9LHS5	70E5	HARD STREAM
IGW9LHS6	70E6	HARD STREAM
IGW9LHS7	70E7	HARD STREAM
IGW9LHS8	70E8	HARD STREAM
IGW9LHS9	70E9	HARD STREAM
IGW9LSLE	70EA	SYSTEM LOG
IGW9LSLN	70EB	SYSTEM LOG
IGW9LSL1	70EC	SYSTEM LOG
IGW9LSR1	70ED	STREAM
IGW9LSR2	70EE	STREAM
IGW9LSR3	70EF	STREAM
IGW9LSR4	70F0	STREAM

Logging reason code prefixes

Module	Reason code prefix	Brief description
IGW9LSR5	70F1	STREAM

Logging services reason codes

Logging services reason codes

Reason Code	Description
70010001	IGW9LING: Initialize class failed
70010002	IGW9LING: GETMAIN failed
70010003	IGW9LING: IGWFTOKM SET failed
70010004	IGW9LING: LGA pointer is zero
70010005	IGW9LING: Primary log stream name is a required parameter
70010006	IGW9LING: Secondary log stream name is a required parameter
70010007	IGW9LING: Log of logs name is a required parameter
70010008	IGW9LING: Log of logs name is invalid
70010009	IGW9LING: Log of logs token is a required parameter
7001000A	IGW9LING: keypoint frequency is a required parameter
7001000B	IGW9LING: IGWFTOKM EXTRACT failed
7001000C	IGW9LING: Set keypoint frequency failed
7001000D	IGW9LING: SQM CREATE failed
7001000E	IGW9LING: Log of logs not defined
7001000F	IGW9LING: Log of logs connect failure
70010010	IGW9LING: Unexpected log of logs connect return code
70010011	IGW9LING: Convert to ESTAE failed
70010012	IGW9LING: Convert to ESTAE failed
70010013	IGW9LING: Convert to FRR failed
70010014	IGW9LING: Convert to FRR failed
70010015	IGW9LING: Start type is a required parameter
70010016	IGW9LING: Connect to system logs failed
70010017	IGW9LING: IGWLSIXL failed
70010018	IGW9LING: APPLID is a required parameter
70020001	IGW9LWFG: An invalid function was requested
70020002	IGW9LWFG: Buffer is full
70020003	IGW9LWFG: Buffer length error
70020004	IGW9LWFG: Purged
70020005	IGW9LWFG: Unknown error
70020006	IGW9LWFG: IGWFTOKM failed
70020007	IGW9LWFG: Undo log record common section is a required parameter
70020008	IGW9LWFG: FORCE is a required parameter
70020009	IGW9LWFG: Not all parameters required for writing data were passed. A write of data required a pointer, ALET, key, length, undo log record common area, and undo log record data area
7002000A	IGW9LWFG: Chain token is zero
7002000B	IGW9LWFG: LGA pointer is zero
7002000C	IGW9LWFG: SQM SUBMIT failed
7002000D	IGW9LWFG: DFSMStvs terminated
7002000E	IGW9LWFG: The undo log record common section has a bad eye-catcher
7002000F	IGW9LWFG: The undo log is broken
70026908	IGW9LWFG: Latch purged and nullified
70030001	IGW9LCBG: An invalid function was requested
70030002	IGW9LCBG: End of data

Logging services reason codes

Reason Code	Description
70030003	IGW9LCBG: IGWFTOKM failed
70030004	IGW9LCBG: Data pointer is a required parameter
70030005	IGW9LCBG: Data length is a required parameter
70030006	IGW9LCBG: Chain token is zero
70030007	IGW9LCBG: LGA pointer is zero
70030008	IGW9LCBG: DFSMStvs terminated
70030009	IGW9LCBG: Latch purged and nullified
7003000A	IGW9LCBG: The undo log is broken
70040001	IGW9LCCG: An invalid function was requested
70040002	IGW9LCCG: IGWFTOKM failed
70040003	IGW9LCCG: No more chains
70040004	IGW9LCCG: User token is a required parameter
70040005	IGW9LCCG: Chain token is a required parameter
70040006	IGW9LCCG: Set keypoint failed
70040007	IGW9LCCG: Keypoint value is out of range
70040008	IGW9LCCG: Chain token is zero
70040009	IGW9LCCG: LGA pointer is zero
7004000A	IGW9LCCG: The undo log is broken
7004000B	IGW9LCCG: DFSMStvs terminated
70046908	IGW9LCCG: Latch purged and nullified
70050001	IGW9LME1: IGWFTOKM failed
70050002	IGW9LME1: LGA pointer is zero
70050003	IGW9LME1: The caller of execute message requested that the server be terminated
70050004	IGW9LME1: The caller of execute message passed a message ID for which there is no message support
70050005	IGW9LME1: One of the message inserts either has a zero length or a zero address
70050006	IGW9LME1: A message insert is too long. Inserts are limited to 72 bytes long (the length of a single line on the console)
70050007	IGW9LME1: A provided message insert is not supported. The insert was not one of the expected inserts
70060001	IGW9LGLG: Log stream name is a required parameter for connect
70060002	IGW9LGLG: Log token is a required parameter for all functions but disconnect syslogs
70060003	IGW9LGLG: Log token is zero
70060005	IGW9LGLG: FORCE is a required parameter for a write
70060006	IGW9LGLG: GLGP is a required parameter for a write
70060007	IGW9LGLG: GLGP array size is bad
70060008	IGW9LGLG: GLGP data length is zero
70060009	IGW9LGLG: GLGP data pointer is zero
7006000A	IGW9LGLG: LGA pointer is zero
7006000B	IGW9LGLG: Log not defined on a connect request - unable to find specified log
7006000C	IGW9LGLG: Connect request failed
7006000D	IGW9LGLG: Unexpected connect return code from connect
70060001	IGW9LGLG: Log stream name is a required parameter for connect
70060002	IGW9LGLG: Log token is a required parameter for all functions but disconnect syslogs
70060003	IGW9LGLG: Log token is zero
70060005	IGW9LGLG: FORCE is a required parameter for a write
70060006	IGW9LGLG: GLGP is a required parameter for a write
70060007	IGW9LGLG: GLGP array size is bad
70060008	IGW9LGLG: GLGP data length is zero

Logging services reason codes

Reason Code	Description
70060009	IGW9LGLG: GLGP data pointer is zero
7006000A	IGW9LGLG: LGA pointer is zero
7006000B	IGW9LGLG: Log not defined on a connect request - unable to find specified log
7006000C	IGW9LGLG: Connect request failed
7006000D	IGW9LGLG: Unexpected connect return code from connect
7006000E	IGW9LGLG: Unexpected return code from disconnect
7006000F	IGW9LGLG: Unexpected return code from write
70060010	IGW9LGLG: Unexpected return code from force
70060011	IGW9LGLG: A terminate request failed with an unexpected return code
70060012	IGW9LGLG: IGWFTOKM failed
70060013	IGW9LGLG: An invalid function was requested
70060014	IGW9LGLG: Buffer length error
70060015	IGW9LGLG: Lost data
70060016	IGW9LGLG: Lost access
70060017	IGW9LGLG: A terminate all request failed with an unexpected return code
70060018	IGW9LGLG: Convert to ESTAE failed
70060019	IGW9LGLG: Convert to FRR failed
7006001A	IGW9LGLG: Force token is a required parameter for a write
7006001B	IGW9LGLG: Force token is a required parameter for a write
7006001C	IGW9LGLG: Unexpected buffer full
70070001	IGW9LMVG: IGWFTOKM failed
70070002	IGW9LMVG: Chain token is zero
70070003	IGW9LMVG: An invalid function was requested
70070004	IGW9LMVG: Chain move failed
70070005	IGW9LMVG: LGA pointer is zero
70070006	IGW9LMVG: DFSMStvs terminated
70070007	IGW9LMVG: The undo log is broken
70076908	IGW9LMVG: Latch purged and nullified
70080001	IGW9LBAG: An invalid function was requested
70080002	IGW9LBAG: End of data
70080003	IGW9LBAG: IGWFTOKM failed
70080004	IGW9LBAG: Unexpected START return code
70080005	IGW9LBAG: Unexpected END return code
70080006	IGW9LBAG: User token is a required parameter
70080007	IGW9LBAG: Data pointer is a required parameter
70080008	IGW9LBAG: Data length is a required parameter
70080009	IGW9LBAG: LGA pointer is zero
7008000A	IGW9LBAG: DFSMStvs terminated
7008000B	IGW9LBAG: Keypoint was ignored
7008000C	IGW9LBAG: Unexpected GETNEXT return code
70090001	IGW9LDS1: Unable to change recovery between ESTAE and FRR
70090002	IGW9LDS1: Unable to change recovery between ESTAE and FRR
70090003	IGW9LDS1: IGWFTOKM failed
70090004	IGW9LDS1: LGA pointer is zero
70650001	Create pool failed
70650002	Get block failed
70650003	Free block failed
70660001	GETMAIN failure
70660002	FREEMAIN failure
70670001	Unsupported log name
70670002	IGW9LSYC: The DFSMStvs logger terminated the server

Logging services reason codes

Reason Code	Description
70670003	IGW9LSYC: SQM SUBMIT failed
70670001	IGWLSOXL failed
70670002	IGWLSRXL failed
70680001	IGWFTOKM failed
70690001	IGW9LLMC: IGWLSIXL failed - initialize latch
70690002	IGW9LLMC: IGWLSOXL failed - obtain latch
70690003	IGW9LLMC: IGWLSRXL failed - release latch
70690004	IGW9LLMC: IGWLSNXL failed - nullify latch
70690005	IGW9LLMC: IGWLSRXL failed - release latch
70690006	IGW9LLMC: Exceeded LRA maximum LOCKS
70690007	IGW9LLMC: Lock not found
70690008	IGW9LLMC: Latch purged and nullified
706A0001	IGW9LDSC: SQM submit failed
706A0002	IGW9LDSC: Suspend failed
706A0003	IGW9LDSC: Unable to change recovery between ESTAE and FRR
706A0004	IGW9LDSC: Unable to change recovery between ESTAE and FRR
706A0005	IGW9LDSC: Resume failed
706B0001	IGW9LTRC: IGWLSOXL failed
706B0002	IGW9LTRC: IGWLSRXL failed
706B0003	IGW9LTRC: Trace buffer first
706B0004	IGW9LTRC: Trace buffer first
706B0005	IGW9LTRC: Trace buffer first
70EA0001	IGW9LSLE: IGWFTOKM failed
70EA0002	IGW9LSLE: LGA pointer is zero

Chapter 5. Operating in the DFSMStvs transaction processing environment

This topic describes how to set up the storage management subsystem and how to control DFSMStvs processing.

Setting up the storage management subsystem

This topic describes DFSMSdfp storage administration for DFSMStvs. For more information about storage administration, see *z/OS DFSMSdfp Storage Administration*.

Preparing for the storage management subsystem

Before you define and activate an SMS configuration, you need to perform the following preparatory steps:

- Allocate control data sets to contain your SMS configuration and to permit the systems in your complex to communicate with each other.
- Modify SYS1.PARMLIB, which contains three members that direct the initialization and activation of SMS.
- Establish access to the ISMF Primary Option Menu for Storage Administrators (shown in *z/OS DFSMSdfp Storage Administration*).

This topic describes how to perform these preliminary steps so that you can begin defining a base configuration for an SMS configuration.

For more information about planning and implementing SMS, see *z/OS DFSMS Implementing System-Managed Storage*.

For information about planning and preparing for SMS with object support, optical libraries, or tape libraries, see *z/OS DFSMS OAM Planning, Installation, and Storage Administration Guide for Object Support* and *z/OS DFSMS OAM Planning, Installation, and Storage Administration Guide for Tape Libraries*.

Allocating control data sets

Before you can activate an SMS configuration, you need to allocate control data sets used by SMS and define their contents. Control data sets are virtual storage access method (VSAM) linear data sets that contain:

- Base configuration information
- SMS class, aggregate group, optical library, tape library, optical drive, and storage group definitions
- ACS routines.

You can allocate control data sets with access method services or TSO/E commands. You can define and alter the contents of control data sets using ISMF. SMS uses three types of control data sets: a *source control data set* (SCDS), an *active control data set* (ACDS), and a *communications data set* (COMMDS).

Restriction: Do not name any of your SMS control data sets the single word 'ACTIVE'. SMS uses the single word 'ACTIVE' as a reserved word indicating the active configuration residing in the SMS address space. Naming an SMS control data set 'ACTIVE' results in errors.

Source control data set (SCDS)

An SCDS contains an SMS configuration, which defines a storage management policy. You can define any number of SMS configurations each of which has its own SCDS. Then, you select one SMS configuration to be the installation storage management policy and make an active working copy of it in an ACDS.

Active control data set (ACDS)

When you activate an SCDS, its contents are copied to an ACDS. The current ACDS contains a copy of the most recently activated configuration. All systems in an SMS complex use this configuration to manage storage. You can define any number of SCDSs, but only one can be put in the ACDS. *z/OS MVS Initialization and Tuning Guide* explains how to specify the ACDS. You can define more than one IGDSMS*xx* member, each specifying a different ACDS, but you can use only one ACDS at a time.

Restriction: You cannot define or alter an ACDS. This also means that you cannot use an ACDS as an SCDS if the SCDS is lost.

You can modify the SCDS from which your current storage management policy was activated without disrupting operations, because SMS manages storage with a copy of the SMS configuration (an ACDS) rather than with the original (an SCDS). While SMS manages storage using an ACDS, you can:

- Create a backup copy of the SCDS
- Build a new SCDS
- Update the SCDS from which the ACDS was activated
- Modify any SCDS

Figure 9 on page 281 shows the relationship among SCDSs and ACDSs in an installation.

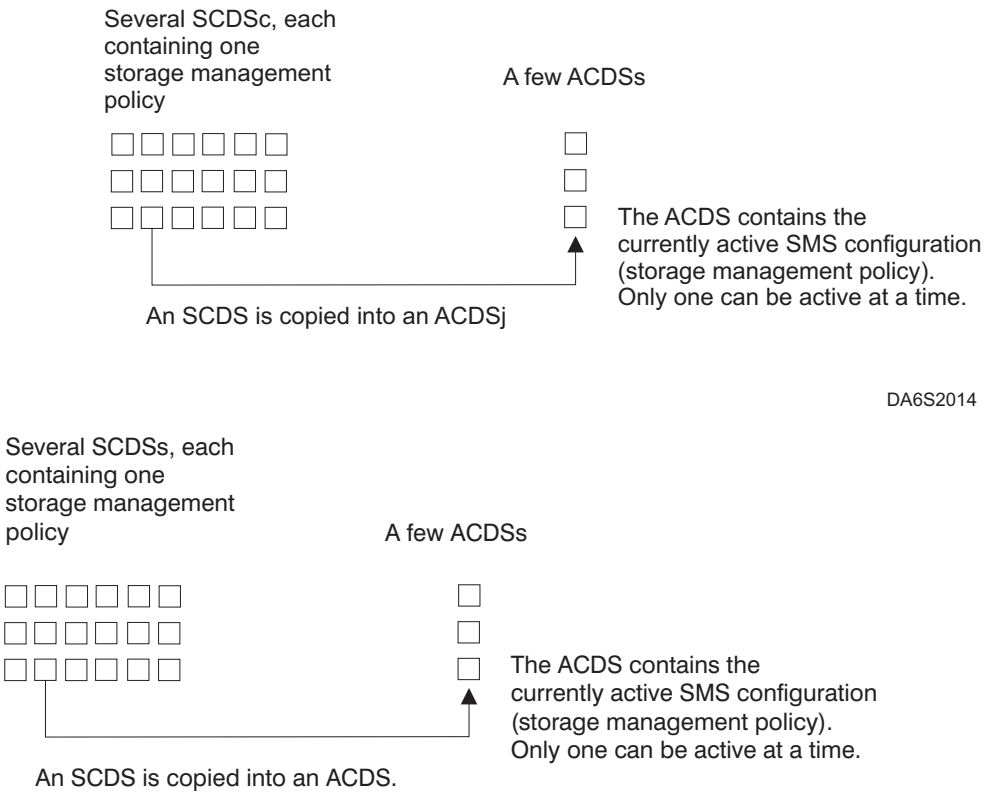


Figure 9. Relationships among SCDSs and ACDSs in an installation

Communications data set (COMMDS)

The COMMDS serves as the primary means of SMS communication among systems in the SMS complex. The active systems in an SMS complex access the COMMDS for current SMS complex information.

The COMMDS contains the name of the ACDS containing the currently active storage management policy, the current utilization statistics for each system-managed volume, and other system information. You can define any number of COMMDSs, but only one can be active in an SMS complex.

Calculating the size of storage and active control data sets: Before you allocate control data sets, you need to estimate their size. When calculating the size of either an ACDS or an SCDS, you have to account for system and base configuration information, SMS class, aggregate group, storage group, optical library and drive, and tape library definitions.

The following formula can help you determine the size of a source or active control data set:

```
size (bytes) = 150000 +
               (14000 * SG) +
               (2312 * (MC + SC + DC + AG + CS)) +
               (12000 * (DRV + LIB + VOL))
```

where:

150000 bytes
represent fixed data fields. For example, suppose you have the following configuration:

- Five storage groups
- Four management classes
- Two storage classes
- Five data classes
- Two aggregate groups
- 6 optical libraries
- 24 optical drives
- 40 DASD volumes

With this configuration, the equation yields a value of 1088600 bytes for control data set allocation.

SG is the estimated number of storage groups

MC is the estimated number of management classes

SC is the estimated number of storage classes

DC is the estimated number of data classes

AG is the estimated number of aggregate groups

CS is the estimated number of cache sets in the base configuration (for VSAM record-level sharing (RLS) only)

DRV is the estimated number of optical drives in the SMS complex to be managed by SMS

LIB is the estimated number of optical and tape libraries in the SMS complex to be managed by SMS

VOL is the estimated number of DASD volumes in the SMS complex to be managed by SMS

If you are running SMS in a Parallel Sysplex environment with different releases of DFSMS, you must allocate your control data sets using calculations that are based on the highest DFSMS level. If you do not, your control data sets might be too small, because the storage requirements for classes and groups might be different between releases or new classes, groups, or other items might be added.

On an IBM 3380 or 9345 DASD, each track can contain 40 KB (40960 bytes). On an IBM 3390 DASD, each track can contain 48 KB (49156 bytes).

If your SCDS or ACDS is not large enough, you might receive SMS reason code 6068 when attempting to save its contents. When reason code 6068 occurs, allocate a new, larger control data set and copy your existing SCDS into your new SCDS, or your existing ACDS into your new ACDS. You can then delete the old SCDS or ACDS and use the new one.

Allocating a new SCDS or ACDS resolves the problem only when reason code 6068 is caused by a data set size problem. Because this reason code is returned when a system service called by data-in-virtual (DIV) fails, the error might have other causes. Messages returned to you or the system console can help determine the cause of the failure.

Restriction: DIV has a current size limit of 4 GB. Make sure you do not exceed this limit.

Calculating the size of a COMMDS: When you calculate the size of a COMMDS, you have to account for both system and volume information. With SMS 32-name support, the amount of space required for a COMMDS increased. A previously allocated COMMDS might have insufficient space to support the changes. You might need to allocate a new COMMDS prior to activating SMS on a current-level system, and you should always review the COMMDS size when migrating from prior DFSMS releases. The following formula can help you determine the size of a COMMDS: (VOL = Estimated number of DASD volumes in the SMS complex to be managed by SMS):

COMMDS size (bytes) =
8192 + (588 * VOL)

For example, if you have 40 DASD volumes in the SMS complex, you need to allocate 31712 bytes for the COMMDS.

Selecting volumes for control data sets: SMS control data sets can be either SMS-managed or non-SMS-managed. Initially, you should ensure that your control data sets have a volume count of 1. The volume count can be explicitly specified, implied by the number of volume serials provided, or derived from the data class assigned to the data set (for more information, see *z/OS DFSMSdfp Storage Administration*). If you give an SMS control data set a volume count that is greater than the number of volumes on which the data set actually resides, you might receive messages IEF244I and IEF489I when you attempt to activate it.

If you have a multivolume SCDS that you are activating into a single volume ACDS, you might receive an error because the ACDS is not large enough and volumes cannot be dynamically added to it. To bypass this problem, you need to create a new multivolume ACDS and then activate the ACDS and SCDS simultaneously using the SETSMS command. See "Changing storage management subsystem parameters" on page 287 for further information on this command.

If your SMS complex includes more than 16 systems, be sure that the ACDS and COMMDS are accessible to every system in the complex. Define your control data sets on volumes which are capable of being attached to more than 16 systems, such as IBM RAMAC™ Virtual Array volumes.

Allocating an SCDS: The following access method services job allocates a 6-track SCDS.

```
//STEP  EXEC PGM=IDCAMS
//SYSUDUMP DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
  DEFINE CLUSTER(NAME(SMS.SCDS1.SCDS) LINEAR VOL(SMSV01) -
  TRK(6 6) SHAREOPTIONS(2,3)) -
  DATA(NAME(SMS.SCDS1.SCDS.DATA))
/*
```

This job creates a VSAM linear data set named SMS.SCDS1.SCDS. You can combine the DEFINE commands for all your allocations into the same job step, but this example shows only one for purposes of illustration. After allocating an SCDS, you define its contents through ISMF dialogs.

You should allocate an SCDS on a device shared by all systems in the SMS complex. If you allocate an SCDS on a device that is not shared by all the systems, then you can activate the SCDS only from systems that have access to it.

You should specify the REUSE option when you define an SCDS to avoid running into space problems (SMS reason code 6068) as result of subsequent SCDS updates, or IMPORT/EXPORT functions.

For information on using access method services commands, see *z/OS DFSMS Access Method Services Commands*.

Allocating an ACDS: The following access method services job allocates a 6-track ACDS.

```
//STEP  EXEC PGM=IDCAMS
//SYSUDUMP DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
  DEFINE CLUSTER(NAME(SMS.ACDS1.ACDS) LINEAR VOL(SMSV02) -
  TRK(6 6) SHAREOPTIONS(3,3)) -
  DATA(NAME(SMS.ACDS1.ACDS.DATA))
/*
```

This job creates a VSAM linear data set named SMS.ACDS1.ACDS.

An ACDS must reside on a shared volume, accessible from all systems in the SMS complex. To ease recovery in case of failure, the ACDS should reside on a different volume than the COMMDS. Also, you should allocate a spare ACDS on a different shared volume. *z/OS DFSMSdfp Storage Administration* provides more information about the backup and recovery of control data sets.

You create the contents of an ACDS by activating a valid SCDS. The distinction between a valid SCDS and one that is not valid is described in *z/OS DFSMSdfp Storage Administration*. The control data set (ACDS or COMMDS) must reside on a volume that is not reserved by other systems for a long period of time because the control data set (ACDS or COMMDS) must be available to access for SMS processing to continue.

You should specify the REUSE option when you define an ACDS to avoid running into space problems (SMS reason code 6068) as result of subsequent ACDS updates, or IMPORT/EXPORT functions.

Allocating a COMMDS: The following access method services job allocates a 1-track COMMDS;

```
//STEP  EXEC PGM=IDCAMS
//SYSUDUMP DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
  DEFINE CLUSTER(NAME(SMS.COMMDS1.COMMDS) LINEAR VOL(SMSVOL) -
  TRK(1 1) SHAREOPTIONS(3,3)) -
  DATA(NAME(SMS.COMMDS1.COMMDS.DATA))
/*
```

This job creates a VSAM linear data set named SMS.COMMDS1.COMMDS.

The COMMDS must reside on a shared volume accessible from all systems in the SMS complex. To ease recovery in case of failure, the COMMDS should reside on a different volume than the ACDS. Also, you should allocate a spare COMMDS on a different shared volume. *z/OS DFSMSdfp Storage Administration* provides additional information on the backup and recovery of control data sets. The control data set (ACDS or COMMDS) must reside on a volume that is not reserved by other

systems for a long period of time because the control data set (ACDS or COMMDS) must be available to access for SMS processing to continue.

Recommendation: Use SHAREOPTIONS(3,3) when allocating an ACDS. This allows full authority to read from and write to an ACDS from any system. The ACDS and COMMDS must be accessed from all systems in the complex simultaneously.

Modifying the SYS1.PARMLIB data set

The IGDSMS*xx*, IEASYS*yy*, and IEFSSN*xx* members of SYS1.PARMLIB direct the initialization and activation of SMS. IGDSMS*xx* provides initialization parameters to SMS. The SMS=*xx* parameter of IEASYS*yy* indicates the name of the SYS1.PARMLIB member IGDSMS*xx* that is used for initialization. For example, if SMS=01 in IEASYS*yy*, then the IGDSMS01 member of SYS1.PARMLIB is used during initialization. The SMS entry in IEFSSN*xx* identifies SMS to z/OS.

For information about how to create an IGDSMS*xx* member in SYS1.PARMLIB and how to define SMS to z/OS through IEFSSN*xx*, see *z/OS MVS Initialization and Tuning Guide*. For information about the syntax of the IGDSMS*xx*, IEASYS*yy*, and IEFSSN*xx* parmlib members, see *z/OS MVS Initialization and Tuning Reference*.

Starting the SMS address space: When you have completed preparations and are ready to start SMS, use the T SMS=*xx* command, in which *xx* identifies IGDSMS*xx* as the SMS initialization member. The T SMS=*xx* command is an abbreviation for the SET SMS=*xx* command, discussed in "Step 2: Prepare one system" on page 286. To eliminate confusion with the SETSMS operator command, the abbreviated T SMS=*xx* form of the SET SMS=*xx* command is used throughout the remainder of this topic.

When you have sufficiently tested your operations and are ready to have SMS automatically started at future IPLs, add the IGDSSIIN module name to the SMS entry.

Here are some examples of the SMS record:

- The following SMS record defines SMS to z/OS without starting SMS at IPLs:

  ```
  SUBSYS SUBNAME(SMS)
  ```

- The following SMS record defines SMS to z/OS and starts SMS at future IPLs:

  ```
  SUBSYS SUBNAME(SMS) INITRTN(IGDSSIIN)
  ```

  The system uses the default values of **ID** and **PROMPT**. IGDSMS00 specifies initialization information, and the operator has no control over the rest of SMS initialization.

- The following SMS record allows the operator to modify SMS initialization:

  ```
  SUBSYS SUBNAME(SMS) INITRTN(IGDSSIIN)
         INITPARM(',PROMPT=YES')
  ```

  The system uses the default value of **ID**, which identifies IGDSMS00 as containing initialization information. The **PROMPT** parameter requests that SMS display IGDSMS00, so that the operator can modify the parameters in IGDSMS00.

- The following SMS record initializes SMS using IGDSMS01:

  ```
  SUBSYS SUBNAME(SMS) INITRTN(IGDSSIIN)
         INITPARM('ID=01,PROMPT=DISPLAY')
  ```

The **PROMPT** parameter requests that the contents of IGDSMS01 be displayed, but the operator cannot modify them.

Accessing the storage administrator Primary Option Menu

The first time you select ISMF, you get the ISMF Primary Option Menu for *end users*. To get the ISMF Primary Option Menu for *storage administrators* (which is shown in *z/OS DFSMSdfp Storage Administration*), select option 0, `ISMF PROFILE`. Within the ISMF Profile Option Menu, select option 0, `USER MODE`, and press ENTER. You get the User Mode Entry panel, where you indicate that you want the storage administrator Primary Option Menu for all future ISMF sessions. To do this, select option 2 on the User Mode Entry panel. After changing the user mode, you must exit ISMF and then return to it to view the Primary Option Menu for Storage Administrators.

z/OS DFSMSdfp Storage Administration explains how to prevent end users from gaining access to the storage administrator Primary Option Menu through the ISMF PROFILE option. The reason for restricting access to the Primary Option Menu for Storage Administrators is to prevent unauthorized users from performing storage administrator tasks.

Activating storage management subsystem configurations

You can activate an SMS configuration manually, or automatically at IPL. This topic shows you how to perform the initial activation of an SMS configuration using a four-step manual approach. It also explains how you can activate an SMS configuration automatically at IPLs. In addition, it explains how you can change individual SMS parameters with the SETSMS operator command.

Prerequisite: When you activate an SMS configuration, ensure that all of the DASD volumes that belong to the configuration are initialized as SMS volumes. Otherwise, attempted allocations to an improperly initialized volume will fail. However, initialization for tape volumes is no different for SMS-managed and non-SMS-managed volumes.

Manually activating a storage management subsystem configuration

IGDSSIIN is the subsystem initialization routine module for SMS. By omitting it from the SMS entry of IEFSSN*xx* for each system in the SMS complex, you can manually control the activation of an SMS configuration. Refer to "Preparing for the storage management subsystem" on page 279.

Step 1: IPL each system in the SMS complex: After defining SMS as a subsystem to z/OS, IPL each system in the SMS complex. The presence of the SMS entry in IEFSSN*xx* tells z/OS to recognize SMS as a valid subsystem within each system. The absence of the IGDSSIIN module name in IEFSSN*xx* tells the system that you want to start SMS manually.

Step 2: Prepare one system: From one system in the SMS complex, issue the T SMS=*xx* command, in which *xx* identifies IGDSMS*xx* as the SMS initialization control member of SYS1.PARMLIB. SMS uses the ACDS and COMMDS identified in IGDSMS*xx* to manage storage. Because the initial ACDS and COMMDS are empty, the system is activated with a *null configuration*. Keep in mind that a null configuration is intended only as a migration path.

Requirement: All systems in the SMS complex must be running in the same mode. When an SMS control data set that supports only eight names is accessed for update on a system running in 32-name mode, you must convert the data set to a

new, incompatible format in order to support 32 names. Confirm this conversion using the operator console or the ISMF. This conversion is permanent, so make copies of your control data sets before the system mode is converted.

Step 3: Activate the configuration from one system: The configuration is only activated once for the SMS complex. It is not necessary to activate a configuration from every system in the SMS complex. After activating SMS with a null configuration, activate an SMS configuration contained in a valid SCDS on the same system. You can use either the ISMF ACTIVATE command or the SETSMS operator command. Both procedures copy the contents of the SCDS to the ACDS specified in IGDSMS*xx*.

When an SMS control data set that supports only eight names is accessed for update on a system running in 32-name mode, you must convert the data set to a new, incompatible format in order to support 32 names. Confirm this conversion, using the operator console or ISMF. This conversion is permanent, so you should make copies of your control data sets before the system mode is converted.

Activating with the ISMF ACTIVATE command: On the Control Data Application Selection panel shown in *z/OS DFSMSdfp Storage Administration*, specify the name of an SCDS and issue the ACTIVATE command from the command line. A *Write to Programmer* message indicates if the activation is successful, provided you have WTPMSG in the TSO/E PROFILE.

Activating with the SETSMS operator command: From the operator console, issue the command:

```
SETSMS SCDS(dsname)
```

In the command, `dsname` identifies the name of the SCDS to be activated. The command syntax is as follows:

```
SETSMS SCDS(SMS.SCDS1.SCDS)
```

Step 4: Activate SMS on the other systems: For the other systems in the SMS complex, use the T SMS=*xx* command to start SMS on those systems, using the SMS configuration identified in Step Three. In each system, IGDSMS*xx* specifies the name of the ACDS containing the SMS configuration. All of the IGDSMS*xx* members must point to the same ACDS. Because the ACDS is no longer empty, the systems use it (and the COMMDS) to manage storage.

Automatically activating a storage management subsystem configuration

For each system in the SMS complex, update the SMS entry in IEFSSN*xx* to include the IGDSSIIN module name.

Make certain that each ID field identifies the IGDSMS*xx* member containing the name of the last used ACDS. All of the IGDSMS*xx* members must point to the same ACDS. At future IPLs, the SMS configuration contained in the ACDS is activated by all systems in the SMS complex. For example, the following command indicates that the ACDS specified in IGDSMS02 contains the SMS configuration to be activated at future IPLs:

```
SMS,IGDSSIIN,'ID=02,PROMPT=DISPLAY'
```

Changing storage management subsystem parameters

After you activate an SMS configuration, you can use the `SETSMS` operator command to change SMS parameters.

Restriction: Some of the parameters contained in the IGDSMS*xx* parmlib member cannot be changed using the SETSMS operator command. The section "Parameters of the SETSMS operator command" lists only those parameters that can be changed using the SETSMS operator command. For a complete list of the IGDSMS*xx* parmlib member parameters, see *z/OS MVS Initialization and Tuning Guide*.

Parameters of the SETSMS operator command: You can change SMS parameters by using the `SETSMS` operator command. For descriptions of the parameters, see *z/OS MVS System Commands* and *z/OS MVS Initialization and Tuning Guide*.

Considerations for changing storage management subsystem configurations: When activating a new SMS configuration, you have two options for keeping the currently active SMS configuration information:

- Keep, but never modify, the original SCDS from which the current SMS configuration was activated.

  If you choose this method, you need to maintain a log of all status changes, such as VARY storage group commands, that you make to the currently active SMS configuration. If in the future you activate a different SMS configuration but then decide you want to fall back to your original, you can reactivate the SCDS. You lose all the status changes you have made since activating the SCDS and must reenter them, but you return to the original SMS configuration.
- Save the current active SMS configuration using the SETSMS operator command:

  ```
  SETSMS SAVEACDS(ACDS.FALLBACK)
  ```

  This is the better alternative for keeping the SMS configuration information..

  ACDS.FALLBACK must be an existing, already allocated data set. By using this command, you not only save the current storage management policy in ACDS.FALLBACK, but you also save the status changes you have made since the original SCDS was activated. You can then activate the new SMS configuration. If in the future you decide you want to fall back to the original SMS configuration, you can use the SETSMS operator command to reactivate it:

  ```
  SETSMS ACDS(ACDS.FALLBACK)
  ```

  This alternative is also useful if you have altered the SCDS that you originally activated.

Restriction: Any SMS status that is changed using the SETSMS command is overridden by the MVS status if a new configuration is activated. For more information on the SETSMS operator command, see *z/OS MVS System Commands*.

B' vnn8vnvvv.l'**Requirement:** All systems in the SMS complex must be running in the same mode.

When an SMS control data set that supports only eight names is accessed for update on a system running in 32-name mode, you must convert the data set to a new, incompatible format in order to support 32 names. Confirm this conversion using the operator console or the ISMF. This conversion is permanent, so you should make copies of your control data sets before the system mode is converted.

OAM considerations for changing SCDSs: SMS notifies OAM when a new SCDS has been activated. OAM takes action depending on the RESTART parameter that is specified on the OAM address space. If RESTART=YES is specified, or defaulted, the OAM address space automatically restarts, rebuilding its configuration to match the newly activated SCDS. If RESTART=NO is specified, OAM does not automatically restart, but issues a message acknowledging that an activation has

taken place. In this case, you must determine if an OAM restart is necessary. If a restart is necessary, use the MODIFY OAM,RESTART command.

When RESTART=YES is specified, the length of the delay between the SCDS activation and the OAM restart depends on the value specified in the INTERVAL keyword in the active IGDSMS*xx* parmlib member. During this reinitialization, all optical libraries and drives defined to the new SCDS are reset to the initial status values specified in the SCDS. After the OAM restart completes, display all optical libraries and drives, and tape libraries, and then set them to the desired online or offline status before the reinitialization occurred.

For more information, see *z/OS DFSMS OAM Planning, Installation, and Storage Administration Guide for Object Support*.

Displaying storage management subsystem information

You can use the DISPLAY SMS MVS command to display information about the storage management subsystem, as follows:

- Active SMS configuration
- SMSVSAM status of sharing control data sets
- SMSVSAM server name
- Coupling facility cache and lock structures

For a description of the DISPLAY SMS command, see *z/OS MVS System Commands*.

Changing storage management subsystem parameters

You can use the SET SMS or SETSMS MVS command to change SMS parameters. Use the SET SMS command to change parameters before you bring up SMS. Use the SETSMS command when SMS is active (running) to change a subset of SMS parameters from the console without changing the active IGDSMS*xx* member of SYS1.PARMLIB.

For descriptions of SET SMS and SETSMS, see *z/OS MVS System Commands*.

Controlling DVSMStvs processing

While DFSMStvs is processing, you can monitor the performance of applications programs and maintain data integrity during backup-while-open (BWO) processing.

Monitoring application programs that use DFSMStvs

For information about monitoring application programs that use DFSMStvs and tuning performance, see *z/OS DFSMStvs Planning and Operating Guide*.

Changing DFSMStvs status

You can use the VARY SMS MVS command to change the status for DFSMStvs in these ways:

- Change the state of a DFSMStvs instance or change the state of all DFSMStvs instances in the sysplex
- Change the state of a log stream to which DFSMStvs has access
- Change the state of a data set for VSAM record-level sharing (RLS) and DFSMStvs access
- Start or stop peer recovery processing for a DFSMStvs instance

Restriction: You cannot use the VARY SMS command to change the state of a DFSMStvs instance while it is initializing. Any attempt to do so is suspended until the initialization completes.

The possible states of a DFSMStvs instance are as follows:

ENABLE
Enables DFSMStvs to begin accepting new units of recovery for processing.

DISABLE
Prevents DFSMStvs from processing new work requests. DFSMStvs does not process new work requests from units of recovery that are currently in progress.

QUIESCE
Prevents DFSMStvs from accepting any new units of recovery for processing. DFSMStvs completes the processing of any units of recovery in progress.

The possible states of a data set follow:

ENABLE
Unquiesces a data set for VSAM RLS and DFSMStvs access.

QUIESCE
Quiesces a data set for VSAM RLS and DFSMStvs access.

For a description of the VARY SMS command, see *z/OS MVS System Commands*.

For information about the effects of DFSMStvs, log stream and data set states on DFSMStvs processing, see *z/OS DFSMStvs Planning and Operating Guide*.

Maintaining data integrity during backup-while-open processing

This topic describes DFSMSdss storage administration for DFSMStvs. For more information, see *z/OS DFSMSdss Storage Administration*.

Data integrity—serialization

DFSMSdss uses **volume serialization** and **data set serialization** functions to ensure that data sets are not modified during the processing of DFSMSdss commands. Volume serialization is accomplished by using the RESERVE macro. Data set serialization is accomplished by using the ENQ macro and the DFSMSdss DYNALLOC function. In the case of shared DASD, volume serialization ensures that data sets are not being added, deleted, renamed, or extended from either the same processor or other processors.

You can control volume serialization with the enqueue exit routine. (Refer to *z/OS DFSMS Installation Exits* for additional information regarding installation enqueue exit routines.) This allows other programs to access volumes while DFSMSdss is running. Volume serialization, however, cannot be overridden.

DFSMSdss supports data sets that are always open, or open for long periods of time, with **backup-while-open serialization**.

DFSMSdss supports backup-while-open processing for DFSMStvs and for two types of database applications: Customer Information Control System (CICS) and information management system (IMS).

CICS support includes both RLS CICS and non-RLS CICS backup-while-open, as "Backup-while-open data sets (CICS and DFSMStvs)" describes.

DFSMStvs supports backup-while-open with or without CICS, as "Backup-while-open data sets (CICS and DFSMStvs)" describes.

DFSMSdss also supports backup-while-open serialization for IMS data sets:

- Indexed VSAM data sets, such as key-sequenced data sets (KSDS)
- Nonindexed VSAM data sets, such as entry-sequenced data sets (ESDS)
- Non-VSAM data sets, such as OSAM

"Backup-while-open data sets (CICS and DFSMStvs)" describes DFSMStvs backup-while-open support.

Backup-while-open data sets (CICS and DFSMStvs)

DFSMSdss supports backup-while-open serialization, which can perform backups of data sets that are open for update for long periods of time. It can also perform a logical data set dump of these data sets even if another application has them serialized. Backup-while-open is a better method than using SHARE or TOLERATE(ENQFAILURE) for dumping Customer Information Control System (CICS) VSAM file-control data sets that are in-use and open for update. When you dump data sets that are designated by CICS as eligible for backup-while-open processing, data integrity is maintained through serialization interactions between CICS (database control program), CICSVR (CICS VSAM Recovery), VSAM record management, DFSMSdfp, and DFSMSdss. When you dump data sets that are designated by DFSMStvs as eligible for backup-while-open processing, data integrity is maintained through serialization interactions between DFSMStvs, VSAM record management, DFSMSdfp, and DFSMSdss.

Although the BWO(TYPECICS) parameter applies to both CICS and DFSMStvs, DFSMStvs enables you to back up a data sets while they are open whether or not you are running CICS.

Figure 10 on page 292 shows the backup-while-open serialization for dumping CICS data sets that are open for update. Backup-while-open processing also ensures that any update activity that may invalidate the dump is detected. Simultaneous recovery or deletion of the data set while it is being dumped is also prevented.

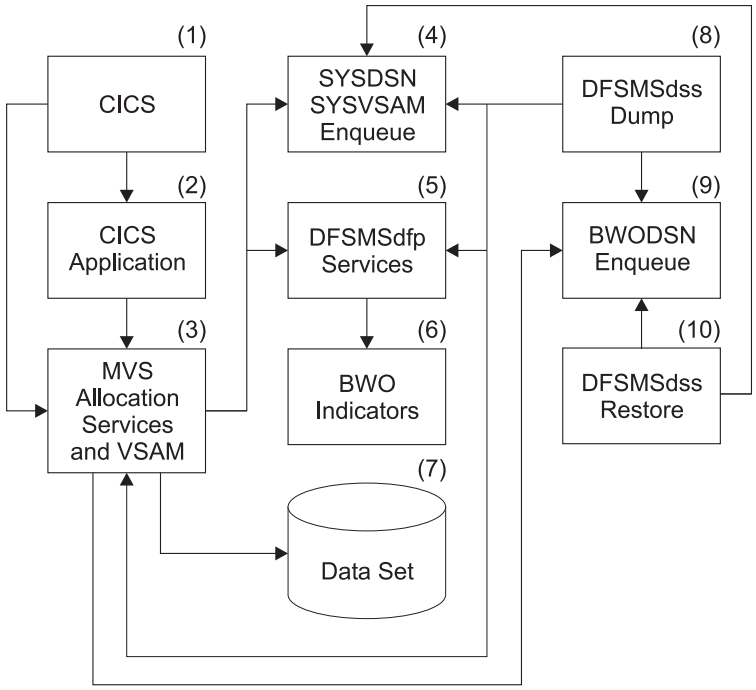


Figure 10. Block diagram for backup-while-open serialization

In Figure 10, a VSAM file-control data set (7) is allocated for CICS (1) through MVS allocation services (3) using JCL or dynamic allocation methods. This results in serialization through an enqueue on the name of the data set and the resource name SYSDSN (4). When the VSAM data set is opened (3), another level of serialization occurs through an enqueue on the names of the components of the VSAM data set and the resource name SYSVSAM (4). For eligible data sets, CICS uses DFSMSdfp (5) to set a status in the backup-while-open indicators (6) in the catalog entry for the data set.

For a dump operation, DFSMSdss (8) attempts to acquire the SYSDSN, SYSVSAM, and backup-while-open (9) enqueues for the data set. When the enqueue on the cluster name of the data set and the resource name of BWODSN is acquired, but not the enqueues for both SYSDSN and SYSVSAM, DFSMSdss uses DFSMSdfp to get the backup-while-open indicators and starts to dump the open data set if it were eligible for backup-while-open processing.

The backup-while-open enqueue is used to prevent more than one DFSMSdss operation, such as a simultaneous dump and restore (10), and to prevent the data set from being deleted while it is being dumped by DFSMSdss.

While the data set is being dumped, a database application program might update the data set in a manner that invalidates the data set. For example, a control-interval or control-area split might occur. If this happens, VSAM record management uses DFSMSdfp to change the backup-while-open status.

When the backup of the open data set is completed, DFSMSdss obtains the current backup-while-open indicators. If the indicators are different from when the dump was started, DFSMSdss invalidates the dump of the data set. When concurrent copy is used, updates made while the data set is being dumped do not cause the dump to be invalidated.

Backup-while-open status definition: DFSMSdss interprets backup-while-open status numbers as follows:

Status	Meaning to DFSMSdss
000	Normal serialization and processing techniques are used to dump the data set.
001	CICSVR forward-recovers the data set.
010	A control-interval split, control-area split, or extend of the data set was either interrupted before the dump started or is currently in process.
011	A control-interval split, control-area split, or an extend of the data set completed successfully. CICS or DFSMStvs closed the data set and there is no mismatch between the base cluster and any alternate index.
100	The data set might be dumped while it is open for update.
101	The data set was restored and is down-level. CICSVR or DFSMStvs must process the data set before a database application uses it.
110	The data set was updated and the completion of a split or extend invalidates any dump that was in progress. When concurrent copy is used, the chances of an in-progress dump being invalidated by a split or extend is significantly reduced.
111	The data set is in an indeterminate state.

Backup-while-open processing: DFSMSdss actions that result from dumping a data set are based on the following conditions:

- Success or failure to acquire an exclusive enqueue for the resource names of SYSDSN, SYSVSAM, and BWODSN
- Backup-while-open status indicators *before* the dump
- Backup-while-open status indicators after the dump

When DFSMSdss acquires all of the exclusive enqueues: The data set is not open for update, even though it has a nonzero backup-while-open status. The particular processing action is a direct result of the initial backup-while-open status, as follows:

- Status is 000; the data set is dumped. When it is restored, the backup-while-open status is restored as 000.
- Status is 001. The data set is dumped and the backup-while-open status is preserved and restored when the data set is subsequently restored. This data set is in an indeterminate state because of an incomplete forward recovery by CICSVR, which cannot forward-recover the restored data set.

  DFSMSdss lets you dump and restore this data set for nonstandard recovery purposes. After restoring the data set and correcting all errors in the data set, reset the BWO status to 000 by using CICS-provided methods. The preferred action is for you to restore an earlier dump of the data set and use CICSVR to do forward recovery processing. This maintains the integrity of the data.
- Status is 010. DFSMSdss did not dump the data set. Take corrective action before using the data set by performing the following actions:
  - Determine if alternate indexes are present for the sphere. If they are, do either of the following tasks:
    - Rebuild the alternate indexes from the base cluster, and reset the BWO status to 000 by using the CICS methods.

    - Restore an earlier dump of the sphere, and use CICSVR to do forward recovery processing.
  – If there are no alternate indexes, the data set is usable as it is. Reset the BWO status to 000 by using CICS methods.
- Status is 101. The data set is dumped and the backup-while-open status is preserved and restored when the data set is subsequently restored. This lets you process the data set with CICSVR before a database application uses it.
- Status is 011, 100, 110, 111. The backup-while-open status is altered to 000 before the data set is dumped. When it is restored, the backup-while-open status is restored as 000.

When DFSMSdss acquires an exclusive enqueue on BWODSN (but not SYSDSN and SYSVSAM): Another program is using the data set or the data set is open. The particular processing action is a direct result of the initial backup-while-open status, as follows:

- Status is 000. The data set is not eligible for backup-while-open processing. The data set is not dumped unless SHARE or TOLERATE(ENQFAILURE) is specified and the necessary conditions for those keywords are met.
- Status is 001, 010, 011, 101, or 111; the data set is not dumped.
- Status is 110; the backup-while-open status is altered to 100 and the data set is dumped (see the following text).
- Status is 100. The data set is dumped, even though it is already in use (including being open for update by another program). When the data set is restored, the backup-while-open status is set to 101. This ensures that CICSVR or DFSMStvs can process the data set prior to its use by a database application. The dump is invalidated if the backup-while-open indicators change while the data set is being dumped. The chances of this invalidation occurring reduce significantly when you use concurrent copy.

Backup-while-open and concurrent copy: Concurrent copy improves backup-while-open processing by significantly reducing the chances of the invalidation of a backup-while-open dump because of updates to the data set. To use concurrent copy, specify the CONCURRENT keyword when you dump backup-while-open data sets. The following list compares the various kinds of dumps that you can ask for:

- **Normal dump.** Use of the data set must be quiesced. DFSMSdss obtains serialization, dumps the data set, and then releases the serialization. The data set cannot be used for the entire time.
- **Concurrent copy dump.** Use of the data set must be quiesced. DFSMSdss obtains serialization, performs concurrent copy initialization, releases serialization, and then dumps the data set. DFSMSdss completes concurrent copy initialization within a very short time (compared to the actual time to dump the data set). DFSMSdss can use the data set as soon as the concurrent copy initialization is complete.
- **Backup-while-open dump.** Use of the data set does not need to be quiesced. DFSMSdss dumps the data set without obtaining serialization. The data set can remain in use for the entire time, but update activity can invalidate the dump at any time during the dump.
- **Backup-while-open dump using concurrent copy.** Use of the data set does not need to be quiesced. DFSMSdss does not obtain serialization. DFSMSdss performs the concurrent copy initialization. DFSMSdss completes concurrent copy initialization within a very short time (compared to the actual time to dump the data set). The data set can remain in use for the entire time. Like the

backup-while-open dump, update activity during the dump can invalidate the dump; however, only update activity that occurs *before* DFSMSdss performs the concurrent copy initialization can invalidate the dump. The chances of the update activity invalidating the dump are significantly reduced because the concurrent copy initialization completes very quickly.

TOLERATE (ENQFAILURE) and SHARE considerations: Using TOLERATE(ENQFAILURE) modifies the processing and serialization for backup-while-open data sets. The data sets are dumped when the backup-while-open status is 100, even though none of the enqueues are successfully acquired.

To maintain data integrity and data security, do not specify either SHARE or TOLERATE(ENQFAILURE) when you dump backup-while-open data sets.

An exclusive enqueue on the BWODSN resource name for the data set is required for DFSMSdss to alter the backup-while-open status. DFSMSdss attempts an exclusive enqueue only on the BWODSN resource name. Unless the backup-while-open status is already 100, the dump fails, even though you specified SHARE or TOLERATE(ENQFAILURE).

Recovery data: CICS maintains recovery data in the form of a date and time-stamp in the catalog entry for backup-while-open data sets.

DFSMStvs maintains recovery data in forward recovery log records. DFSMSdss does not use or process this recovery data. The data is dumped and restored to preserve the information for CICSVR or possibly for another forward recovery utility. The recovery data is also printed in selected messages to assist you in your recovery efforts.

Chapter 6. Diagnosing DFSMStvs problems

This topic describes DFSMSdfp diagnosis information for DFSMStvs.

Any problem from a module that has a name beginning with "IGW8" or any reason code that starts with "6F*xxxxxx*" is from DFSMStvs recoverable file services, and any module that has a name beginning with "IGW9" or any reason code that starts with "70*xxxxxx*" is from the DFSMStvs logger. The DFSMStvs log records include the subsystem ID, like IGWTVS01 and IGWTVS02. Also, any IGW8*xx* and IGW10*xxx* messages are from DFSMStvs.

Dumping all the servers in case of a hang applies to DFSMStvs as well as to RLS. Recoverable resource services (RRS) or the system logger can also cause apparent hangs in DFSMStvs or batch jobs that use it, so you might want to dump RRS and the system logger too if you think you have a hang related to DFSMStvs. They might not all fit into one dump, and partial dumps are generally useless, so this might require more than one dump. If you think you have a hung batch job, be sure to dump both the SMSVSAM server and the hung job.

Incorrect output keyword

Use this topic when a program or the system does not produce the expected output.

Incorrect output failures can be identified by the following:

- Expected output is missing.
- Output is different than expected.
- Output should not have been generated.
- System indicates damage to the VTOC or VTOC index.
- ISMF panel information or flow is erroneous.

Incorrect output can be the result of a previous failure and can often be difficult to analyze because the component affected might not be the one that caused the problem. Review previous messages, abends, console logs, or other system responses. They could indicate the source of the failure.

Procedure

1. If a message accompanied the failure, append the message identifier to the prefix MSG according to the procedures starting on “Message keyword” on page 304 and add this keyword to the keyword string. If the system did not issue a message, try to identify any failure-related control blocks, user areas, or data records and record them on the Keyword Worksheet, in *z/OS DFSMSdfp Diagnosis*, as modifier keywords.

   Specify the incorrect output keyword as INCORROUT.
2. If the system indicates damage to the VTOC or VTOC index, then DADSM or CVAF normally issues an error message. In this case, examine the Standard Modifier Keyword list and go to the DADSM/CVAF-related Incorrect Output Failure Modifier Keywords procedure, both in *z/OS DFSMSdfp Diagnosis*, to identify applicable symptom keywords. If VTOC problems are not indicated, continue with this procedure.

3. Accumulate as much of the following information as possible. It can help you isolate or resolve your problem, and the IBM Support Center will request it if trap or trace information is needed.
   - When was the problem first noticed?
   - How was the problem identified (good output versus bad output)?
   - Were any system changes or maintenance recently applied? For example, a new device, software product, APAR, or PTF?
   - Does the problem occur with a specific data set, device, time of day, and so forth?
   - Does the problem occur in batch or TSO/E mode?
   - Is the problem solid or intermittent?
   - Can the problem be re-created?
4. Select the procedure for the failure-related component from the following table.

This table lists the procedure for the failure-related components.

Subcomponent	Procedure
Catalog Management	See *z/OS DFSMSdfp Diagnosis*.
Device Console Services	See *z/OS DFSMSdfp Diagnosis*.
ISMF	See *z/OS DFSMSdfp Diagnosis*.
Media Manager	See *z/OS DFSMSdfp Diagnosis*.
O/C/EOV (Common)	See *z/OS DFSMSdfp Diagnosis*.
Storage management subsystem (SMS)	See *z/OS DFSMSdfp Diagnosis*.
VSAM Block Processor or Record Management	See *z/OS DFSMSdfp Diagnosis*.
VSAM RLS	See "VSAM RLS—incorrect output keyword."
DFSMStvs	See "DFSMStvs—incorrect output keyword" on page 299.
All Other DFSMSdfp Subcomponents	See *z/OS DFSMSdfp Diagnosis*.

VSAM RLS—incorrect output keyword

Use this topic to gather detailed information about an incorrect output type-of-failure related to VSAM RLS.

Incorrect output could have been caused by a previous failure. Examine the system and console logs for failure-related abends, messages, or return codes. A damaged VSAM data set can also cause incorrect output. Add any failure-related return codes to the keyword string, exactly as the system presents them. You can also add the abend or message type-of-failure keywords to the incorrect output keyword string to define the symptoms more closely.

Procedure

1. Determine whether failure-related VSAM RLS return codes and reason codes exist.

   VSAM RLS provides return codes in register 15 and reason codes in either the access method control block (ACB) or the request parameter list (RPL). Reason codes in the ACB indicate VSAM open or close errors. Reason codes in the RPL indicate record management error indications returned to the caller of RLS.

2. Record any failure-related RPL feedback word (a hexadecimal full word) and RPL return code on the Keyword Worksheet, in *z/OS DFSMSdfp Diagnosis*, as modifier keywords. The IBM Support Center can use these values to identify a failure-related module and the nature of the incorrect output.

   Example: If the RPL feedback word is X'000C0010', specify the following keywords:

   ```
   RPLFDBWD 000C0010
   ```

3. Determine whether you have a damaged VSAM data set.

   Some incorrect output failures involve a damaged VSAM data set. To determine whether you have a damaged data set, use the IDCAMS EXAMINE command as described in the chapter on functional command format in *z/OS DFSMS Access Method Services Commands* and the chapter on checking a VSAM key-sequenced data set cluster for structural errors in *z/OS DFSMS Using Data Sets*. The EXAMINE command provides details about the nature of data set damage.

   If these service aids indicate that the data set is not damaged, inform the IBM Support Center if you call for assistance. If they indicate that the data set is damaged, keep a copy of the output for possible use by the IBM Support Center. Be prepared to describe the type of data set damage. You should attempt to recover the data set and rerun the failing job to determine whether the problem is resolved.

   The system can indicate a damaged data set by one of the following:

   - Messages (discussed in Message section)
   - ABEND0C4 (discussed in ABEND section)
   - Wait/Loop (discussed in Wait/Loop section)
   - RPL feedback word: *nn*X'08'*nn*X'9C' or *nn*X'08'*nn*X'20'.

4. If the data set is damaged, rebuild it as directed in the topic that describes VSAM record management damaged data sets in *z/OS DFSMSdfp Diagnosis*, and rerun the job.
5. See *z/OS DFSMSdfp Diagnosis*.

DFSMStvs—incorrect output keyword

Use this topic to gather detailed information about an incorrect output type-of-failure related to DFSMStvs.

Incorrect output could have been caused by a previous failure. Examine the system and console logs for failure-related abends, messages, or return codes. A damaged VSAM data set can also cause incorrect output. Add any failure-related return codes to the keyword string, exactly as the system presents them. You can also add the abend or message type-of-failure keywords to the incorrect output keyword string to define the symptoms more closely.

Procedure

1. Determine whether failure-related DFSMStvs return codes and reason codes exist.

   DFSMStvs provides return codes in register 15 and reason codes in either the access method control block (ACB) or the request parameter list (RPL). Reason codes in the ACB indicate VSAM open or close errors. Reason codes in the RPL indicate record management error indications returned to the caller of DFSMStvs.

2. Record any failure-related RPL feedback word (a hexadecimal full word) and RPL return code on the Keyword Worksheet, in *z/OS DFSMSdfp Diagnosis*, as

modifier keywords. The IBM Support Center can use these values to identify a failure-related module and the nature of the incorrect output.

Example: If the RPL feedback word is X'000C0010', specify the following keywords:

```
RPLFDBWD 000C0010
```

3. Determine whether you have a damaged VSAM data set.

   Some incorrect output failures involve a damaged VSAM data set. To determine whether you have a damaged data set, use the IDCAMS EXAMINE command as described in the chapter on functional command format in *z/OS DFSMS Access Method Services Commands* and the chapter on checking a VSAM key-sequenced data set cluster for structural errors in *z/OS DFSMS Using Data Sets*. The EXAMINE command provides details about the nature of data set damage.

   If these service aids indicate that the data set is not damaged, inform the IBM Support Center if you call for assistance. If they indicate that the data set is damaged, keep a copy of the output for possible use by the IBM Support Center. Be prepared to describe the type of data set damage. You should attempt to recover the data set and rerun the failing job to determine whether the problem is resolved.

   The system can indicate a damaged data set by one of the following:

   - Messages (discussed in message section)
   - ABEND0C4 (discussed in ABEND section)
   - Wait/Loop (discussed in wait/loop section)
   - RPL feedback word: *nn*X'08'*nn*X'9C' or *nn*X'08'*nn*X'20'.

4. If the data set is damaged, rebuild it as directed in the topic that describes VSAM record management damaged data sets in *z/OS DFSMSdfp Diagnosis* and rerun the job.
5. See *z/OS DFSMSdfp Diagnosis*.

Catalog management—incorrect output keyword

Use this topic to define the keyword when the system produces other than the expected output and you suspect a failure in the catalog management area.

Procedure

Determine the extent of the incorrect output.

1. Use the LISTCAT command as described in the chapter on functional command format in *z/OS DFSMS Access Method Services Commands* to obtain a complete listing of the catalog.
2. Use the IEHLIST program as described in *z/OS DFSMSdfp Utilities* to obtain a listing of the VTOC. This might be useful for diagnosing problems in managing DASD volume space or in using access method services commands.
3. Use the DIAGNOSE command as described in the chapter on functional command format in *z/OS DFSMS Access Method Services Commands* to determine whether a catalog structure is correct. Include any reason codes produced by DIAGNOSE in your search argument.

   Example: If the reason code is 23, you would specify it as follows:

   ```
   DIAGNOSE   RC23
   ```

4. Use the IDCAMS EXAMINE command as described in the chapter on functional command format in *z/OS DFSMS Access Method Services Commands* and the chapter on checking a VSAM key-sequenced data set cluster for

structural errors in *z/OS DFSMS Using Data Sets* to determine whether the catalog being used has been damaged, and the nature of the damage.

If the output of these service aids (LISTCAT, IEHLIST, DIAGNOSE, or EXAMINE) indicates that the catalog is not damaged, inform the IBM Support Center if you call for assistance. If they indicate that the catalog is damaged, keep a copy of the output for possible use by the IBM Support Center. Be prepared to describe the type of catalog damage. You should attempt to recover the catalog and rerun the failing job to determine whether the problem is resolved.

5. See *z/OS DFSMSdfp Diagnosis*.

Abend keyword

Use this topic when your program (or ISMF session) abnormally ends (abends).

Symptoms of the failure

You can identify an abend by one or more of the following indicators:

- The printed system output of a program
- The text of a system message
- An ISMF abend panel
- An ISPF abend panel
- A TSO/E message that identifies an abend condition
- A SYS1.LOGREC record
- An SVC DUMP

The means by which the system indicates an abend condition provides sufficient evidence to determine which DFSMSdfp component received the abend. The evidence can be a message prefix or text, the operation performed, the module that detected the failure, an ISMF abend panel, and so forth.

A damaged VSAM data set can cause an ABEND0C4 abend condition in any of the modules in the following table. Repairing the data set resolves the problem.

A damaged VSAM data set can cause an ABEND0C4 abend condition in any of the modules in this table.

Module	Module	Module	Module
IDA019RC	IDA019RE	IDA019RF	IDA019RG
IDA019RH	IDA019RI	IDA019RJ	
IDA019RN	IDA019RW	IDA019R4	

To determine whether you have a damaged data set, use the IDCAMS EXAMINE command as described in the information on functional command format in *z/OS DFSMS Access Method Services Commands* and the informaiton on checking a VSAM key-sequenced data set cluster for structural errors in *z/OS DFSMS Using Data Sets*. The EXAMINE command provides details about the nature of data set damage.

For more information on diagnosing problems with damaged VSAM RLS data sets, see *z/OS BDT Diagnosis Reference*.

Procedure

When the system encounters an abend condition, it produces one or more of the following kinds of documentation: an SVC dump, a SYSABEND, a SYSMDUMP, or a SYSUDUMP. To determine the abend code, go to the procedure that the following table indicates.

Procedures for determining abend codes

Subcomponent	Procedure
ISMF/ISPF abend panel	See *z/OS DFSMSdfp Diagnosis*.
TSO/E message	See *z/OS DFSMSdfp Diagnosis*.
All other DFSMSdfp subcomponents	Continue with the next section, "Procedure for SVC dump."

Procedure for SVC dump

SVC dumps invoked by the SDUMP macro are usually written as a result of an entry into a *functional recovery routine* (FRR) or ESTAE routine. The *component recovery routine* specifies the addresses that are dumped and directs the dump to one of the SYS1.DUMP*xx* data sets. The SVC dump contains enough information for you to build the keyword string.

You can find the structured search keywords in the Summary Diagnostic Worksheet, in *z/OS DFSMSdfp Diagnosis*, under the section RETAIN® SEARCH ARGUMENT. Use these keywords in freeform searches.

If the abend code is X'08B', then SMS has experienced a "data in virtual" (DIV) abend. Do the following:

- Obtain the registers from the time of abend, using either the `IGD300I` message or the system diagnostic work area (SDWA).
- Examine the contents of register 15. The two low-order bytes contain the DIV reason code related to the abend. Append the reason code to the keyword prefix RC and record it on the Keyword Worksheet, in *z/OS DFSMSdfp Diagnosis*.

  See the description of the applicable DIV reason code listed under abend code 08B in *z/OS MVS System Codes*. It might help you define more closely the source of the failure. If it indicates that the problem is external to DFSMSdfp, continue the diagnosis process within the component involved.

If the abend code is X'0F4', then an error occurred during program management binder, DCME, HFS, PDSE, VSAM RLS, or DFSMStvs. Take the following actions:

- Review SYS1.LOGREC for X'0F4' software records, PDSE symptom records, and any other records produced at the time of the error. Program management binder, DCME, HFS, PDSE, VSAM RLS, and DFSMStvs symptom records are primarily used to identify incidents identified with program management binder, DCME, HFS, PDSE, VSAM RLS, and DFSMStvs X'0F4' abends. When program management binder, DCME, HFS, PDSE, VSAM RLS, or DFSMStvs symptom records occur without an X'0F4', use the symptom strings to search for matching problems in the IBM software support database, and if no errors exist, contact the IBM Support Center.
- Prior to the ABEND error, a return code is placed in general register 15 and a unique reason code is placed in general register 0 describing the exceptional condition. Append the reason code from general register 0 to the keyword prefix RSN and record it as a modifier keyword on the Keyword Worksheet, in *z/OS DFSMSdfp Diagnosis*. Remove the leading zeroes from the return code from general register 15 and append it to the keyword prefix RC. Record it as a modifier keyword on the Keyword Worksheet. Using the information from the Summary Diagnostic Worksheet in *z/OS DFSMSdfp Diagnosis* as an example, the modifier keywords would be as follows:

  ```
  abend0f4   rsn21042716  rc24
  ```

The primary use for PDSE reason codes is to search the IBM software support database; therefore, *z/OS DFSMSdfp Diagnosis* does not document reason code descriptions.

To determine keywords for SVC dumps, do the following:

1. Use IPCS to print the summary dump (SUMDUMP). See *z/OS MVS IPCS User's Guide*.
2. The title page of the Summary Diagnostic Worksheet contains the dump header and title page, which provide failure-related symptoms extracted from the dump. (See the Summary Diagnostic Worksheet in *z/OS DFSMSdfp Diagnosis*.) One or more of the following symptoms should be present:
   - ABEND*nnn*
   - Module or CSECT name or both
   - Component Identifier
   - Release Level
   - Service Level
   - FMID

   Refer to *z/OS MVS Diagnosis: Reference* and *z/OS MVS Diagnosis: Tools and Service Aids* for a detailed explanation of each symptom.
3. If you can identify the ABEND CODE by using the dump header and title page, see "Procedure for building the abend keyword" on page 304.
4. If the dump does not have a header title or does not otherwise enable you to identify the ABEND CODE, use the SUMDUMP printed from the SYS1.DUMP*xx* data set and continue with "Procedure for SYSABEND, SYSMDUMP, or SYSUDUMP."

Procedure for SYSABEND, SYSMDUMP, or SYSUDUMP

Depending on the JCL used, the system directs a dump to the SYSUDUMP, SYSABEND, or SYSMDUMP data set. If the system did not produce a dump, you might need to recreate the failure and obtain one. For information about obtaining a dump, see *z/OS MVS Diagnosis: Tools and Service Aids*.

SYSUDUMP data sets usually do not contain enough information to be useful in diagnosing a failure.

1. Obtain a system storage dump that contains the user's program.
2. Determine the system abend code by using either of the following sources:
   - The job-related information about the abend in the job log.

     This information includes the abend code, PSW contents, and general-purpose register contents. The abend code is 3 characters long. To obtain the job log, you must specify the JCL parameter `MSGLEVEL=(1,1)` on your JCL JOB card.
   - The system storage dump.
     - Locate the formatted section at the beginning of the dump. Determine the abending job by locating the job whose abend code field (TCBCMP at TCB + X'11') contains a nonzero value.
     - The field is only 3 characters long. Ignore the first (left-most) byte. If the abend code appears in the first 12 bits following the first byte, it is a system abend code. If it appears in the next 12 bits, it is a user abend code and the value must be converted to decimal.
3. When you determine the ABEND CODE, continue with "Procedure for building the abend keyword" on page 304.

Procedure for building the abend keyword

Take the following actions to build the abend keyword:

1. Use the ABEND CODE that you have extracted from the system-produced documentation.

   System abends are expressed in hexadecimal; user abends are expressed in decimal.

   - For a system abend, append the 3-character code to the keyword prefix ABEND.

     Example: If the abend code is 0C4, specify the abend type-of-failure keyword as:

     ```
     ABEND0C4
     ```

   - For a user abend, append the 4-digit code to the keyword prefix ABENDU.

     Example: If the abend code is 0222, specify the abend type-of-failure keyword as follows:

     ```
     ABENDU0222
     ```

2. If a message containing a return code accompanies the abend, include the return code in your keyword string as a modifier keyword. Append the code (specified exactly as it appears in the message) to the keyword prefix RC.

   Example: If the return code is 04, specify the keyword string as follows:

   ```
   ABEND0C4 RC04
   ```

   Tip: If the *z/OS MVS System Codes* description of the abend code indicates a return code and the reason code is associated with the abend, including both the return code and the reason code in your keyword string could restrict the scope of a software database search that results in no matches.

3. If the *z/OS MVS System Codes* description of the abend code indicates a reason code is associated with the abend, include the reason code in your keyword string as a modifier keyword. Append the code (specified exactly as it appears in the register) to the keyword prefix RSN.

   Example: If the abend code is X'0F4' then the reason code is found in general register 0. If the reason code is 0409F023, specify the keyword string as follows:

   ```
   ABEND0F4 RSN0409F023
   ```

4. See *z/OS DFSMSdfp Diagnosis* (the subcomponent-specific section, if one exists).

Message keyword

Use this topic for all DFSMSdfp message-related problems.

You can identify a message type-of-failure when one of the following conditions occurs:

- Message reports program or operation failure.
- Message is missing data, or contains invalid data.
- Message reports a data failure (catalog, user data).
- No message appears when one should have been issued.

Procedure

Before using this information, examine *z/OS MVS System Codes* and *z/OS MVS System Messages, Vol 1 (ABA-AOM)*, through *z/OS MVS System Messages, Vol 10 (IXC-IZP)*. These might help you generate additional keywords by identifying failure-related functions and providing message-to-module cross-reference tables.

Definitions of message keyword terms

The component-specific message keyword information uses the terms that Table 47 defines.

Table 47. Definitions of terms related to message keywords. Definitions of terms related to message keywords

Term	Definition
Message identifier	A three-letter prefix to identify the component that produced the message and a message serial number to identify the individual message (for example, IDC3009I).
Message keyword prefix	The characters MSG, to which the message identifier is appended. This comprises the message type-of-failure keyword.
Return or reason code	A numeric code contained in the message text. Either the message text or *z/OS MVS System Messages, Vol 1 (ABA-AOM)*, through *z/OS MVS System Messages, Vol 10 (IXC-IZP)*, can identify the type of code.
Return or reason code keyword prefix	The characters RC, to which each return or reason code (exactly as it appears in the message) is appended. (Each code in the text requires its own keyword prefix.) This comprises a modifier keyword to specify the failure-related symptom.

Go to one of the procedures that the following table indicates.

Locations of procedures.

Subcomponent	Procedure
DADSM/CVAF	See *z/OS DFSMSdfp Diagnosis*.
ISMF	See *z/OS DFSMSdfp Diagnosis*.
OAM	See *z/OS DFSMSdfp Diagnosis*.
SMS	See *z/OS DFSMSdfp Diagnosis*.
VSAM, DFSMStvs, and VSAM RLS record management	See "VSAM, DFSMStvs, and VSAM RLS record management—message keyword" on page 306.
All other DFSMSdfp components	See *z/OS DFSMSdfp Diagnosis*.

1. Append the message identifier to the keyword prefix MSG. Include in the keyword string any return codes and reason codes from the message text. Append the codes, exactly as they appear in the message to the keyword prefix RC.

   Example: If the message identifier is IDC3009I, the return code is 04, and the reason code is 032, you would specify the keyword string as follows:

   ```
   MSGIDC3009I RC04 RS032
   ```

   Rule: For PDSE-related failures (IGW messages), use RSN as the prefix for reason codes and RC as the prefix for return codes.
2. Message text might contain additional information that you can use as modifier keywords (function, subfunction, device-related information, and so forth); record it on the Keyword Worksheet, in *z/OS DFSMSdfp Diagnosis*.

3. For input/output or hardware-related errors, review the SYS1.LOGREC for keyword information.
4. See *z/OS DFSMSdfp Diagnosis*.

VSAM, DFSMStvs, and VSAM RLS record management—message keyword

VSAM record management does not issue any messages directly. However, the results of a record management request can be translated into a message that the user of record management issues. Use this information when your program or the system indicates that a VSAM data set is being processed.

Before using this information, examine *z/OS MVS System Codes* and *z/OS MVS System Messages, Vol 1 (ABA-AOM)*, through *z/OS MVS System Messages, Vol 10 (IXC-IZP)*. These might help you generate additional keywords because they identify failure-related functions and provide message-to-module cross-reference tables.

See "Definitions of message keyword terms" on page 305 for definitions of the following terms used in this topic:

- Message identifier
- Message keyword prefix
- Return or reason code
- Return or reason code keyword prefix

Procedure

1. A damaged data set can cause one of the following messages to be issued by the caller of VSAM record management or by a system service routine (for example, EOV or IOS) that was invoked by record management:
   - MSGIDC3302I—Action error
   - MSGIDC3308I—Duplicate records
   - MSGIDC3314I—Out-of-sequence records, missing records, duplicate records, no record found
   - MSGIDC3351I—VSAM logic I/O error RC156, RC24, or RC32
   - MSGIDC3350I—No record found or incorrect length
   - MSGIEC070I—RC32, RC202, RC104, or RC203
   - MSGIOS000I—Command reject
2. If the system issues one of these messages while processing a key-sequenced data set (KSDS), determine whether you have a damaged data set. Issue the IDCAMS EXAMINE command as described in the information on functional command format in *z/OS DFSMS Access Method Services Commands* and the information on checking a VSAM key-sequenced data set cluster for structural errors in *z/OS DFSMS Using Data Sets*. The EXAMINE command provides details about the nature of data set damage.

   Example: If a damaged data set caused message IDC3302I to be issued, you would specify the message type-of-failure keyword as follows:

   ```
   MSGIDC3302I
   ```
3. See *z/OS DFSMSdfp Diagnosis*.

VSAM diagnostic aids

This topic contains overviews of the major diagnostic aids that are provided for VSAM subcomponents and cites publications that contain more detailed information.

Access method services (AMS) diagnostic aids

This topic explains the diagnostic aids provided for access method services (IDCAMS), explains how to find key areas in a dump, and offers suggestions for isolating different types of problems.

The following major diagnostic aids are provided for access method services:

- Trace tables, which provide a trace of the flow of control between modules and within modules
- Dump points, which provide the facility to dump selected areas of virtual storage and make a full region dump
- The TEST option, which you can set to print out the trace tables or to obtain dumps at selected dump points if access method services is invoked with a batch job
- Termination codes and full-region dumps, which are produced when the processor detects an unrecoverable condition

Trace tables

The processor maintains two trace tables during each execution: the intermodule trace table, which records the flow of control *between*modules, and the intramodule trace table, which records the flow of control *within* modules.

You can find the trace tables in any full region dump, you can print them using the TEST option, or you can display them on a TSO/E terminal. "TEST option" on page 308 explains how to print the tables in a dump.

Intermodule trace table: The intermodule trace table begins with the characters INTER and contains the IDs of the last 20 modules that had control. The module IDs are the last 4 characters of the module name. For example, if the trace appears as follows:

```
INTER ... SA01 EX01 RI01 RI02
```

then you know that IDCRI02 had control at the time of the dump.

The intermodule trace table is updated by the system adapter not only as each module is entered, but also upon return from a module. Thus, if RI01 calls TP01 which calls IO01 and then returns back to RI01, the trace table appears as follows:

```
INTER ... RI01 TP01 IO01 TP01 RI01
```

Intramodule trace table: The intramodule trace table begins with the characters INTRA and contains the last 20 trace points encountered within modules. Each

module has trace points placed at key locations, for example, at the start of procedures and around calls to other modules.

The IDs of the trace points consist of 4 characters: the first 2 characters are the mnemonic identifier of the module being traced, and the last 2 characters identify a specific point within the module. The expansion of the UTRACE macro for trace ID DLLC appears as follows:

```
OLDERID2 = NEWERID2;
NEWID2 = 'DLLC';
```

Dump points

The IBM Support Center VSAM Customer Support will provide dump point information when a dump of a region or selected area is required for diagnosing a problem.

Each module has built-in dump points that invoke diagnostic dumping routines if the TEST option is in effect. The dump points have been placed at key locations in each module (for example, around calls to other processor and nonprocessor modules). Each dump point specifies what information will be dumped. Some dump points allow symbolic dumping of selected areas of virtual storage (for example, parameter lists or return codes); all dump points allow dumping of the full region and printing of the trace tables.

Certain access method services modules have the dumping of selected areas of virtual storage built in. Dump points can be used to dump these selected areas. The areas dumped vary with each dump point and are identified with descriptive codes.

Dumping of selected areas can occur with or without a full region dump, as described in "TEST option."

TEST option

If you invoked access method services in a batch job, you can use the TEST option to activate the printing of diagnostic output at selected points within access method services. The TEST keyword controls the TEST option, as "TEST keyword" explains.

The TEST option enables you to print the following information:

- The intermodule and intramodule trace tables.
- Selected areas of virtual storage.
- Full region dump.

Each variation of the TEST option provides an additional level of information. The possible variations follow:

- Print the trace tables only
- Print the trace tables and selected areas of virtual storage
- Print the trace tables and selected areas of virtual storage and take a full region dump.

TEST keyword: You can enter the TEST keyword either in the PARM field of the EXEC card that invokes the processor, or on a PARM command. By using the PARM command, you can turn the TEST option on and off or change the TEST option for different function commands.

The format of the PARM command follows.

The format of the PARM command

Command	Parameters
PARM	**[TEST({[TRACE]** **[AREAS** (*areaid* [*areaid*...])] **[FULL** ((*dumpid* [*begin* [*count*]]) [(*dumpid*...)....])]| **OFF})]**

where:

TEST({[TRACE] [AREAS (*areaid***[*areaid*...])] [FULL ((***dumpid***[*begin* [*count*]]) [(***dumpid***...)...])]| OFF})**
Specifies the diagnostic aids to be used. After the TEST option has been established, it remains in effect until it is reset by another PARM command. The TRACE, AREAS, and FULL parameters may be used concurrently.

TRACE
Specifies that trace tables are to be listed whenever the built-in dump points of the processor are encountered.

AREAS(*areaid* **[***areaid***...])**
Identifies modules that are to have selected variables dumped at their dump points. areaid is a 2-character area identifier defined within the implementation.

FULL ((*dumpid* **[** *begin* **[** *count***]]) [(***dumpid***...)....])**
Specifies that a region dump, as well as the trace tables and selected variables, is to be provided at the specified points. dumpid specifies the 4-character identifier of the dump point.

begin A decimal integer that specifies the number of times the program is to go through the dump point before beginning the dump listing. The default is 1.

count A decimal integer that specifies the number of times through the dump point that dumps are to be listed. The default is 1.

If the FULL keyword is used, an AMSDUMP DD statement must be provided. For example:

```
//AMSDUMP DD SYSOUT=A
```

OFF Specifies that testing is to stop.

Each time a PARM command is specified, the TEST parameters override the TEST parameters in effect from the previous PARM command.

Figure 11 on page 310 shows a portion of the output from the command:

```
PARM TEST ( FULL (LCTP,2,1) )
```

and a portion of the dump produced.

The trace tables and the selected area, DARGLIST, are printed each time the dump point LCTP is encountered. A full region dump is produced the second time that dump point LCTP is encountered.

How to use the TEST option: If a problem occurs and you have no idea which modules are involved, run the job again with the TRACE keyword. From the

intermodule trace table you can identify the modules involved. The TRACE keyword, however, produces a large amount of output.

If you suspect that specific modules are involved, you can rerun the job with the AREAS keyword and specify the identifiers of several suspected modules. You will obtain trace output only for the specified modules.

When VSAM Customer Support identifies the the dump points at which a full dump should be made, rerun the job with the FULL keyword. The AREAS and FULL keywords can be used in combination to obtain trace tables and selected areas throughout several modules, but a full region dump will be taken only at selected points.

```
IDCAMS  SYSTEM SERVICES                        TIME: 23:23:00          06/21/86          PAGE    1


   PARM    TEST ( FULL (LCTP,2,1) )

IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0

   LISTCAT  ENTRY MN01.CLO01041/CLMR ) ALL

 IDC0924I  DUMP ROUTINE INVOKED AT 'LCTP'

INTERMODULE TRACE: EX01 SA02 LC01 SA02 LC01 SA02 LC01 SA02 LC01 SA02 LC01 SA02 LC01 SA02 LC01 SA02 LC01 SA02 LC01 DP01
INTRAMODULE TRACE: R137 RINN SACL RITM SAFP R199 SADP EX1F EXFS SACL LCIN SAGP SAGP SAGP SAGP SAGP SAGP SAGP SAGP LCTP

   DARGLIST = 00000000 0023803C D3C3F0C1 00000000 01300000   <-- Selected fields; Text Processor Argument List
                                                                                  Module that called for dump
IDC0924I  DUMP ROUTINE INVOKED AT 'LCTP'

INTER-MODULE TRACE: DB01 TP01 D801 TP01 DB01 TP01 LC01 DB01 LC01 SAC2 DB01 SA02 DP01 SA02 LC01 DB01 LC01 DB01 LC01 DB01
INTRA-MODULE TRACE: SALD SAGP TP5N TPCC TP2I TP2I TPDB TPDB TPDB TP2N TB2N TB1N LCEN LCBL SACA ZZCA ZZCA LCAL LCWA LCTP

   DARGLIST = 00C00000 002405C5 D3C3F002 00000001 00790000 000A0C2C 0024063E   <-- Selected fields; Text Processor Argument List

IDC0925I  DUMP 001 PRODUCED AT DUMP POINT 'LCTP'
              ^
              |-- ID of snap-dump
```

DA6R1033

```
                    .
                    .                                   ID of snap-dump
                    .
JOB AMSLIST         STEP                TIME 232310     DATE 06/21/86     ID = 001            PAGE 0001

PSW AT ENTRY TO SNAP   071D2000 002033A8

SEGMENT TABLE ORIGIN REGISTER 02018A00

TCB     012C68  RB   0023F270  PIF    00000000 DEB  0024BD88  TIOT  0024F730  CMP   00000000  TRN  00000000
                MSS  00012198  PK/FLG 10010008 FLG  000001F5  LLS   00000000  JLB   00000000  JST  00012068
                FSA  0A24F6F0  TCP    000122F8 TME  000121D8  PIB   80000053  NTC   00000000  OTC  00000000
                LTC  00000000  IQE    000000C0 ECB  00000000  XTCB  00000000  LP/FL 55000000  RESV 00000000
                STAE 0024F7A8  TCT    0024FC18 USER 00000000  NDSP  00000000  MDIDS 00000000  JSCB 0001ED74
                RESV 00000000  RESV   00000000 RESV 00000000  EXT1  00000000  BITS  0000080C  DAR  00000000
                EXT2 00012148  PCB    00012158 GQE  0023F73C  ABP   00235310
        EXT2    GTF  00000000  ST/RCM 00000000


ACTIVE RBS

PRB    24FDF8  NM IDCAMS      SZ/STAB 2C0500D0   USE/EP 002C240C   PSW 071D2000 002033A8   C 000000   WT/LNK 00012068
               CDP   NOTE/LOAD 00200000   MODLNTH EE000A9C

SVRB   23F27C  NM SVC-A05A    SZ/STAB 0C14D062   USE/EP 002FBC00   PSW 07000000 002900E8   C 000273   WT/LNK 0024FDF8
               RG 0-7   00000030    0024EE00    0024B598    0023D7C0    000006BC    00012248    0024F2C0    0024C070
               RG 8-15  0024B598    0024F2C0    60202B38    0024EBC0    00203B37    0024EBC0    00202B54    00000000

JOB PACK AREA
                    .
                    .                                                                          DA6R1034
                    .
```

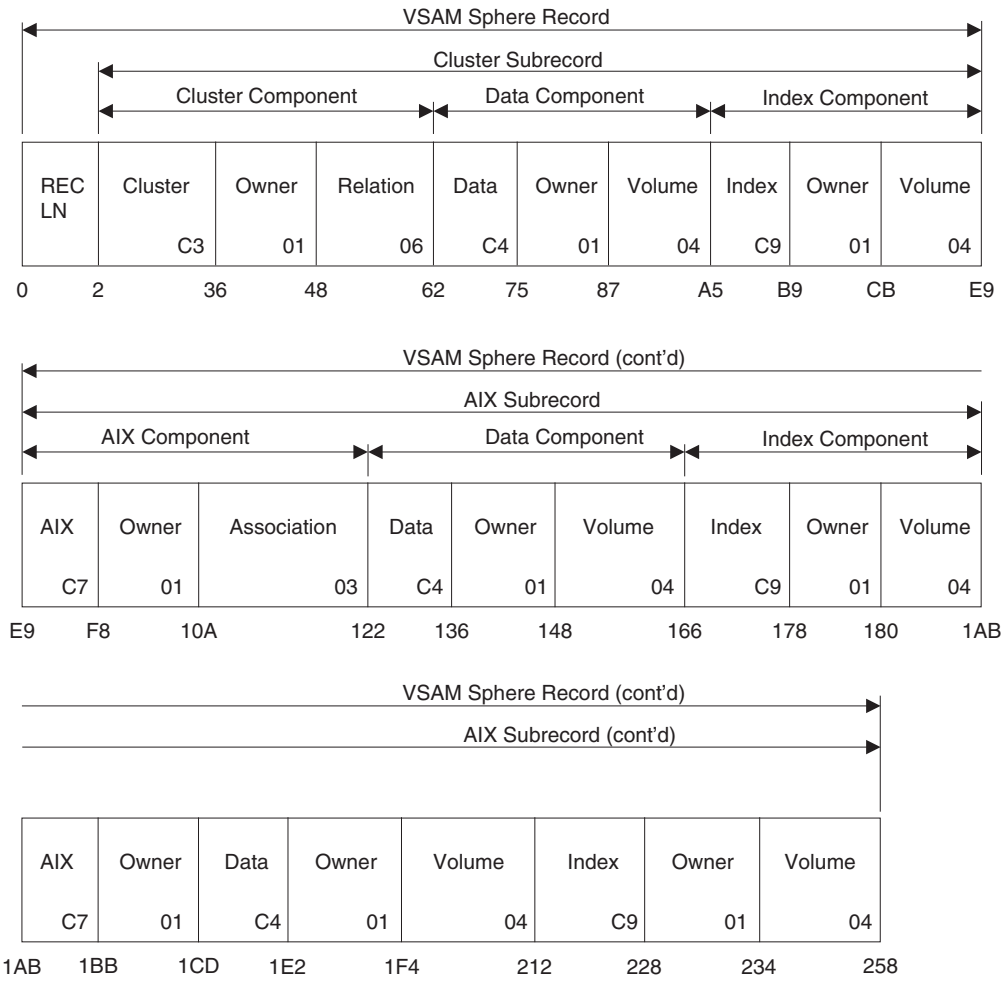
Figure 11. Example of TEST option output

Catalog management diagnostic aids

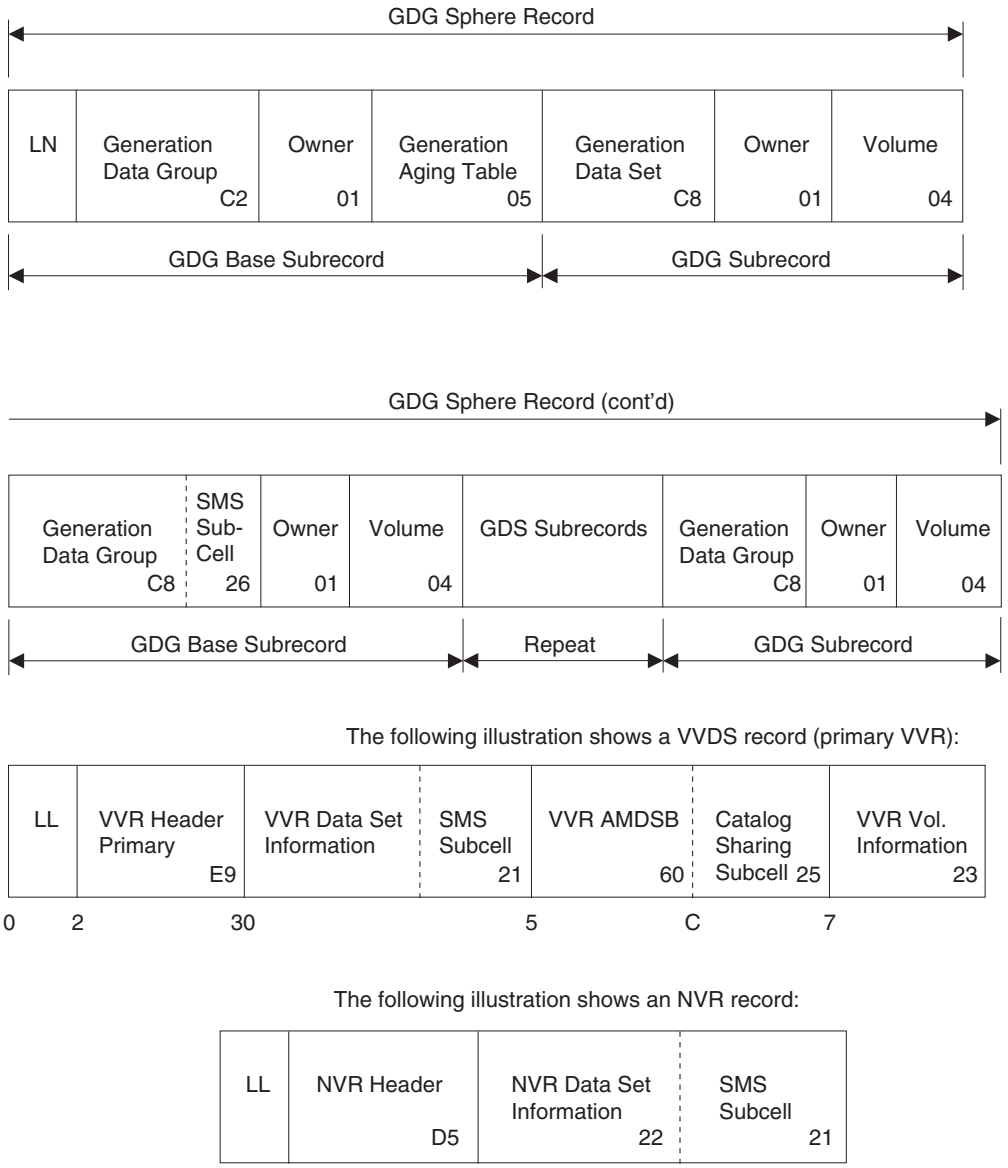
The DIAGNOSE command output is designed to help you diagnose catalog management problems.

Illustrations of catalog records follow, in the format in which they would be displayed after execution of the DIAGNOSE command.

The following illustration shows a VSAM sphere record for a KSDS data set:



The following illustration shows a GDG sphere record consisting of a base record and 10 subrecords (as indicated in the generation aging table cell):



VSAM OPEN/CLOSE/end-of-volume (O/C/EOV) diagnostic aids

The following major diagnostic aids are provided for VSAM O/C/EOV:

- A description of the GTF trace facility, which is used to trace VSAM O/C/EOV activity
- A description of failure-related messages that are issued by VSAM O/C/EOV and the function codes that they contain.

The diagnostic aids and return codes that are described in this topic are also applicable when RLS is active.

Generalized trace facility

Use the generalized trace facility (GTF) to obtain diagnostic information during the processing of VSAM Record Management (R/M). If GTF is active, and either **TRACE=** (record management tracing) is specified in your DD statement, VSAM GTF trace records, X'FF5' are written to the GTF data set.

VSAM GTF X'F61' record: The VSAM GTF X'F61' was designed to provide a method for tracing I/O activity for data in a shared resource pool. This GTF record is not supported for VSAM RLS.

Considerations for requesting X'F61' records: These items should be considered when you request X'F61'records:

- GTF must be started with the USRP (or USR) parameter specified with an event identifier of X'F61'.
- GTF X'F61' records will only be generated for those jobs processed with VSAM LSR or GSR.
- If the ATTACH macro is used, the SZERO=YES parameter must be coded.

Mapping of the X'F61' record: The X'F61' record is an information record to be used for specialized purposes. Because of its dependence on detailed design and implementation, IBM might change the mapping of the record with new product releases or versions or as a result of service. The following table gives detailed information about the X'F61' record.

Table 48. X'F61' record information. X'F61' record information

Offset	Name	Length	Description
0(X'00')	ASCB	4	ASCB address
4(X'04')	JOBNAME	8	Jobname
12(X'0C')	ASID	2	Current ASID
14(X'0E')	REQ_TYPE	1	Type of request
15(X'0F')	SHR_POOL	1	LSR resource pool ID
16(X'10')	CI	4	Requested CI
20(X'14')		4	Reserved
24(X'18')	FLAGS	1	Processing flags **Bit** — **Meaning when set** **0** — GSR (LSR if OFF) **1** — Index (data if OFF) **2** — CI found in Hiperspace or read from DASD (CI found in address space buffer pool if OFF) **3** — CI found in Hiperspace **4 – 6** — Reserved **7** — Cross-System Share Option 4 (share option 3 if OFF)
25(X'19')	XREG_SHROPT	1	Cross-Region Share Options • SHR(1)=X'01'X • SHR(2)=X'02'X • SHR(3)=X'03'X • SHR(4)=X'04'X
26(X'1A')	CISIZE	2	CI size for the component
28(X'1C')	DSNAME	44	VSAM component name
72(X'48')	VOLSER	6	Volume serial for the component

Table 48. X'F61' record information (continued). X'F61' record information

Offset	Name	Length	Description
78(X'4E')	BUFRSIZE	2	Buffer size
80(X'50')	TRANSID	1	RPL transid
81(X'51')	RECLEN	3	Record length

Interactive problem control system (IPCS)

The interactive problem control system (IPCS) provides installations with an interactive, online facility for diagnosing software failures. Using unformatted dumps, IPCS formats and analyzes them to produce reports that you either view at the terminal or print. As you examine the dump of a software failure, IPCS accumulates information about the dump. IPCS uses the information it gathers each time you process the dump.

See *z/OS MVS IPCS User's Guide* for information on how to use IPCS and *z/OS MVS IPCS Commands* for information on the IPCS commands and examples.

For product information and programming requirements needed to use IPCS, customize access to IPC and examples on writing exit routines, see *z/OS MVS IPCS Customization*.

IPCS enables you to analyze these types of dumps:

- System dumps
  - ABEND dumps
  - SVC dumps
  - Stand alone dumps.

  Tip: System dumps might include GTF trace data.
- Live system storage
- External GTF trace information written to direct access or tape storage devices

Printing GTF records: You can use the GTFTRACE subcommand of IPCS to format and print GTF records contained in a dump or a trace data set. GTF trace records can be formatted interactively in an IPCS session or in batch mode. You can select the kinds of trace data you want to process by using the appropriate trace data selection keywords.

This example creates a formatted report of the GTF trace records for SVC events:

```
gtftrace svc
```

The following two examples create a formatted report of the GTF trace record for VSAM R/M control blocks.

Interactively in an IPCS session:

1. The IPCS dump source is the trace data set (via DDNAME or DSNAME).
2. Enter the IPCS subcommand:

   ```
   GTFTRACE USR(AM01)
   ```

When using batch mode, you need to specify the name of the input trace data set and the names of the three output data sets in the JCL, as shown:

Input: Trace data set 'SYS1.TRACE'(TRACE DD)

Output:

- IPCS dump directory data set (IPCSDDIR DD)
- Formatted output (IPCSPRNT DD)
- TSO/E messages (SYSTSPRT DD)

IPCSPRNT, the IPCS print file, contains the dump or trace output. The SETDEF PRINT subcommand tells IPCS to direct output to the IPCS print file. The SETDEF TERMINAL subcommand tells IPCS to direct output to the SYSTSPRT file. TSO/E messages and some IPCS messages are directed to SYSTSPRT even if SETDEF PRINT NOTERMINAL was entered.

The IPCSDDIR dump directory data set is a VSAM indexed cluster that IPCS uses to store information. Usually, the dump directory is used internally by IPCS and IPCS formatting routines but there are subcommand interfaces for you to store and delete information. See *z/OS MVS IPCS User's Guide* for more information about the IPCSPRNT, SYSTSPRT, and IPCSDDIR files.

```
//JOBNAME   JOB  ,accounting
//STEP1     EXEC PGM=IKJEFT01,REGION=4096K,DYNAMNBR=50
//IPCSDDIR  DD DSN=IPCSU1.DUMP.DIR.DISP=SHR
//TRACE     DD DSN=SYS1.TRACE,UNIT=293,VOL=SER=338003,DISP=SHR
//IPCSTOC   DD SYSOUT=*
//IPCSPRNT  DD SYSOUT=*
//SYSTSPRT  DD SYSOUT=*
//SYSTSIN   DD *
 IPCS
 SETDEF NOCONFIRM PRINT NOTERMINAL DDNAME(TRACE)
 DROPDUMP
 GTFTRACE USR(AM01)
 DROPDUMP
 END
 /*
```

VSAM OPEN/CLOSE/End-of-Volume return and reason codes

VSAM O/C/E macros return to their caller return codes set in register 15 and reason codes set in the access method control block (ACB) ACBERFLG field (1 byte at ACB + X'31').

VSAM record-level sharing diagnostic aids

This topic explains the diagnostic aids provided for VSAM record-level sharing (RLS) and offers suggestions for isolating different types of problems.

The following diagnostic aids and types of problems are discussed:

- VSAM RLS component trace
- VSAM RLS IPCS processing
- VSAM RLS SMSVSAM abends
- VSAM RLS server recycle and reactivation
- Console dumps
- SMSVSAM initialization errors
- Problems sharing control
- VSAM RLS hang conditions
- VSAM RLS deadlock and timeout problems

When RLS is active, the return codes for VSAM O/C/EOV and record management macros are still applicable. See “VSAM OPEN/CLOSE/End-of-Volume return and reason codes” on page 315 and “VSAM record management return and reason codes” on page 337 for a list of these return codes and their explanations.

VSAM RLS component trace

DFSMSdfp provides a trace service for use with all functions provided by VSAM RLS. When the tracing is activated, a trace table is created in the VSAM RLS address space.

This trace facility is also used by other functions. Specific trace options for other functions are described in their sections.

You can use SYS1.PARMLIB member CT*ncccxx* to specify trace options for the TRACE CT command and to turn off tracing. CT*ncccxx* must exist at IPL for tracing to be available on the system. The default is CTISMS00. For more information on using CT*ncccxx* to specify tracing options, see *z/OS MVS Initialization and Tuning Reference*.

Trace also enables external writers (WTR) to capture trace buffers. Capturing trace buffers increases diagnostic capability and decreases the possibility of losing trace data.

For complete syntax and usage information for TRACE, see *z/OS MVS System Commands*.

TRACE command for VSAM RLS component trace: The syntax of the TRACE command for tracing the VSAM RLS component follows: TRACE CT,WTRSTART=*membername* or TRACE CT,WTRSTOP=*membername* or TRACE CT,ON | *nnn*K | *nnn*M | OFF,COMP=SYSSMS,PARM=CTISMS*xx*-->

PARM=CTISMS*xx* identifies the SYS1.PARMLIB member that contains the tracing options.

The default SYSSMS trace table size is 72 KB.

The *nnn*K or *nnn*M keywords can be used to specify trace table sizes of 16–999 KB or 1–2047 MB, respectively.

You are then prompted to specify the trace options to be in effect. You must respond with the REPLY command, using the following syntax: REPLYid ,JOBNAME=SMSVSAM ,ASID=(smsvsam-asidlist.) ,OPTIONS=(namename...),WTR=procname | DISCONNECT ,END-->

Note:

1. The *id* value is the identification number, 0-99, specified in the prompting message.
2. The REPLY may be continued on multiple lines; the END must be given as the final parameter to identify the end of the REPLY.
3. OPTIONAL (only needed when using CTRACE WRITER) - Create a member in SYS1.PROCLIB to allocate a CTRACE WRITER data set(s). For an example procedure, see *z/OS MVS Initialization and Tuning Reference*. The name of the

procedure is the name specified when starting the trace writer. For example, to start the trace writer with a procedure named CTWDASD, issue the following command:

```
TRACE CT,WTRSTART=CTWDASD
```

The TRACE and REPLY commands are fully described in *z/OS MVS System Commands*

Trace options: Table 49 shows the valid options for use with this trace facility.

Table 49. Valid trace facility options. Valid trace facility options

Trace event	Description
ENTRY	All module entries.
EXIT	Module exits with nonzero return codes only.
EXITA	All module exits.
CALL	Equivalent to specifying both ENTRY and EXIT.
UEXIT	Entries and exits of user exits.
RRTN	Entries and exits of recovery routines.
CB	Control block changes.
POST	Usage of certain MVS services.
SPECIAL	Entries and exits of commonly shared functions.
SUSP	Suspension or resumption of a work unit.
CONFIG	Configuration changes.
RLS1	Special purpose trace event for use under specific IBM programming support direction.
RLS2	Special purpose trace event for use under specific IBM programming support direction.
RLS3	Special purpose trace event for use under specific IBM programming support direction.
RLS4	Special purpose trace event for use under specific IBM programming support direction.
RLS5	Special purpose trace event for use under specific IBM programming support direction.
DLKPD	Deadlock/timeout/retained-lock problem determination trace event.
COMP=(SMSVSAM)	Specifies all the functional areas of VSAM RLS component.
SUBCOMP=(*component- list*)	Specifies specific subcomponents within the VSAM RLS component.

The following terms designate the subcomponents of VSAM RLS:

The subcomponents of VSAM RLS

Name	Name	Name	Name
BLC	SHM	SQM	VSS
CERS	SLCH	VOC	
SCM	SMLS	VQUI	
SHC	SMPM	VRM	

Note:

1. You can also specify **ALL** rather than list the **COMP** or **SUBCOMP** keywords. **COMP=(SMSVSAM)** is equivalent to specifying **ALL** subcomponents. However, in most cases you would trace only the specific subcomponents.
2. The **SUBCOMP** keyword can be specified without the **COMP** keyword.
3. Any of these options except for **ALL**, **COMP**, and **SUBCOMP** can be turned off by prefixing them with **NO** (for example, **NOVOC**, **NOSLCH**).

The following command turns off the trace facility:

```
TRACE CT,OFF,COMP=SYSSMS
```

Here are some examples of typical replies to activating the SMS trace:

- Trace record management entry/exit

```
R XX,OPTIONS=(ENTRY,EXITA,SUBCOMP=(VRM)),WTR=CTWDASD,END
```

- Trace deadlock/timeout problem determination information

```
R XX,OPTIONS=(DLKPD,SUBCOMP=(VRM)),WTR=CTWDASD,END
```

If special trace code were provided for RLS1, the following would be recommended by IBM programming support to utilize that special trace code:

```
R XX,OPTIONS=(RLS1,SUBCOMP=(VRM)),WTR=CTWDASD,END
```

Formatting the VSAM RLS component trace table: There is no mechanism to flush trace buffers in storage to the trace writer data sets. So, in addition to having trace writer data sets available, a storage dump of the SMSVSAM address space and data space is required. If the problem being traced does not result in a dump the MVS DUMP command can be used to produce an SVCDUMP-type dump containing the trace buffer data. For a complete description of this service, see *z/OS MVS IPCS Commands*.

It is possible with a sufficiently large trace table (8M recommended, use 16M if possible) that there is no data in the trace writer data sets.

The IPCS CTRACE command is used to format and view the information in the component trace writer data sets and the trace buffers in storage:

```
CTRACE COMP(SYSSMS) FULL LOCAL
```

SMSVSAM abends

Abends in the SMSVSAM address space can be identified by COMPID=DF122. The dump title (also available in syslog) contains:

- COMPID=DF122
- CSECT and offset of the instruction in error
- Compile date and maintenance level of the CSECT
- The abend code
- Return and reason codes

When the failing CSECT cannot be identified, **????** will appear in the dump title for the CSECT name.

In some cases you can identify the failing CSECT using IPCS option 6, ST command to find :

- The failing PSW
- The general purpose registers and ARs
- The calling sequence

Use the panel provided by the IPCS IGWFPMAN to run option Q to find the calling sequence.

Note: This procedure might not provide information for all abends.

Activation of the SMSVSAM address space

The SMSVSAM address space automatically starts at IPL if the RLSINIT (YES) keyword is specified in the IGDSMS*xx* member of SYS1.PARMLIB. You can also start the SMSVSAM address space after IPL by issuing the following command from the MVS console:

```
V SMS,SMSVSAM,ACTIVE
```

Termination of the SMSVSAM address space

To terminate the SMSVSAM address space, issue the following command from the MVS console:

```
V SMS,SMSVSAM,TERMINATESERVER
```

Use the VSMS,SMSVSAM,TERMINATESERVER command before you partition a system out of the XCF Sysplex. Failure to do so can result in unexpected abends in the SMSVSAM address space.

To terminate and automatically restart the SMSVSAM address space, issue the following command from the MVS console:

```
FORCE SMSVSAM,ARM
```

You should use the FORCE SMSVSAM,ARM command if the SMSVSAM address space did not teminate successfully with the V SMS,SMSVSAM,TERMINATESERVER command. In this case, make a console dump of the hung SMSVSAM address before you issue the FORCE SMSVSAM,ARM command and report the problem to your IBM representative.

The SMSVSAM address space automatically terminates and restarts after detecting fatal internal errors. If this situation occurs, report the failure, along with any dumps that were produced, to your IBM representative.

After any six consecutive restarts of the SMSVSAM address space, the system issues message IGW418D, which prompts the MVS operator for a reply to cancel or to continue with SMSVSAM initialization.

Console dumps

Console dumps that are generated when VSAM RLS is running will contain the SMSVSAM address space, its data spaces, and the catalog address space.

For example:

```
DUMP COMM=(EXAMPLE OF SMSVSAM DUMP)
R nn,JOBNAME=(SMSVSAM,CATALOG),CONT
R nn,DSPNAME=('SMSVSAM'.SMSVSAM,'SMSVSAM'.MMFSTUFF),END
```

SMSVSAM initialization errors

Message IGW414I is issued when the SMSVSAM server initializes successfully.

When initialization is not successful, use the D SMS,SMSVSAM command to gather data that can be used to determine the cause of failure.

You can use Table 50 on page 320 as a guide to what actions to take for specific initialization problems.

Table 50. Initialization errors. Guide to what actions to take for specific initialization problems

Symptom	Action
Server cannot successfully connect to the lock structure, IGWLOCK00.	Save the output from the following command: `D XCF,STR,STRNAME=IGWLOCK00`
Connectivity is not available.	If you have another Coupling Facility available, initiate a rebuild to cause the lock structure to become resident in the other lock structure. If a rebuild cannot be done, SMSVSAM will not initialize until the connectivity problem is resolved.
Waiting for replies from other systems.	1. Dump the SMSVSAM address spaces from all the systems in the Parallel Sysplex. 2. Recycle SMSVSAM address spaces on the systems that have not replied.
Structure does not exist.	If possible, modify your policy to allow the lock structure to be defined. The SMSVSAM address spaces should attempt to reconnect when they are notified that a new policy is in place. If they do not reconnect, you might want to recycle all the servers.
Other problems that occur during initialization.	Take a console dump of the SMSVSAM address space on all systems and contact IBM's Customer Support for problem diagnosis.

Sharing-control problems

Problems in sharing control are usually indicated by abends in the IGWXS*xxx* modules or by incorrect output from the SHCDS command.

Contact IBM's Customer Support when you have the following documentation:

- DFSMSdss dump of the active sharing-control data sets
- Abend dumps
- Syslog
- SHCDS command output
- Console dump (only when the SHCDS command output is incorrect)

The example job for obtains a DFSMSdss dump of the sharing-control active data set. Change the OUT and DUMP statements as appropriate.

```
//DMPSHCDS JOB ...
//* JOB TO DSS DUMP SHARING CONTROL ACTIVE DATA SET. CHANGE JOB
//* AND OUT DD STATEMENTS AS APPROPRIATE.
//DSSDMP  EXEC  PGM=ADRDSSU,REGION=4M
//SYSPRINT DD SYSOUT=H
//OUT DD DSN=DUMP.SHCDS,DISP=(,KEEP),UNIT=CART,
// VOL=SER=CART01,LABEL=(1,SL)
```

```
//SYSIN    DD  *
  DUMP   DS(INCLUDE(SYS1.DFPSHCDS.ACTIVE.SPLXPK))  SHARE -
         TOLERATE(ENQFAILURE) -
         OUTDD(OUT)
```

VSAM RLS hang conditions

Should a hang condition occur, you can generate dumps of the following spaces from all systems:

- SMSVSAM address space
- SMSVSAM dataspace
- Catalog address space

The following example shows how to generate a dump SYS01 for systems SYS01, SYS02, and SYS03.

```
SYS01   DUMP COMM=(MULTI SYSTEM DUMP)
SYS01   R nn,JOBNAME=(SMSVSAM,CATALOG),CONT
SYS01   R nn,REMOTE=(SYSLIST=(SYS02,SYS03)),CONT
SYS01   R nn,DSPNAME=('SMSVSAM'.SMSVSAM,'SMSVSAM'.MMFSTUFF),END
SYS01   R nn,DSPNAME=('SMSVSAM'.*),END
```

To get all data spaces, use .* as the last line of the example shows.

You might need to dump one or more of the users' address spaces if it appears that they are involved in the hang condition.

IBM's Customer Support might request a report of the active threads. Invoke IPCS IGWFPMAN and run options 1 and 2 to obtain a summary of TCBs running in the SMSVSAM address space and the users' requests that were active at the time of the failure.

VSAM RLS deadlock and timeout problems

Deadlock or timeout errors in SMSVSAM will cause CICS to issue messages DFHFC0164, DFHFC0165, DFHFC0166, and DFHFC0167. They should provide enough information to determine the cause of the problem.

If additional information is needed, the CICS FC level 2 trace will provide the VPDI (mapped by IFGVPDI). The VPDI is the source of the information provided in the CICS messages. The SMSVSAM trace OPTIONS=DLKPD can be used to trace the VPDI.

Note: The information returned in the VPDI can be out of date. There is a lag between the time the problems are detected and the request fails. For a timeout, information on the blocking holder might not be returned because that request completed before the information could be extracted.

VSAM record-level sharing return and reason codes

This topic describes the return codes from SMSVSAM, SMPM_CFPurge, and SMPM_CFQuery.

Return codes from SMSVSAM

The SMSVSAM address space is the server for VSAM RLS.

The following return and reason codes from SMSVSAM might appear in messages issued by CICS, CICSVR, and DFSMSdss. They might also include ERRDATA, that can be used to explain the return and reason code information.

For return codes from VSAM O/C/EOV and record management macros, see "VSAM OPEN/CLOSE/End-of-Volume return and reason codes" on page 315 and "VSAM record management return and reason codes" on page 337.

Contact IBM Customer Support for errors with return code DEC 36 (X'24'). A system dump might have been taken.

Return code 4 is given for information errors.

Return code 8 can indicate an error using the SMSVSAM interfaces (application logic error) or setup problems (such as missing RACF authority).

Server-not-available type errors indicate that the SMSVSAM server was not active at the time the request was made, and are not an error condition.

Table 51. SMSVSAM return and reason codes. SMSVSAM return and reason codes

Return code	Reason code	Explanation
4	X'60EF0008'	There are no locks to bind or unbind.
4	X'61FF0000'	The request is successful, but non-RLS mode is established.
4	X'61FF0011'	The request is successful, RLS mode established. Other connectors constant.
4	X'61FF0012'	The request is successful, RLS mode established. Lost locks constant.
4	X'61FF0013'	The request is successful, RLS mode established. Retained locks constant.
4	X'61FF0014'	The request is successful, No quiesce operation is performed. System is in local mode (as opposed to sysplex mode). Local mode constant.
8	X'60FF0001'	Request to connect a subsystem that is already active in the Parallel Sysplex.
8	X'60FF0002'	Request to connect an online application, which is already connected as a batch application.
8	X'60FF0003'	Request to connect/disconnect an online application, subsystem name was not passed.
8	X'60FF0004'	Request to connect or disconnect an online application, and the subsystem name has an invalid format.
8	X'60FF0005'	Request to connect an online application, and no quiesce exit was provided.
8	X'60FF0006'	Application is not authorized to the RACF SUBSYSNM class.
8	X'60FF0007'	Unable to obtain virtual storage.
8	X'60FF0008'	The subsystem name specified is not registered.
8	X'60FF0009'	Attempt to disconnect an online application, but the application has OPEN RLS ACBs.
8	X'60FF000A'	Attempt to connect or disconnect an online application, and a control ACB was not passed.
8	X'60FF000B'	SMSVSAM server is not available
8	X'60FF000C'	Connect or Disconnect request issued in invalid mode (e.g. SRB mode instead of TCB mode)

Table 51. SMSVSAM return and reason codes (continued). SMSVSAM return and reason codes

Return code	Reason code	Explanation
8	X'60FF000D'	Application has been canceled.
8	X'61FF0001'	Catalog Locate Failure for the specified sphere. ERRDATA contains additional information
8	X'61FF0002'	The specification is not an SMSVSAM data set.
8	X'61FF0003'	A Quiesce operation is already in progress for this sphere. Would occur if two DFSMSdss jobs concurrently tried to dump the same sphere.
8	X'61FF0004'	A batch application has the sphere open for output and this is an attempt to take a non-BWO copy.
8	X'61FF0005'	Request was not issued in supervisor state.
8	X'61FF0007'	The request to copy/dump has been canceled.
8	X'61FF0008'	The request type is not supported.
8	X'61FF0009'	Virtual storage was not available.
8	X'61FF000A'	SMSVSAM server was not available.
8	X'61FF000B'	The copy is not valid, since the SMSVSAM server failed during the copy.
8	X'61FF000C'	The task which issued the request already has an active non-RLS open for the sphere.
8	X'61FF000D'	Request to terminate the operation, but there is no operation in effect
8	X'61FF000E'	Request mismatch.
8	X'61FF000F'	Request to terminate the operation, but the operation is being canceled.
8	X'61FF0010'	There is a non-RLS quiesce active within this address space. The situation across the sysplex is unknown.
8	X'60EF0001'	Error accessing sharing control. ERRDATA has qualifying information.
8	X'60EF0002'	Catalog locate failed. ERRDATA contains qualifying information.
8	X'60EF0003'	The requestor does not have update authority to the sphere.
8	X'60EF0004'	The request passed an invalid ACB.
8	X'60EF0005'	The request did not specify an SMS VSAM data set.
8	X'60EF0006'	The request was issued in an invalid mode.
8	X'60EF0009'	No virtual storage.
8	X'60EF000A'	Invalid request parameter list.
8	X'60EF000B'	Open for output failed. ERRDATA contains qualifying information.
8	X'60EF000C'	Dynamic allocation failure. ERRDATA contains qualifying information.
8	X'60EF000D'	Dynamic unallocation failure. ERRDATA contains qualifying information.

Table 51. SMSVSAM return and reason codes (continued). SMSVSAM return and reason codes

Return code	Reason code	Explanation
8	X'60EF000E'	Open failure. ERRDATA contains qualifying information.
8	X'60EF000F'	Close failure. ERRDATA contains qualifying information.
8	X'60EF0010'	Incorrect parameter list passed.
8	X'60EF0011'	Number of clusters in original sphere and restored sphere do not match.
8	X'60EF0012'	Recovery required expected, but was not set. ERRDATA contains qualifying information.
8	X'60EF0013'	Request issued in invalid mode.
8	X'60EF0014'	Data set not found.
8	X'60EF0015'	An RLS cell does not exist. The operation cannot be performed.
36	Varies	Severe error. ERRDATA contains qualifying information.

Return codes from SMPM_CFPurge

Table 52 shows the return and reason codes that the SMPM_CFPurge interface can return.

Table 52. SMPM_CFPurge return and reason codes. SMPM_CFPurge return and reason codes

Return code	Reason code	Explanation
4	X'64xxFA10'	Request completed successfully. Retained locks were not purged as part of the request, and recovery is required.
4	X'64xxFA12'	Request completed successfully. Retained locks were purged as part of the request, and recovery is required.
4	X'64xxFA1E'	Request completed successfully. No purge was necessary because the system is not using VSAM RLS.
4	X'64xxFA1F'	Request completed successfully. No purge was necessary because the system is not running in Parallel Sysplex mode.
4	X'64xxFA20'	Request completed successfully. Retained locks were purged as part of the request.
8	X'64xxFA0E'	The base cluster name passed was not found in the catalog. No CF Cache information is retained for this sphere.
8	X'64xxFA0F'	An open sphere prevented the operation from completing.
8	X'64xxFA11'	Locks were held and the request specified SMPM_Fail.
36	Varies.	Severe error. ERRDATA contains qualifying information.

Return codes from SMPM_CFQuery

The return and reason codes shown in Table 53 can be returned from the SMPM_CFQuery interface.

Table 53. SMPM_CFQuery return and reason codes. SMPM_CFQuery return and reason codes

Return code	Reason code	Explanation
4	X'64xxFA1E'	Request completed successfully. No query was necessary because the system is not using VSAM RLS.
4	X'64xxFA1F'	Request completed successfully. No query was necessary because the system is not running in Parallel Sysplex mode.
8	X'64xxFA0E'	The base cluster name passed was not found in the catalog. No CF Cache information is retained for this sphere.
36	Varies.	Severe error. ERRDATA contains qualifying information.

VSAM record management (R/M) diagnostic aids

The following major diagnostic aids are provided for record management:

- Explanations of how to analyze exclusive control errors and physical I/O errors
- Explanations of how to analyze ABEND0CX errors
- An explanation of how to diagnose and recover from damaged indexes and damaged data control intervals
- A description of the record management trace facility
- A description of record management return codes.

Control block information

VSAM R/M uses two types of control blocks:

- User control blocks (RPL, ACB, and EXLST)
- Control blocks built for record management by VSAM OPEN (all other control blocks).

The RPL's RPLFDBWD field (4 bytes at offset X'0C') provide valuable failure-related codes. For descriptions of the return codes and reason codes, see "Understanding VSAM macro return and reason codes" on page 190.

The VSAM SNAP dump facility provides hexadecimal listings of some control blocks and data areas. To obtain a SNAP dump, perform one of the following tasks:

- Code the SNAP macro with the SDATA=CB option (described in *z/OS MVS Programming: Authorized Assembler Services Reference SET-WTO*).
- Specify this option to ABDUMP via the CHNGDUMP operator command or the IEAABD*nn* or IEADMP*nn* members of SYS1.PARMLIB.

The SNAP dump facility is only available at SYSABEND time.

UPAD exit or WAITX

If a UPAD exit is provided (RPLWAITX flag - bit 5 at RPL +X'29') for a deferred request, VSAM takes the UPAD exit for event completion. When the request can be restarted, VSAM takes the UPAD exit for a POST if in cross-memory or SRB mode. If not in cross-memory or SRB mode, VSAM issues a POST SVC. The

user-provided UPAD exit must return control to the posted request in VSAM at the point just following where control was given to the UPAD WAIT exit. For more information about UPAD and WAITX, see *z/OS DFSMS Using Data Sets*.

VSAM RLS does not support UPAD exits.

RLSWAIT exit

For synchronous VSAM RLS requests only, by specifying RPLWAITX the application can provide an RLSWAIT exit that issues WAIT for the completion of the request. When the request is completed, VSAM POSTs the request. The RLSWAIT exit ensures the ECB is marked POSTed before returning to VSAM. Note that RLSWAIT is entered only for a request completion wait, never for a resource or I/O completion wait. See *z/OS DFSMS Using Data Sets* for more details.

Error conditions

VSAM R/M returns to its caller return codes in register 15 and feedback words set in the request parameter list (RPL + X'0C'). For a description of such codes, see "Understanding VSAM macro return and reason codes" on page 190.

When diagnosing a problem, record the 4-byte RPL feedback word on the keyword worksheet in *z/OS DFSMSdfp Diagnosis*. Append the 4-byte field to the keyword prefix **RC**.

Example: The RPL feedback word is **2D08009C**; specify this modifier keyword as shown:

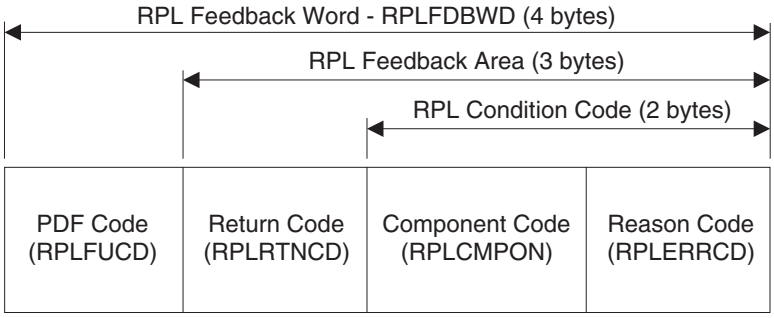
```
RC2D08009C
```

The IBM Support Center can use these codes to determine which module is in control and the reason the error occurred. You can use the codes in the RPL feedback word with other portions of the VSAM record management diagnostic aids to obtain additional failure-related information. The IBM Support Center may request an additional dump using the SLIP service aid to trap the error code when it is being set by a module.

The RPL feedback word is located at offset 12 (X'0C') in the RPL and contains the following information.

Information in RPL feedback word

RPL	Length	Description
12(X'0C')	1	Problem determination function (PDF) code. This helps identify the module that detected the error.
13(X'0D')	1	RPL return code. This code is returned in register 15.
14(X'0E')	1	Component code. This code identifies the component being processed when the error occurred.
15(X'0F')	1	Reason code. This code, when paired with the return code in the second byte above, identifies the reason for an error.



When analyzing a problem involving record management, consider the following items:

- What kind of application program was running when the error was encountered?
  - User application
  - IMS application
  - CICS application
  - CICS running under IMS.
  - System function (SMF/catalog)
  - Other application.
- What was the application program attempting to do when the error occurred?
- What were the request options? (Get/Put, Dir/Seq, Syn/Asy, and so forth).
- Where, within record management, was the error encountered?
  - Abend type-of-failure: use failing PSW instruction address.
  - Wait type-of-failure: use information from the RB that issued the Wait SVC.
  - RPL feedback error (RPL +X'0C'): See "Understanding VSAM macro return and reason codes" on page 190.
- What was record management attempting to do when the error occurred?
  - If the failure-related documentation is produced while record management is processing, you can use the PLH R14 push-down stack to identify the function being performed when the failure occurred.
- Is the data set damaged?
  - You may use the EXAMINE command to analyze a data set's integrity (whether it is damaged or not). If a data set is damaged, EXAMINE provides details about the nature of the damage. See the chapter describing functional command format in *z/OS DFSMS Access Method Services Commands* for details about the EXAMINE command.

Exclusive control error analysis

If a second request requires a record that is in a buffer in which a previous request is positioned, the second request might fail with an exclusive control error (RPLERRCD field at RPL +X'0F' = X'14'). If a message area is provided in the second RPL, VSAM record management returns the address of the RPL that has the position of the resource in the message area of the second RPL. You are responsible for relinquishing position in the initial RPL before redriving the second RPL.

An exclusive control conflict might also arise if an initial request causes a CI or CA split. A second request might receive RPLERRCD = X'14' as just mentioned, if it needs a record in the same CI or CA. You should either wait for completion of the

request whose RPL is pointed to by the second RPL's message area, or issue "ENDREQ" against the RPL pointed to by the second RPL's message area before redriving the second request.

ABEND0CX error analysis

The dump analysis procedure used for a record management ABEND0CX is essentially the same as that used for other DFSMS components. First, locate the registers at the time of the abend and the address at which the abend occurred. Using a listing of the failing module, determine the reason for the abend. You can use the PLH R14 push-down stack to determine the flow of the request through record management.

Damaged data sets (Non-RLS access)

If multiple systems share a data set and you fail to run an explicit **VERIFY** before opening the data set for output from either system (distinct from the implicit **VERIFY** initiated by VSAM Open), you may cause data set damage.

Determining which data set might be damaged: If you have a dump resulting from an ABEND0C4 in load module IDA019L1, or a console dump of a wait or loop with the PSW pointing into load module IDA019L1, then you can find the data set name (DSN) in the area pointed to by register 3. In most cases, Register 3 points to the AMB, and the data component name appears at offset X'88' into the AMB.

If you suspect a wait or loop in record management and the registers do not point into load module IDA019L1 (indeed, the symptoms of the dump seem to indicate a problem external to record management, for example, CICS waits), you can locate the AMB by using the RPL. Figure 12 shows how to do this:

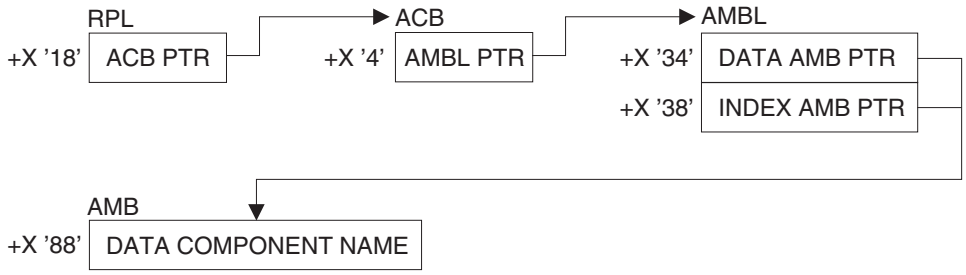


Figure 12. VSAM record management—how to find a damaged data set

Note: Because of its dependency on design and implementation, the control block offsets may change in new product releases or versions as a result of service.

RLS has a different structure. The user's RPL at RPL+4 points to another RPL in the VSAM server address space. The ACB, AMBL, and AMB are all in the VSAM server address space.

Damaged indexes

If the index component of a key-sequenced data set (KSDS) is damaged, the system may indicate the problem in any of several different ways. These include:

- Abends
- Loops
- Waits
- Missing records

- Duplicate records
- Out-of-sequence records.

Table 54 lists typical symptoms of a damaged data set when IDCAMS detects the damage.

- Messages

Table 54. Messages that IDCAMS detects. Messages that IDCAMS detects

Message ID	Condition
MSGIDC3302I	Action error
MSGIDC3308I	Duplicate record
MSGIDC3314I	Out-of-sequence record, missing records, duplicate records, no record found
MSGIDC3351I	VSAM I/O error RC156 or RC24 or RC32
MSGIDC3350I	No record found (NORECF) or incorrect length
MSGIEC070I	RC32 RC202 or RC104 RC203
MSGIEA000I	Command reject (CMD REJ)

- ABEND0C4 in any of the following VSAM modules:

```
IDA019RC      IDA019RE      IDA019RG      IDA019RH
IDA019RI      IDA019RJ      IDA019RN      IDA019RW
```

- Loops:
  - Loops issuing SVC121 (SVC X'79') or Start I/O (SIO)
  - Loops in any of the following VSAM modules:

```
IDA019RA      IDA019RB      IDA019RC      IDA019RE
IDA019RH      IDA019RI      IDA019RJ      IDA019RN
IDA019RW      IDA019R2      IDA019R3 (IDAM19R3)
```

- Waits issued from IDA019RZ
- Waits with the TCB structure indicating a wait in IDAM19R3 (IDA019R3)
- RPL feedback word (RPLFDBWD) as shown:

```
2D08009C      9208009C      A608009C      A708009C
9108009C      D808009C      E008009C      D708009C
2A080020      2B080020      2C080020      DB080020
DF080020
```

- RPL error code, (RPLERRCD) as shown:

```
RC32   (Return Code x'20')
RC156  (Return Code x'9C')
```

Recovering from index damage:

1. Use the access method services REPRO command to copy the data component of the KSDS. Specify the data component name (not the cluster name) in the REPRO *INFILE* parameter. See the chapter describing functional command format in *z/OS DFSMS Access Method Services Commands* for details about the REPRO command.
2. Use a sort utility to sort the copied file. This eliminates any duplicate keys that might exist in the file.
3. DELETE and re-DEFINE the damaged cluster.
4. REPRO from the sorted file to the newly defined cluster. Record management rebuilds the index component.

   Note:

a. This method does not work for a catalog or for a keyed cluster that contains spanned records.
b. Processing performed against the data set after initial index damage occurs will corrupt the data set content. This method is an attempt to recover as much as possible of the data set. If a backup copy of the data set exists, it may contain more desirable data set content than available by this method.

Damaged data control intervals

Another form of data set damage can occur to KSDS, ESDS, and RRDS data sets. A control interval (CI) may become damaged. This kind of damage may cause several symptoms, including RC156 (X'9C'), and I/O failures such as incorrect length or no record found.

Recovering from data CI damage:

1. REPRO. the data component of the cluster. Specify the data component name (not the cluster name) in the REPRO *INFILE* parameter.
2. This repro will fail; however, you will have copied all the available data that precedes the damaged portion of the data set. The high-used RBA value in the catalog entry for the data set that you specify in the REPRO *OUTFILE* parameter indicates how many bytes were successfully recovered before the damage was encountered. To recover valid data following the damaged CI, add the CISIZE (from the LISTCAT of the damaged data set) to the high-used RBA of the OUTFILE. Copy from the damaged data set data component using the REPRO *FROM* parameter. Copy *FROM* the high-used RBA + CISIZE, effectively skipping the damaged CI.
3. Continue this process, skipping damaged CIs until you have retrieved all valid data.

   Note: The extent of the damage affects how much data you can successfully recover. If a backup copy of the data set exists, it may contain more valid data than the data set that results from this recovery method.

Physical I/O errors

The IBM Support Center may request a CCWTRACE of I/O activity to examine the I/O to the DASD device. Do this by invoking GTF with the CCW, IO, IOSB, and SSCH options. Use the trace output to examine the data sent to and received from the DASD device by the VSAM block processor. See *z/OS MVS Diagnosis: Tools and Service Aids* for details about using CCWTRACE.

VSAM record management trace facility (non-RLS access)

Use the VSAM record management trace facility (R/M trace) to record VSAM record management control blocks while VSAM is processing. GTF must also be active for the R/M trace function to write records to the trace data set. See “Generalized trace facility” on page 312 for additional information on using GTF for tracing in VSAM.

Use **IPCS** to print the GTF trace records. Because a trace work area for the data set being traced is obtained in private low storage at the time the data set is opened, activating R/M trace may cause some storage overload.

Note: The VSAM Record Management Trace is a tool. Overlaying or releasing VSAM's control blocks while the trace is active may cause unpredictable results, such as, invalid trace information and various abends.

When to use the record management trace facility: Use the R/M trace to do these tasks:

- Capture data when a problem occurs.

  Problems include incorrect data in a data set, missing records, incorrect control block information, and program checks because of incorrect data and/or fields in VSAM R/M control blocks.
- Capture VSAM R/M control blocks **before an error code is passed back to the caller**.

  VSAM R/M control blocks are captured before the calling program can:
  - Erase or overwrite the VSAM data
  - Abnormally terminate
  - Close the data set, freeing VSAM R/M control blocks.

Starting the record management trace function: You must take the following steps to use trace:

1. Activate GTF on your system. When you start GTF, specify **USR** or **USRP** with an **AM01** event identifier, which is X'FF5'. If neither USR or USRP is specified, GTF ignores the data passed to it by trace.
2. Place an **AMP=**('TRACE=(subparameters)') on the DD statement of the data set that you want to trace.

The **AMP=('TRACE=(**subparameters**)')** specifies to VSAM that you want to trace the control blocks associated with the data set identified by the DD statement. For complete syntax information on the AMP parameter, see the section on coding the DD statement in *z/OS MVS JCL Reference*.

The following are the valid subparameters for trace, including their brief descriptions.

- **HOOK=(**n,n,...**)** specifies where in the VSAM R/M code tracing is to occur. The default is HOOK=(1). HOOK is an optional subparameter.

  Table 55 lists the predefined trace point IDs, their associated modules, and their functions.

Table 55. Predefined trace IDs, modules, and functions. Predefined trace IDs, modules, and functions

Trace point ID	Module	Description
0000	IDA019R1	Entry to VSAM.
0001	IDA019R1	Exit from VSAM.
0002	IDAM19R3	Prior to SVC 121 for writes of CIs (no reads).
0003	IDA019RZ	After I/O, wait for CI reads and writes.
0004	IDA019SE	Prior to call to EOV (SVC 55).
0005	IDA019SE	After return from EOV.
0006	IDA019RE	Start of a CI split.
0007	IDA019RE	After completion of a CI split.
0008	IDA019R3	All I/O occurring during a CI split.
0009	IDA019RF	Start of a CA split.
0010	IDA019RF	After completion of a CA split.
0011	IDA019R3	All I/O occurring during a CA split.
0012	IDA019RJ	Prior to index CI split (IDA019RJ entry).

Table 55. Predefined trace IDs, modules, and functions (continued). Predefined trace IDs, modules, and functions

Trace point ID	Module	Description
0013	IDA019RI	After call to IDA019RJ (index split).
0014	IDA019RU	After completion of an upgrade request.
0015	IDA019RW, IDA019SY	Shared resources—after I/O, no errors.
	IDA019RZ	Non-shared resources—after I/O, no errors.
0016	IDA019RW, IDA019SY	Shared resources—after I/O, error occurred.
	IDA019RZ	Non-shared resources—after I/O, error occurred.
0017	IDA019RP	After return from JRNAD exit.
0018	IDA019S7	Before SVC 109 call to update the VSI block.
0019	IDA019S7	Before control blocks are updated from VSI.
0020	IDA019R4	Before data record is compressed.
0021	IDA019R4	After data record is compressed.
0022	IDA019R4	Before data record is decompressed.
0023	IDA019R4	After data record is decompressed.
0024	IDAM19R3	Prior to SVC 121 for reads of CIS.
0025 through 0099		Reserved.
0100 through 0255		User trace points.

Note:

1. There is no limit to the number of trace points you can specify to trace.
2. The HOOK subparameter must be enclosed in parentheses, even if only one trace point ID is specified (for example, HOOK=(1)).

- **ECODE=ANY|**codenumber Limits tracing. When used, tracing occurs only if an error code is being returned to the caller. If ANY is specified, tracing is performed for any nonzero return code. If codenumber (a specific error code) is provided, tracing occurs only if the RPLFDBK code matches that error code.

  ECODE is an optional subparameter. If ECODE is not used, the RPLFDBK code will not determine if tracing is to occur.

  When an error code (codenumber) is given, it must be a positive decimal number. For example, ECODE=12 causes tracing when one of the following situations occurs:

  – A buffer needs to be written
  – An attempt was made to store a record out of ascending order sequence
  – A physical read error occurred for an index component sequence set.

  The three error situations are indicated by the RPLFDBK code of X'0C' and the content of register 15 which is 0, 8, or 12, respectively.

  Note: If the user's LERAD or SYNAD exit routine resets the return code before VSAM returns to the caller, this exit routine may fail. Its failure depends on which trace point is active, and when the call to the user's exit routine is made. See *z/OS DFSMS Installation Exits* for more information on user exit routines.

- **KEY=**keydata | lowRBA-highRBA Limits tracing. When used, tracing only occurs if the record key matches keydata, or if the record's RBA value is within the range of the lowRBA and highRBA values. Keydata is the EBCDIC representation of the whole key or the first few characters of a range of keys. KEY is an optional subparameter.

  If **KEY=**keydata is specified, the keydata may be any length up to 44 bytes. The keydata value does not need to be the same length as the record's key length. The shorter key length is used to determine the amount of bytes to be compared; this allows you to use generic key values and specify a range of keys.

  If **KEY=**lowRBA-highRBA is specified, tracing occurs only if the RBA of the record being processed is within the range of lowRBA and highRBA values.

  PARM1, byte 5 bit 5 determines the value of the KEY. If this bit is 0, the KEY field contains a key value; if this bit is 1, the KEY field contains an 8-byte lowRBA value, a dash, and an 8-byte highRBA value.

  Note: The KEY subparameter must not contain quotes, commas, or parentheses.
- **PARM1=**trace options

  PARM1 controls the tracing of any user-opened data sets.

  The PARM1 subparameter specifies which VSAM Record Management data areas are to be traced. It controls the tracing of the user-opened data sets.

  PARM1 is required. If TRACE is specified without any subparameters, only VSAM Open/Close/EOV GTF records are built.

  For an extended format data set, the CPA is not a valid control block to trace. If specified, the CPA will be ignored.

  For a compressed data set, the buffers might contain data in a compressed format.

  The PARM1 value must be entered in hexadecimal. Each bit in the subparameter represents a trace option. If the bit is active (1), the corresponding control block is traced or, in the case of bits in byte 5, the corresponding condition is taken. The bits are as follows:

Table 56. PARM1 subparameter bits, byte 0. PARM1 subparameter bits, byte 0

Byte 0	Notes	Description
1...	1	ABP–actual block processor.
.1..	1	ACB–access method control block.
..1.		AMB–access method block.
...1	1	AMBL–access method block list.
.... 1...		AMBXN–access method block extension.
.... .1..		AMDSB–access method data set statistics block.
.... ..1.	1	ARDB–address range definition block.
....1	1	BIB–base information block.

Table 57. PARM1 subparameter bits, byte 1. PARM1 subparameter bits, byte 1

Byte 1	Notes	Description
1...	2	BSPH–buffer subpool header.
.1..	2	BUFC–buffer control block.
..1.	1	CMB–cluster management block.
...1	7	CPA–channel program area.
.... 1...	1	CSL–core save list.

Table 57. PARM1 subparameter bits, byte 1 (continued). PARM1 subparameter bits, byte 1

Byte 1	Notes	Description
.... .1..		DIWA–data insert work area.
.... ..1.	1	EDB–extent definition block.
....1	1	HEB–header element block.

Table 58. PARM1 subparameter bits, byte 2. PARM1 subparameter bits, byte 2

Byte 2	Notes	Description
1...		ICWA–index create work area.
.1..		IICB–ISAM interface control block.
..1.		IMWA–index modification work area.
...1		IOMB–I/O management block.
.... 1...	7	IOSB–I/O supervisor block.
.... .1..		IXSPL–index search parameter list.
.... ..1.	1	LPMB–logical-to-physical mapping block.
....1	2	PLH–placeholder.

Table 59. PARM1 subparameter bits, byte 3. PARM1 subparameter bits, byte 3

Byte 3	Notes	Description
1...		RPL–request parameter list.
.1..		SRA–sphere record area.
..1.		UPT–upgrade table.
...1	1	VAT–valid AMBL table.
.... 1...	1	VMT–volume mount table.
.... .1..		VSI–VSAM shared information block.
.... ..1.	1	VVT–VSRT vector table.
....1		VSRT–VSAM shared resources table.

Table 60. PARM1 subparameter bits, byte 4. PARM1 subparameter bits, byte 4

Byte 4	Notes	Description
1...		WAX–work area for path processing.
.1..		WSHD–working storage header.
..1.	2	Buffers.
...1		User's search argument or key
.... 1...		User's record.
... .1..		Caller's registers.
... ..1.	6	Compression blocks
... ...x		Reserved.

Table 61. PARM1 subparameter bits, byte 5. PARM1 subparameter bits, byte 5

Byte 5	Notes	Description
1...		Do not trace data control blocks.
.1..		Do not trace index control blocks.

Table 61. PARM1 subparameter bits, byte 5 (continued). PARM1 subparameter bits, byte 5

Byte 5	Notes	Description
..1.	2	Trace all control blocks.
...1	1	Limit: one trace of control blocks.
.... 0...		KEY=keydata (the KEY contains a key value).
.... 1...		KEY=lowRBA-highRBA (the KEY contains RBA values).
.... .1..		Trace AIX, PATH, or UPGRADE processing (PARM2 required).
.... ..1.	3	Validity-check control blocks; trace if bad.
....x		Reserved.

- **PARM2=**trace options

  PARM2 controls the tracing of any VSAM-opened data sets.

  The PARM2 subparameter is used to control the tracing of:

  – An alternate index's base cluster when opened as a path

  – A base cluster's UPG (upgrade) data set.

  PARM2 is used only if PARM1, byte 5 bit 5 (X'04') is specified. It has the same options as PARM1's except for the last byte (byte 5), which Table 62 shows.

Table 62. PARM2 subparameter bits. PARM2 subparameter bits

Byte 5	Notes	Description
xxxx		Same usage as in PARM1.
.... *xx*..		Reserved.
.... ..1.	4	Trace all associated data sets.
....1	5	Trace UPGRADE control blocks.

Notes:

1. When **limit: one trace of control blocks** (byte 5 bit 3) is specified, the control blocks indicated with note 1 are traced only on the first call to R/M trace. These control blocks, generally, do not change after the data set is opened.
2. When **trace all control blocks** (byte 5 bit 2) is specified, the control blocks indicated with note 2 are traced even when the current request does not use them. If this bit is off, only those control blocks directly associated with the active request are traced.

   Attention: Turning this bit on can cause a large amount of GTF data, depending on the number of strings and buffers and the size of buffers.
3. This option causes R/M trace to validity-check the pointers in the VSAM R/M control blocks, and if a chaining error is detected, the trace is taken.
4. When byte 5 bit 6 of PARM2 is off, only the data set being processed when R/M trace was called is traced. When this bit is on, R/M trace locates and traces data sets associated with the calling data set.
   – If the calling or associated data set was user-opened, PARM1 is used.
   – If the calling or associated data set was VSAM-opened, PARM2 is used.
   – When byte 5 bit 7 of PARM2 is off, UPGRADE data sets are not traced unless the UPGRADE was the calling data set. When byte 5 bit 7 of PARM2 is on, R/M trace treats UPGRADES as associated data sets; they are traced when the calling data set is a path, alternate index, or base.
   – When byte 4 bit 6 of PARM1 is on, tracing of compression related control blocks will occur. This includes control blocks CMSP, CMWA, and PCB.

The data buffers for a compressed data set will not be traced unless both the buffers and compression block bits are set on.

– Tracing of the CPA and IOSB is ignored for data sets defined as extended format or compressed.

Example of a DD statement requesting trace:

```
//KSDS01 DD DSN=VSAM.DATA.SET,DISP=SHR,
//     AMP=('TRACE=(PARM1=F00203000010',
//     'HOOK=(1,5),KEY=C1C2C3C4F7F8,PARM2=F123456789AB)')
```

The **TRACE=** on this DD statement activates the R/M trace for **VSAM.DATA.SET**. The R/M trace records are written out when the following conditions are met:

1. The record being processed has a key that begins with the character string ABCD78 as specified by **KEY=C1C2C3C4F7F8**.
2. Record management returns to the caller (HOOK 1) or returns from a call to EOV (HOOK 5) as specified by **HOOK=(1,5)**.

When these conditions are met, R/M trace writes to GTF the following data as requested by **PARM1**: ABP, ACB, AMB, AMBL, EDB, LPMB, and PLH The 10 in byte 5 of **PARM1** requests the ABP, ACB, AMBL, EDB, LPMB be traced only on the first call to R/M trace. **PARM2** has no effect on tracing because PARM1, byte 5 bit 5 (trace AIX, PATH, or UPGRADE processing) was not specified.

Adding trace points: Several trace points are predefined into VSAM; each has a unique trace point ID.

If one of these predefined trace points (specifiable by HOOK) is insufficient, you can add your own trace point by providing the code, as follows:

```
VER nnnn  XXXX,XXXX                          ORIGINAL INSTRUCTION
VER nnn4  ????,???? MAIN                     NEXT SEQUENTIAL INSTRUCTION
VER pppp  ZZZZ,ZZZZ PATCH                    LOCATION OF PATCH
REP nnnn  47F0,pppp       BC    15,PATCH     BRANCH TO THE PATCH
REP pppp  BFFF,3078       ICM   15,15,AMBTRACE IS TRACE ACTIVE?
REP ppp4  4780,pp10       BZ    EXIT         NO, SKIP HOOK CODE
REP ppp8  58F0,F000       L     15,0(15)     LOAD TRACE ADDRESS
REP pppC  05EF            BALR  14,15        GO TO TRACE
REP pppE  0064            DC    X'0064'      ANY USER'S TRACE POINT ID
REP pp10  XXXX,XXXX EXIT                     ORIGINAL INSTRUCTION
REP pp14  47F0,nnn4       B     MAIN         RETURN TO NEXT SEQUENTIAL
                                             INSTRUCTION
```

You must ensure that the following registers contain the indicated data:

- Register 2 Address of the PLH
- Register 3 Address of the AMB (data or index)
- Register 15 Address of IDA019ST

Failure to have these registers set could cause program checks and other unpredictable results. Register 0 is altered during the execution of trace routine. Care must be used in the selection of trace points, and in the selection of control blocks to be traced, because not all control blocks may be valid at a given time.

Ending the record management trace function: The tracing of a data set terminates when that data set is closed. If you want to terminate the tracing before closing the data set, you must stop GTF. You can then restart GTF, but if you specify USR or USRP with FF5, R/M tracing will resume.

Printing the record management trace output: VSAM trace records are formatted and printed with IPCS. "Printing GTF records" on page 314 gives some examples of this service. For information on how to use IPCS, see *z/OS MVS IPCS User's Guide*. For information on the command syntax, see *z/OS MVS IPCS Commands*.

VSAM record management return and reason codes

VSAM provides return codes and reason codes to indicate the results of macro calls.

The return codes listed in this topic are also applicable when RLS is active.

Return codes from record-management (request) macros are set in register 15; reason codes are set in the RPL RPLFDBWD.

VSAM control block manipulation macros set return codes in registers 15 and 0.

Return codes and reason codes from VSAM macros can also be found in "Understanding VSAM macro return and reason codes" on page 190.

Return codes from the record-management (request) macros

After a request macro or a CHECK or ENDREQ macro is issued, register 15 contains a return code.

After an asynchronous request to access a data set, VSAM indicates in register 15 whether the request was accepted, as Table 63 shows.

Table 63. Return codes from the record-management (request) macros - asynchronous requests. Return codes from the record-management (request) macros - asynchronous requests

Return code	Description
0(X'00')	Request was accepted.
4(X'04')	Request was not accepted because the request parameter list indicated by the request (RPL=address) was active for another request.
16(X'10')	• SMSVSAM server is not active • invalid OPEN connection

After a synchronous request, or a CHECK or ENDREQ macro, register 15 indicates whether the request was completed successfully, as Table 64 shows.

Table 64. Return codes from the record-management (request) macros - synchronous requests. Return codes from the record-management (request) macros - synchronous requests

Return code	Description
0(X'00')	Request completed successfully.
4(X'04')	Request was not accepted because the request parameter list indicated by the request (RPL=address) was active for another request.
8(X'08')	Logical error; specific error is indicated in the feedback field in the RPL.
12(X'0C')	Physical error; specific error is indicated in the feedback field in the RPL.

Table 64. Return codes from the record-management (request) macros - synchronous requests (continued). Return codes from the record-management (request) macros - synchronous requests

Return code	Description
16(X'10')	• The SMSVSAM server is not active. • The OPEN connection is invalid. • DFSMStvs is not active.

The feedback area of the request parameter list (RPL) contains additional diagnostic information that is used with the return codes in register 15 to determine the cause of an error.

The feedback area in the RPL is a fullword field:

- Byte0 Problem determination code
- Byte1 RPL return code (same as register 15)
- Byte2 Component code
- Byte3 Reason code

VSAM does not branch to an exit routine when register 15 is 0 on return from a request. The list in Table 65 describes the reason codes that might be in the RPL feedback area when register 15 is 0.

Table 65. Return codes from the record-management (request) macros - R15=0. Return codes from the record-management (request) macros - R15=0

RPLRTNCD code	Condition
0(X'00')	Request completed successfully.
4(X'04')	Request completed successfully. For retrieval, VSAM mounted another volume to locate the record; for storage, VSAM allocated additional space or mounted another VSAM EOV was called.
8(X'08')	For GET requests, indicates that a duplicate key follows; for PUT requests, indicates that a duplicate key was created in an alternate index with the nonunique attribute.
12(X'0C')	(Shared resources only.) A buffer needs to be written.
16(X'10')	Control area split was required because a sequence set control interval had free space insufficient to contain the key to be inserted.
24(X'18')	Buffer found but not modified: no buffer writes performed.
28(X'1C')	A CI split for the CI was interrupted. The CI was read as non-update with address access. This warning condition indicates that duplicate data records may exist. The RBA for this CI can be acquired from RPLDDDD for non-extended-addressable data sets or from the lower six bytes of RPLRBAR for an extended-addressable data set.
32(X'20')	Possible causes: • Request deferred for a resource held by the terminated RPL is asynchronous and cannot be restarted by TERMRPL. • A MRKBFR request is invalid because no candidate buffer could be found. • For IDARETLK, there were no locks to retain since no update locks exist for this SUBSYSNM/LUWID/SPHERE.

Table 65. Return codes from the record-management (request) macros - R15=0 (continued). Return codes from the record-management (request) macros - R15=0

RPLRTNCD code	Condition
36(X'24')	Possible data set error condition was detected by TERMRPL: 1. The request was abnormally terminated in the middle of its I/O operation. 2. One of the data/index BUFCs of the string contains data that needs to be written (BUFCMW=ON), but it was invalidated by TERMRPL.
40(X'28')	Error in PLH data BUFC pointer was detected by TERMRPL.
43(X'2B')	EOV called to retrieve or update the dictionary token in the catalog for a compressed data set.
44(X'2C')	EOV called to update catalog statistics.

See the following discussions for the logical-error and physical-error return codes.

Function codes for logical and physical errors: When a logical or physical error occurs during processing that involves alternate indexes, VSAM provides a code in the RPLCMPON field that indicates whether the base cluster, its alternate index, or its upgrade set was being processed and whether upgrading was satisfactory or may have been incorrect because of the error (see Table 66).

Table 66. Function codes for logical and physical errors. Function codes for logical and physical errors

Code	What was being processed	Status of upgrading
0(X'00')	Base cluster	Satisfactory.
1(X'01')	Base cluster	Might be incorrect.
2(X'02')	Alternate index	Satisfactory.
3(X'03')	Alternate index	Might be incorrect.
4(X'04')	Upgrade set	Satisfactory.
5(X'05')	Upgrade set	Might be incorrect.

Logical-error return codes: When a logical-error-analysis exit routine (LERAD) is provided, it gets control for logical errors, and register 15 does not contain 8, but contains the entry address of the LERAD routine.

Table 67 gives the contents of the registers when VSAM exits to the LERAD routine.

Table 67. Contents of registers when a LERAD routine gets control. Contents of registers when a LERAD routine gets control

Register	Contents
0	Unpredictable.
1	Address of the request parameter list that contains the feedback field the routine should examine. The register must contain this address if the exit routine returns to VSAM.
2-13	Same as when the request macro was issued. Register 13, by convention, contains the address of the processing program's 72-byte save area, which must not be used as a save area by the LERAD routine if the routine returns control to VSAM.

Table 67. Contents of registers when a LERAD routine gets control (continued). Contents of registers when a LERAD routine gets control

Register	Contents
14	Return address to VSAM.
15	Entry address to the LERAD routine. The register does not contain the logical-error indicator.

If a logical error occurs and a LERAD exit routine is not provided (or the LERAD exit is inactive), VSAM returns control to the processing program following the last executed instruction. Register 15 indicates a logical error (8), and the feedback field in the request parameter list contains a code identifying the error. Register 1 points to the request parameter list.

See *z/OS DFSMS Installation Exits* for additional information on the LERAD exit routine.

Table 68 gives the logical-error return codes in the feedback field and explains what each means.

Table 68. Logical-error reason codes found in the RPL feedback field when R15=8 and LERAD is not active.. Logical-error reason codes found in the RPL feedback field when R15=8 and LERAD is not active.

RPLERRCD code	Symbol	Condition
4(X'04')	RPLEODER	End of data set encountered (during sequential retrieval). Either no EODAD routine is provided, or one is provided and it returned to VSAM and the processing program issued another GET.
8(X'08')	RPLDUP	Attempt was made to store a record with a duplicate key.
12(X'0C')	RPLSEQCK	Attempt was made to store a record out of ascending key sequence; record may also have a duplicate key.
16(X'10')	RPLNOREC	Record not found.
20(X'14')	RPLEXCL	• For MACRF=RLS, this code means that there was an intra-LUDWID exclusive control conflict. • For non-RLS, the record is already held in exclusive control by another requester.
21(X'15')		For MACRF=RLS, the request was rejected due to deadlock; the deadlock was resolved by rejecting this request.
22(X'16')		For MACRF=RLS, the request was rejected due to timeout. The request was presumed to be an inter resource manager deadlock.
24(X'18')	RPLNOMNT	• For MACRF=RLS, lock request rejected because the lock is held by a failed sybsystem. • For non RLS, the record resides on a volume that cannot be mounted.
28(X'1C')	RPLNOEXT	Data set cannot be extended because VSAM cannot allocate additional direct-access storage space. Either there is not enough space left in the data space for the secondary-allocation request or an attempt was made to increase the size of a data set by splitting the control area (high used RBA change) during processing with SHROPT=4 and DISP=SHR.
32(X'20')	RPLINRBA	An RBA was specified that does not give the address of any data record in the data set. This error will be returned by register 10 and register 8 if the RBA provided addresses a relative CI greater than 4GB.
36(X'24')	RPLNOKR	Key ranges were specified for the data set when it was defined, but no range was specified that includes the record to be inserted.

Table 68. Logical-error reason codes found in the RPL feedback field when R15=8 and LERAD is not active. (continued). Logical-error reason codes found in the RPL feedback field when R15=8 and LERAD is not active.

RPLERRCD code	Symbol	Condition
40(X'28')	RPLNOVRT	Insufficient virtual storage in the address space to complete the request. Or there is insufficient storage available to add another string dynamically. For DFSMStvs, this indicates that DFSMStvs was unable to expand the pool for its context or unit of recovery-related control blocks.
44(X'2C')	RPLINBUF	Work area not large enough for the data record (GET with OPTCD=MVE).
48(X'30')	RPLINTRM	Invalid options, data set attributes, or processing conditions specified for TERMRPL request: • CNV processing • The specified RPL is asynchronous • Chained RPLs • PATH processing • Shared resources (LSR/GSR) • Create mode • RRDS • Data set contains spanned records • User not in Key 0 and supervisor state • EOV in process (secondary allocation).
52(X'34')	RPLPTERM	The previous request was TERMRPL.
56(X'38')	RPLCTERR	• For MACRF=RLS, RPL reuse violation. The RPL request contained positioning information from a previous request and the ACB or LUDWID specified in the RPL did not match the previous ACB or LUDWID. • For non RLS, error from catalog update at the beginning of a CI/CA split for a backup while opening a data set.
60(X'3C')	RPLCTERR	Available for RLS and DFSMStvs use.
64(X'40')	RPLNOPLH	• For RLS and DFSMStvs, the limit of 1024 outstanding requests for this ACB has been exceeded. • As many requests are active as the number specified in the STRNO parameter of the ACB macro; therefore, another request cannot be activated. • Or there is insufficient storage available to add another string dynamically.
68(X'44')	RPLINACC	Attempt was made to use a type of processing (output or control-interval processing) that was not specified when the data set was opened.
72(X'48')	RPLINKEY	• For RLS and DFSMStvs, add GETIX or PUTIX to any data set organization. • For non-RLS or DFSMStvs, one of the following items might apply: – Keyed GET request for an ESDS – GETIX or PUTIX to ESDS – Fixed length RRDS
76(X'4C')	RPLINADR	• An ADR or CI PUT was issued to add a record to a KSDS or VRRDS • Or a CI PUT was issued to a fixed length RRDS.
80(X'50')	RPLERSER	An ERASE request was issued for access to an ESDS or CI.
84(X'54')	RPLINLOC	• OPTCD=LOC was specified for a PUT request or in the *previous* request parameter list, in a chain of parameter lists. • LOC not supported (RLS only)

Table 68. Logical-error reason codes found in the RPL feedback field when R15=8 and LERAD is not active. (continued). Logical-error reason codes found in the RPL feedback field when R15=8 and LERAD is not active.

RPLERRCD code	Symbol	Condition
88(X'58')	RPLNOPTR	A sequential GET or PUT request was issued without VSAM having been positioned for it, or a change was made from addressed access to keyed access without VSAM having been positioned for keyed sequential retrieval, or an illegal switch between forward and backward processing was attempted.
92(X'5C')	RPLINUPD	A PUT, ERASE, or IDALKCD was issued without a previous GET for UPDATE. Or a PUTIX was issued without a previous GETIX.
96(X'60')	RPLKEYCH	• For RLS and DFSMStvs, a PUT NUP attempt was made to change the key specified by a previous IDALKADD request. • For non-RLS or DFSMStvs, an attempt was made to change the prime key or the reference key during an update.
100(X'64')	RPLDLCER	Attempt was made to change the length of a record during an addressed update.
104(X'68')	RPLINVP	The RPL options are either invalid or conflicting in one of the following ways: • SKP was specified and either KEY was not specified or BWD was specified. XRBA was not specified in the RPL OPTCD when a GET DIR or POINT REQUEST was issued in ADR|CNV mode with LRD=OFF and PLARG points to a non-zero argument (RBA), during the processing of an EA data set. • BWD was specified for CNV processing. • FWD and LRD were specified. • Neither ADR, CNV, nor KEY was specified in the RPL. • BFRNO is invalid (less than 1 or greater than the number of buffers in the pool). • WRTBFR, MRKBFR, or SCHBFR was issued, but either TRANSID was greater than 31 or a shared-resources option was not specified. • ICI processing was specified, but a request other than a GET or a PUT was issued. CNV processing for a compressed data set was specified. Only VERIFY and VERIFY REFRESH are allowed. For RLS and DFSMStvs only, • IDARECOV, IDALKREL, IDAINQRC, IDARETLK, or IDAQUIES issued and ACB is not an RLS Control ACB. • A Record Management GET or PUT was issued against an RLS Control ACB. • No LUWID specified for IDALKREL, IDARETLK, or IDALKCD. • SUBSYSNM was specified in ACB at OPEN, but GET, IDALKADD, PUT, POINT, or ERASE specified LUWID=0. • RLS options (LUWID, read integrity options, cold start) are not supported for NSR/GSR/LSR access. • CNV access specified for RLS access or an extended function data set. • ADR access to KSDS for RLS. • Invalid record management request issued against a control ACB. • Non-commit protocol application specified: – CRE on POINT or GET NUP – KL on GET UPD to a recoverable sphere • Invalid RLS request: LSR or MSS request macros, GETIX, or PUTIX. • IDAEADD is invalid for KSDS, RRDS, or VRRDS.

Table 68. Logical-error reason codes found in the RPL feedback field when R15=8 and LERAD is not active. (continued). Logical-error reason codes found in the RPL feedback field when R15=8 and LERAD is not active.

RPLERRCD code	Symbol	Condition
108(X'6C')	RPLINLEN	RECLEN specified was larger than the maximum allowed, equal to 0, smaller than the sum of the length and the displacement of the key field, or not equal to record (slot) length specified for a relative record data set.
112(X'70')	RPLKEYLC	KEYLEN specified was too large or equal to 0.
116(X'74')	RPLINLRQ	• Available for RLS and DFSMStvs. • For non RLS, invalid request during load mode. A GET, POINT, ERASE, direct PUT, skip sequential PUT, or PUT with OPTCD=UPD not permitted during initial data set loading (that is, for storing records in the data set the first time it is opened).
120(X'78')	RPLINTCB	• Current job step TCB is not correct one. • For RLS, request issued in cross-memory mode.
124(X'7C')	RPLUEXCL	• Available for RLS and DFSMStvs. • For an application that does not use RLS or DFSMStvs, a request was canceled from a user JRNAD exit.
128(X'80')	RPLIXHHP	Index is invalid, request cannot be completed.
132(X'84')	RPLSRLOC	• Available for RLS and DFSMStvs. • For an application that does not use RLS or DFSMStvs, an attempt was made in locate mode to retrieve a spanned record.
136(X'88')	RPLARSRK	• Available for RLS and DFSMStvs. • For an application that does not use RLS or DFSMStvs, an addressed GET was issued for a spanned record in a key-sequenced data set.
140(X'8C')	RPLSRISG	Inconsistent spanned-record segments.
144(X'90')	RPLNBRCD	Invalid pointer in an alternate index (no associated base record).
148(X'94')	RPLNXPTR	The maximum number of pointers in the alternate index has been exceeded.
152(X'98')	RPLNOBFR	• For LSR, RLS, and DFSMStvs, not enough buffers are available to process the request. • For RLS and DFSMStvs, the dataspace buffer pool was exhausted.
156(X'9C')	RPLINCNV	An invalid control interval was detected during keyed processing. The possible invalid conditions are: 1. A key is not greater than the previous key. 2. A key is not in the current control interval. 3. A spanned record RDF is encountered. 4. A freespace pointer is invalid. 5. The number of records does not match a group RDF record count.
160(X'A0')	RPLBMWER	A request was issued to invalidate a modified buffer. For RLS, the required quiesce exit does not exist.
161(X'A1')	RPLQCLRJ	QUICLOSE request is rejected because the sphere is already marked quiesced.

Table 68. Logical-error reason codes found in the RPL feedback field when R15=8 and LERAD is not active. (continued). Logical-error reason codes found in the RPL feedback field when R15=8 and LERAD is not active.

RPLERRCD code	Symbol	Condition
162(X'A2')	RPLQUIRJ	The quiesce status of the sphere means this IDAQUIES request cannot be accepted from this application. Issued for the following: • This is a QUICMP request and field QUIERTOK in the IFGQUIES parameter area does not contain a request token corresponding to a QUIESCE exit invocation that VSAM is waiting for, to get a QUICMP response from this application. • This QUICEND or QUIBEND request is rejected because the application signalled completion of its processing for this QUICOPY or QUIBWO event. • This QUICEND or QUIBEND request is rejected because the applications' QUIESCE exit was not driven for this QUICOPY or QUIBWO event.
163(X'A3')	RPLQACBO	IDAQUIES type QUICMP request rejected. ACBs for the sphere remain open for the application and this is an IDAQUIES type QUICLOSE.
164(X'A4')	RPLQCNCL	IDAQUIES request did not complete successfully. The request is canceled.
165(X'A5')		For RLS and DFSMStvs, either the IDARECOV request was specified as TYPE=LL and the sphere was not in lost locks state for this subsystem, or TYPE=NONRLS was specified and the sphere was not in NONRLSUPDATE permitted state.
166(X'A6')		Available for RLS and DFSMStvs.
167(X'A7')	RPLQNSUP	The field QUIESTYP in the IFGQUIES parameter area specifies an invalid request type, or the eyecatcher in IGFQUIES is invalid.
168(X'A8')		For RLS and DFSMStvs, the RPLAREA was 0.
169(X'A9')	RPLQCTGF	For RLS and DFSMStvs, the IDAQUIES, IDARETLK TYPE=SS, IDARECOV TYPE=LL, or IDARECOV TYPE=NONRLS request failed because the Catalog Locate command issued for the specified sphere or component name failed.
170(X'AA')	RPLQUNFL	The QUIOPEN, QUICEND, or QUIBEND request is rejected because the requested unquiesce operation is already started for this sphere.
172(X'AC')	RPLACQER	For RLS and DFSMStvs, the IDAQUIES, IDARETLK TYPE=SS, IDARECOV TYPE=LL, or IDARECOV TYPE=NONRLS request failed because the specified sphere is not an SMS VSAM data set.
176(X'B0')	RPLSTGER	For RLS and DFSMStvs, the ACB specified in the IDARETLK TYPE=SS, IDARECOV TYPE=LL, or IDARECOV TYPE=NONRLS request is not a valid ACB open for RLS or DFSMStvs processing to the sphere.
180(X'B4')		For RLS, an invalid request for a nonrecoverable data set.
181(X'B5')	RPLQRACF	This IDAQUIES request is rejected because the requestor does not have update authority for the sphere: • This is a type QUICLOSE request. Successful completion of the request results in a catalog update to mark the sphere quiesced. Because the requestor does not have update authority, the request is rejected. • The catalog shows that this sphere is quiesced. Successful completion of the QUIOPEN request would result in a catalog update to reset the quiesced state of the sphere. Because the requestor does not have update authority, the request is rejected.

Table 68. Logical-error reason codes found in the RPL feedback field when R15=8 and LERAD is not active. (continued). Logical-error reason codes found in the RPL feedback field when R15=8 and LERAD is not active.

RPLERRCD code	Symbol	Condition

182(X'B6')	RPLQINPR	For RLS, the IDAQUIES request rejected because an IDAQUIES is already in progress for this sphere. If an RPL message area (address in RPLERMSA) of sufficient length (specified in RPLEMLEN) is specified, the following information is returned:
		`Offset Length Description`
		`------ ------ -----------`
		`0 1 Type of quiesce event already in progress for this sphere. Quiesce type constants are defined in IFGQUIES macro.`

183(X'B7')	RPLMIGRA	IDAQUIES request rejected because data set is migrated. .
184(X'B8')	RPLABEND	For RLS, an ABEND condition occurred while processing this VSAM request. The VSAM RLS FRR (Functional Recovery Routine) intercepted the failure and failed the VSAM request with this reason code.
185(X'B9')		For RLS, the user task was canceled while the request was being processed.
186(X'BA')	RPLEOVER	For RLS, end-of-volume initialization failed when DATASET tried to extend.
187(X'BB')		For RLS, an error occurred with partial EOV processing.
188(X'BC')	RPLNO241	For RLS, the storage in subpool 241 is not available.
189(X'BD')		For RLS, a lock for the VSAM request required space in the record table, which is full. You must modify the CFRM policy and rebuild the lock structure.
192(X'C0')	RPLIRRNO	Invalid relative record number.
196(X'C4')	RPLRRADR	An addressed request was issued to a relative record data set.
200(X'C8')	RPLPAACI	Addressed or control-interval access was attempted through a path.
201(X'C9')		For RLS or DFSMStvs, the IDARETLK TYPE=SS request failed because the specified data set does not exist.
204(X'CC')	RPLPUTBK	PUT-insert requests (or for RLS, IDALKADD requests) are not allowed in backward mode.
205(X'CD')		• DFSMStvs was unable to complete the request because DFSMStvs restarted while the unit of recovery was inflight. To continue processing, the application must issue a commit or a backout, then begin a new unit of recovery. • For LSR, invalid CONTOKEN.
206(X'CE')		• For DFSMStvs, indicates that the request was rejected because the data set is quiesced or quiescing for copy. Retry the request. • For applicaitons that do not use RLS or DFSMStvs, this is a validity check error for share 3,4.
207(X'CF')		For DFSMStvs, indicates that transactional processing is currently unavailable because DFSMStvs is disabling or quiescing. Close all data sets so the process can complete.
208(X'D0')	RPLINVEQ	Invalid ENDREQ request.
209(X'D1')		• For DFSMStvs, indicates that the forward recovery log is unavailable because it is disabling. • For LSR, indicates a cache structure failure.

Table 68. Logical-error reason codes found in the RPL feedback field when R15=8 and LERAD is not active. (continued). Logical-error reason codes found in the RPL feedback field when R15=8 and LERAD is not active.

RPLERRCD code	Symbol	Condition
210(X'D2')		• For DFSMStvs, indicates that forward recovery logging failed because the record length is greater than the installation maximum supported by the log. • For shared resources, the buffer is being invalidated, or the buffer use chain changing.
211(X'D3')		• For DFSMStvs, indicates that a permanent I/O error was detected in the forward recovery log. For appropriate action, see the accompanying DFSMStvs logger messages. • For LSR, indicates that the cache request was purged.
212(X'D4')	RPLNOSPL	Unable to split index during a CA split.
213(X'D5')		• For DFSMStvs, indicates that the undo log is unavailable for processing. • For LSR, indicatesw no connectivity to the cache structure.
214(X'D6')		For DFSMStvs, indicates that a permanent I/O error was detected in the undo log. For appropriate actions, see the accompanying DFSMStvs logger messages.
216(X'D8')		For RLS, the LUWID specified in the RPL does not exist for the subsystem name specified in the ACB.
217(X'D9')		DFSMStvs is unable to complete the request because RRS, which had been available, went down and restarted. To continue processing, the application must issue a commit or a backout and then begin a new unit of recovery.
218(X'DA')		Unrecognizable return code from SVC 109.
220(X'DC')		For DFSMStvs, this reason code is no longer being used.
224(X'E0')	RPLMOIB	A MRKBFR request was issued for an invalid buffer.
228(X'E4')	RPLINVMD	A cross-memory caller is not in supervisor state, in SRB, or in cross-memory mode, or callers of RPL do not specify SYN processing.
229(X'E5')	RPLDELCH	The record length changed during decompression processing.
232(X'E8')	RPLUPERR	A cross-memory mode caller did not post the ECB in the UPAD exit routine.
235(X'EB')	RPLBMWER	• If the SMSVSAM server is not available, the IDAQUIES request fails with R15=16. • For RLS and DFSMStvs, this is an internal error.
236(X'EC')	RPLINVSI	Validity check error from SVC 109 for share option 3 or 4.
240(X'F0')	RPLUSTAT	Buffer pool status is unknown. The buffer use chain may be changing or a buffer is being modified or invalidated. Reissue the request.
244(X'F4')	RPLSVR14	Register 14 stack size is not large enough.
245(X'F5')	RPLCMSCE	Severe error returned from CMS for a compress call.
246(X'F6')	RPLCMSDE	Severe error returned from CMS for a decompress call.
247(X'F7')		Error in the last active record number (DFM).
248(X'F8')	RPLRST14	Register 14 return offset is negative.
249(X'F9')		• For DFSMStvs, indicates that undo logging failed because the record length is greater than the installation maximum supported by the log. • For LSR XI, indicates an invalid vector token.
250(X'FA')	RPLINVDT	No valid directory token exists. The data set cannot be decompressed.
251(X'FB')	RPLBMWER	Internal VSAM error.
252(X'FC')	RPLER252	Record-mode access is not valid for a linear data set.

Table 68. Logical-error reason codes found in the RPL feedback field when R15=8 and LERAD is not active. (continued). Logical-error reason codes found in the RPL feedback field when R15=8 and LERAD is not active.

RPLERRCD code	Symbol	Condition
253(X'FD')	RPLER253	Verify function is not valid for a linear data set.
254(X'FE')	RPLERQUS	I/O activity on the data set was not quiesced before WTBFR TYPE=DS was issued.

Table 69. Reason codes associated with R15=12. Reason codes associated with R15=12

RPLRTNCD code	Condition
4(X'04')	Read error for data component.
8(X'08')	Read error for index component.
12(X'0C')	Read error for sequence set.
16(X'10')	Write error for data component.
24(X'18')	Write error for sequence set.
28(X'1C')	Available.
32(X'20')	Available.
36(X'24')	For RLS, coupling facility cache connectivity loss.
40(X'28')	For RLS, coupling facility cache structure failure.
44(X'2C')	For extended function data sets, the suffix for a physical record in the CI, at the RBA specified in the RPL, is invalid.

Table 70. Reason codes associated with R15=16. Reason codes associated with R15=16

RPLRTNCD code	Condition
12(X'0C')	DFSMStvs processing is currently unavailable because DFSMStvs is initializing.

Physical-error return codes: When a physical-error-analysis exit routine (SYNAD) is provided, it gets control for physical errors, and register 15 does not contain 12, but contains the entry address of the SYNAD routine.

For additional information on the SYNAD exit routine, see *z/OS DFSMS Installation Exits*.

Table 71 gives the contents of the registers when VSAM exits to the SYNAD routine.

Table 71. Contents of registers when a SYNAD routine gets control. Contents of registers when a SYNAD routine gets control

Register	Contents
0	Unpredictable.
1	Address of the request parameter list that contains a feedback return code and the address of a message area, if any. If a request macro was issued, the RPL is the one pointed to by the request macro; if a CLOSE macro was issued, the RPL was built by VSAM to process the close request. Register 1 must contain this address if the exit routine returns to VSAM.

Table 71. Contents of registers when a SYNAD routine gets control (continued). Contents of registers when a SYNAD routine gets control

Register	Contents
2-13	Same as when the request macro or CLOSE macro was issued. Register 13, by convention, contains the address of the processing program's 72-byte save area, which may not be used by the SYNAD routine if it returns control to VSAM.
14	Return address to VSAM.
15	Entry address to the SYNAD routine. The register does not contain the physical-error indicator. **Note:** The SYNAD exit, like other user exits, might not return control to VSAM.

If a physical error occurs and a SYNAD exit routine is not provided (or the SYNAD exit is inactive), VSAM returns control to the processing program following the last executable instruction. Register 15 indicates a physical error (12), and the feedback field in the request parameter list contains a code identifying the error. Register 1 points to the request parameter list.

Table 72 gives the physical-error return codes in the feedback field and explains what each indicates. If the user provided a message area, it contains a physical-error message with more details about the error.

Table 72. Physical-error reason codes in the RPL feedback field from a request macro. Physical-error reason codes in the RPL feedback field from a request macro

RPLERRCD code	Symbol	Condition
4(X'04')	RPLRDERD	Read error occurred for a data component.
8(X'08')	RPLRDERI	Read error occurred for the index set of an index component.
12(X'0C')	RPLRDERS	Read error occurred for the sequence set of an index component.
16(X'10')	RPLWTERD	Write error occurred for a data component.
20(X'14')	RPLWTERI	Write error occurred for the index set of an index component.
24(X'18')	RPLWTERS	Write error occurred for the sequence set of an index component.
		All physical errors are detected by IDA019R5 from I/O Management abnormal-end appendage, IDA121A4.

Table 73 gives the format of a physical-error message. The format and some of the contents of the message are purposely similar to the format and contents of the SYNADAF message, which is described in *z/OS DFSMS Macro Instructions for Data Sets*.

Table 73. Format of physical-error messages. Format of physical-error messages

Field	Bytes	Length	Discussion
Message Length	0-1	2	Binary value of 128
	2-3	2	Unused (0)

Table 73. Format of physical-error messages (continued). Format of physical-error messages

Field	Bytes	Length	Discussion
Message Length-4	4-5	2	Binary value of 124 (provided for compatibility with SYNADAF message)
	6-7	2	Unused (0)
Address of I/O Buffer	8-11	4	The I/O buffer associated with the data in relation to which the error occurred
The rest of the message is in printable format:			
Date	12-16	5	YYDDD (year and day)
	17	1	Comma (,)
Time	18-25	8	HHMMSSTH (hour, minute, second, and tenths and hundredths of a second).
	26	1	Comma (,)
RBA	27-38	12	Relative byte address of the record in relation to which the error occurred.
	39	1	Comma (,)
Data-Set Type	40	1	'D' for data or 'I' index
	41	1	Comma (,)
Volume Serial Number	42-47	6	Volume serial number of the volume in relation to which the error occurred.
	48	1	Comma (,)
Job Name	49-56	8	Name of the job in which error occurred.
	57	1	Comma (,)
Step Name	58-65	8	Name of the job step in which error occurred.
	66	1	Comma (,)
Unit	67-70	4	Device number on which the error occurred.
	71	1	Comma (,)
Device Type	72-73	2	The type of device in relation to which the error occurred (always DA for direct access).
	74	1	Comma (,)
ddname	75-82	8	The ddname of the DD statement defining the data set in relation to which the error occurred.
	83	1	Comma (,)
Channel Command	84-89	6	The channel command that occasioned the error in the first two bytes, followed by '-OP'
	90	1	Comma (,)

Table 73. Format of physical-error messages (continued). Format of physical-error messages

Field	Bytes	Length	Discussion
Message condition codes:	91-105	15	Messages are divided according to ECB X'41'—'INCORR LENGTH' 'UNIT EXCEPTION' 'PROGRAM CHECK' 'PROTECTION CHK' 'CHAN DATA CHK' 'CHAN CTRL CHK' 'INTFCE CTRL CHK' 'CHAINING CHK' 'UNIT CHECK' 'SEEK CHECK'
If the type of the unit check can be determined, this message is replaced by one of the following:			
			'CMD REJECT' 'INT REQ' 'BUS OUT CK' 'EQP CHECK' 'DATA CHECK' 'OVER RUN' 'TRACK COND CK' 'SEEK CHECK' 'COUNT DATA CHK' 'TRACK FORMAT' 'CYLINDER END' 'INVALID SEQ' 'INVALID SUFFFIX' 'NO RECORD FOUND' 'FILE PROTECT' 'MISSING A.M.' 'OVERFL INCP' X'48'—'PURGED REQUEST' X'4A'—'I/O PREVENTED' X'4F'—'R.HA.RO. ERROR' For any other ECB completion code—'UNKNOWN COND'
	106	1	Comma (,)
Physical Direct-Access Address	107-120	14	BBCCHHR (bin, cylinder, head and record)
	121	1	Comma (,)
Access Method	122-127	6	'VSAM'

Control block manipulation return codes

When the control block manipulation routine returns to the caller after successful completion, register 15 contains 0. If the request is GENCB, register 0 contains the total length of the area that contains the control block(s). Register 1 contains the address of the area.

When the control block manipulation routine returns to the caller with a nonzero value in register 15, an error occurred. If the request is TESTCB and the caller supplied an ERET keyword, return is to the location specified by the ERET keyword. Otherwise, the control block manipulation routine returns control to the point of invocation, via the return address in register 14.

Register 15 contains a return code, which Table 74 explains.

Table 74. Control block manipulation return codes. Control block manipulation return codes

Reg 15	Condition
0(X'00')	Successful completion.
4(X'04')	An error has been detected. The error code in register 0 indicates the type of error.
8(X'08')	Invalid use of the execute form of this macro. Since the return code is set by the macro expansion and not by the control block manipulation routine, the register 0 contents do not indicate an error code.

Register 0 contains an error code, which Table 75 explains.

Table 75. Control block manipulation error codes. Control block manipulation error codes

Code	Applicable macros[1]	Condition
1(X'01')	G,M,S,T	The function type is invalid.
2(X'02')	G,M,S,T	The control-block type is invalid.
3(X'03')	G,M,S,T	The keyword type is invalid.
4(X'04')	M,S,T	The control block to be processed is not of the type specified.
5(X'05')	S,T	The ACB to be processed is closed—it must be open.
6(X'06')	S,T	The cluster whose index component was to be processed is not key-sequenced (does not include an index).
7(X'07')	M,S	The EXLST entry to be processed is not present.
8(X'08')	G	Not enough virtual storage is available, or (with AM=VTAM specified) list and execute forms are inconsistent.
9(X'09')	G,S	User area is too small.
10(X'0A')	G,M	Exit address is not specified in the input.
11(X'0B')	M	The RPL to be processed is active, or it is already being processed.
12(X'0C')	M	The ACB to be processed is open—it must be closed.
13(X'0D')	M	No exit address is specified in the input for the exit to be activated.
14(X'0E')	G,M,T	An invalid combination of option codes (for example, for MACRF or OPTCD) is specified.

Table 75. Control block manipulation error codes (continued). Control block manipulation error codes

Code	Applicable macros[1]	Condition
15(X'0F')	G,S	The user area is not on a fullword boundary.
16(X'10')	G,M,S,T	A VTAM keyword is specified with AM=VTAM not specified.
19(X'13')	M,S,T	A specified keyword refers to a field beyond the end of the control block to be processed.
20(X'14')	S	A specified keyword requires processing with shared resources to be specified, but it is not.
21(X'15')	S,T	The block to be displayed or tested does not exist, because the data set is a dummy data set.
22(X'16')	S	AM=VTAM is specified with SHOWCB for RPL fields=NIB, but the RPLNIB is off, or SHOWCB RPL fields=Arg (CID) but the RPLNIB bit is on.
23(X'17')	G	The value specified in the length parameter exceeds the 65535 byte limit. All errors in control block manipulation are detected by IDA019C1.
26(X'1A')	S	A request was made using a field name which allows the returned data to have a one word length, but the value being returned requires two words. A value of X'FFFFFFFF' is returned in place of the true value. Register 15 will contain a value of zero.

Note:

1. G=GENCB, M=MODCB, S=SHOWCB, T=TESTCB

Appendix. Accessibility

Accessible publications for this product are offered through the z/OS Information Center, which is available at www.ibm.com/systems/z/os/zos/bkserv/.

If you experience difficulty with the accessibility of any z/OS information, please send a detailed message to mhvrcfs@us.ibm.com or to the following mailing address:

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

Accessibility features

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

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

Using assistive technologies

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

Keyboard navigation of the user interface

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

Dotted decimal syntax diagrams

Syntax diagrams are provided in dotted decimal format for users accessing the z/OS Information 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 refer to separate syntax fragment OP1.

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

- ? 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.
- ! means a default syntax element. A dotted decimal number followed by the ! symbol and a syntax element indicates 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 only applies 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.

- * 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.

  Note:

  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 U.S.A. or elsewhere.

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

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

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

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

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

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

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

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

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

Site Counsel
IBM Corporation
2455 South Road
Poughkeepsie, NY 12601-5400
USA

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

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

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.

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

COPYRIGHT LICENSE:

This information might contain sample application programs in source language, which illustrate 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.

Policy for unsupported hardware

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

for service, and current service activity will cease if a problem is determined to be associated with out-of-support devices. In such cases, fixes will not be issued.

Minimum supported hardware

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

- For information about software support lifecycle, see: IBM Lifecycle Support for z/OS (http://www.ibm.com/software/support/systemsz/lifecycle/)
- For information about currently-supported IBM hardware, contact your IBM representative.

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 (http://www.ibm.com/legal/copytrade.shtml).

Glossary

This glossary defines technical terms and abbreviations. If you do not find the term you are looking for, refer to the index of the appropriate DFSMS manual, or view the *Glossary of Computing Terms* at this Web address:

http://www.ibm.com/ibm/terminology/

This glossary includes terms and definitions from the following sources:

- The *American National Standard Dictionary for Information Systems*, ANSI X3.172-1990, copyright 1990 by the American National Standards Institute (ANSI). You can purchase copies from the American National Standards Institute, 11 West 42nd Street, New York, New York 10036. The symbol (A) after a definition identifies it as a definition from this source.
- The *Information Technology Vocabulary* developed by Subcommittee 1, Joint Technical Committee 1, of the International Organization for Standardization and the International Electrotechnical Commission (ISO/IEC JTC1/SC1). The symbol (I) after a definition identifies it as a definition from the published part of this vocabulary. The symbol (T) after a definition identifies it as a definition from a draft international standard, committee draft, or working paper that ISO/IEC JTC1/SC1 is developing, indicating that the participating National Bodies of SC1 have not yet reached final agreement on the definition.
- The *IBM Dictionary of Computing*, New York: McGraw-Hill, 1994.

The following cross-reference is used in this glossary:

See Refers to a preferred term, a synonym, or a term that is the expanded form of an abbreviation or acronym, .

See also
Refers to a related term.

Contrast with
Refers to a contrasting term.

access method control block (ACB)
A control block that links an application program to VSAM or VTAM programs.

access method services
A multifunction service program that manages VSAM and non-VSAM data sets. Access method services provides the following services:

- Define and allocate space for data sets and catalogs
- Convert indexed-sequential data sets to key-sequenced data sets
- Modify data set attributes in the catalog
- Reorganize data sets
- Facilitate data portability among operating systems
- Create backup copies of data sets
- Assist in making inaccessible data sets accessible
- List the records of data sets and catalogs
- Define and build alternate indexes

ACID transaction
A transaction involving multiple resource managers using the two-phase commit process to ensure ACID (atomic, consistent, isolated, and durable) properties.

- **Atomic:** When an application changes data in multiple resource managers as a single transaction, and all of the changes are accomplished through a single commit request by a syncpoint manager, the transaction is called atomic. If the transaction is successful, all the changes will be committed. If any piece of the transaction is not successful, then all of the changes will be backed out. An atomic instant occurs when the syncpoint manager in a two-phase commit process logs a commit record for the transaction.
- **Consistent:** Applications involved in an ACID transaction must be written to maintain a consistent view of data. The transaction either makes valid changes to data or returns all the data to its state before the transaction was started.
- **Isolated:** Databases involved in an ACID transaction isolate the updates to their data so that only the application

changing the data knows about the individual update requests until the transaction is complete.

- **Durable:** Databases involved in an ACID transaction ensure that the data is persistent, both before and after the transaction, regardless of success or failure.

ACS routine
See *automatic class selection (ACS) routine*.

activity keypoint (AKP)
In DFSMStvs, a record of task-entry status in the system log made on a periodic basis to facilitate the identification of transaction backout information during restart. In the event of an uncontrolled shutdown and subsequent restart, activity keypoints can shorten the process of backward scanning through the system log.

activity keypoint interval
The number of logging operations that the system logger performs between keypoints.

addressed-direct access
In VSAM, the retrieval or storage of a data record identified by its relative byte address (RBA), independent of the record's location relative to the previously retrieved or stored record.

addressed-sequential access
In VSAM, the retrieval or storage of a data record in its entry sequence relative to the previously retrieved or stored record.

addressing mode (AMODE)
An attribute of an entry point in a program that identifies the addressing range in virtual storage that the module is capable of addressing. In 24-bit addressing mode, only 24-bit addresses can be used.

AKP See *activity keypoint*.

alias An alternative name for a catalog, a non-VSAM data set, or a member of a partitioned data set (PDS) or partitioned data set extended (PDSE).

alias entry
An entry that relates an alias to the real entry name of a user catalog or non-VSAM data set.

allocation
Generically, the entire process of obtaining a volume and unit of external storage, and setting aside space on that storage for a data set.

The process of connecting a program to a data set or devices.

alternate index
A key-sequenced data set that contains index entries organized by the alternate keys of its associated base data records. It provides an alternate means of locating records in the data component of a cluster on which the alternate index is based.

alternate key
One or more characters within a data record used to identify the data record or to control its use. Unlike the primary key, the alternate key can identify more than one data record. An alternate key is used to build an alternate index or to locate one or more base data records through an alternate index. See also *generic key, key,* and *key field*.

application owning region
A CICS address space whose primary purpose is to manage application programs.

application
The use to which an access method is put or the end result that it serves, contrasted to the internal operation of the access method.

ARM See *automatic restart manager*.

atomic
Pertaining to a transaction's changes to the state of resources: either all changes happen or none happen. It would not be possible for some updates to be made but for others to fail and still maintain data integrity. The process of making final changes to the data is *committing*.

automatic class selection (ACS) routine
A procedural set of ACS language statements. Based on a set of input variables, the ACS language statements generate the name of a predefined SMS class, or a list of names of predefined storage groups, for an MVS data set.

automatic restart manager (ARM)
A z/OS recovery function that can

automatically restart a batch job, started task, or abstract resource after it ends unexpectedly or after the system on which it is running goes down unexpectedly.

backout
A request to remove all changes to resources since the last commit or backout or for the first unit of recovery, since the beginning of the application. Backout is also called rollback or abort.

backout log
See *undo log*.

binder
The DFSMS program that processes the output of language translators and compilers into an executable program (load module or program object). It replaces the linkage editor and batch loader in in z/OS.

blocking
The process of combining two or more records into one block.

block size
The number of records, words, or characters in a block; usually specified in bytes.

CA See *control area*.

catalog
A data set that contains extensive information required to locate other data sets, to allocate and deallocate storage space, to verify the access authority of a program or operator, and to accumulate data set usage statistics. (A) (I)

central processor complex (CPC)
A physical collection of hardware that consists of main storage, one or more central processors, timers, and channels.

CI See *control interval*.

CICS Customer Information Control System.

CICSVR
Customer Information Control System VSAM Recovery, a forward recovery utility, which can can perform forward recovery for DFSMStvs and others as well as for CICS.

class See *SMS class*.

cluster
A data component and an index component in a VSAM key-sequenced data set; or a data component alone in a VSAM entry-sequenced data set.

commit
A request to make all changes to recoverable resources permanent since the last commit or backout or, for the first unit of recovery, since the beginning of the application.

component
A named, cataloged collection of stored records. A component, the lowest member of the hierarchy of data structures that can be cataloged, contains no named subsets.

compress
(1) To reduce the amount of storage required for a given data set by having the system replace identical words or phrases with a shorter token associated with the word or phrase. (2) To reclaim the unused and unavailable space in a partitioned data set that results from deleting or modifying members by moving all unused space to the end of the data set.

compressed format data set
A type of extended format data set created in a data format which supports record level compression.

configuration
The arrangement of a computer system as defined by the characteristics of its functional units.

See *SMS configuration*.

consistent read
A level of read integrity that VSAM RLS obtains for a share lock on the record that is accessed by a GET or POINT request. Consistent read ensures that the reader does not see an uncommitted change made by another transaction.

consistent read explicit
A level of read integrity that is the same as *consistent read*, except that VSAM RLS keeps the share lock on the record until the end of the transaction. This option is available only to CICS transactions and to

DFSMStvs. VSAM does not recognize the end of the transaction for usage other than by CICS or DFSMStvs. This capability is often referred to as repeatable read.

context
Sometimes called a work context, a context is a representation of a work request, or part of a work request, in an application. A context might have a series of units of recovery associated with it. See also *native context* and *privately managed context*.

control area (CA)
(1) A group of control intervals used as a unit for formatting a data set before adding records to it.

(2) In a key-sequenced data set, the set of control intervals, pointed to by a sequence-set index record, that is used for distributing free space and for placing a sequence-set index record adjacent to its data.

control blocks in common (CBIC)
A facility that allows a user to open a VSAM data set so the VSAM control blocks are placed in the common service area (CSA) of the MVS operating system. This provides the capability for multiple memory accesses to a single VSAM control structure for the same VSAM data set.

control interval (CI)
A fixed-length area of auxiliary storage space in which VSAM stores records. It is the unit of information (an integer multiple of block size) transmitted to or from auxiliary storage by VSAM.

control interval definition field (CIDF)
In VSAM, the 4 bytes at the end of a control interval that contain the displacement from the beginning of the control interval to the start of the free space and the length of the free space. If the length is 0, the displacement is to the beginning of the control information.

control program
A routine, usually part of an operating system, that aids in controlling the operations and managing the resources of a computer system.

control unit
A hardware device that controls the reading, writing, or displaying of data at one or more input/output devices. See also *storage control.*

cross memory
A synchronous method of communication between address spaces.

coupling facility (CF)
The hardware that provides high-speed caching, list processing, and locking functions in a Parallel Sysplex.

coupling facility (CF) cache structure
The CF hardware that provides a data cache.

coupling facility (CF) lock structure
The CF hardware that supports Parallel Sysplex-wide locking.

CPC See central processor complex.

data class
A collection of allocation and space attributes, defined by the storage administrator, that are used to create a data set.

data extent block (DEB)
A control block that describes the physical attributes of the data set.

Data Facility Storage Management Subsystem (DFSMS)
An operating environment that helps automate and centralize the management of storage. To manage storage, SMS provides the storage administrator with control over data class, storage class, management class, storage group, and automatic class selection routine definitions.

Data Facility Storage Management Subsystem data facility product (DFSMSdfp)
A DFSMS functional component or base element of z/OS that provides functions for storage management, data management, program management, device management, and distributed data access.

Data Facility Storage Management Subsystem data set services (DFSMSdss)
A DFSMS functional component or base

element of z/OS that is used to copy, move, dump, and restore data sets and volumes.

Data Facility Storage Management Subsystem Transactional VSAM Services (DFSMStvs)
An IBM licensed program for running batch VSAM processing concurrently with CICS online transactions. DFSMStvs users can run multiple batch jobs and online transactions against VSAM data, in data sets defined as recoverable, with concurrent updates. DFSMStvs is a licensed component of DFSMS

data record
A collection of items of information from the standpoint of its use in an application, as a user supplies it to the system storage. Contrast with *index record*.

data security
Prevention of access to or use of data or programs without authorization. As used in this publication, the safety of data from unauthorized use, theft, or purposeful destruction.

data set control block (DSCB)
A control block in the VTOC that describes data set characteristics.

data synchronization
The process by which the system ensures that data previously given to the system via WRITE, CHECK, PUT, and PUTX macros is written to some form of nonvolatile storage.

device number
The reference number assigned to any external device.

DFSMS
See *Data Facility Storage Management Subsystem*.

DFSMSdfp
See *Data Facility Storage Management Subsystem data facility product*.

DFSMSdss
See *Data Facility Storage Management Subsystem data set services*.

DFSMStvs
See *Data Facility Storage Management Subsystem Transactional VSAM Services*.

dictionary
A table that associates words, phrases, or data patterns to shorter tokens. The tokens are used to replace the associated words, phrases, or data patterns when a data set is compressed.

direct access
The retrieval or storage of data by a reference to its location in a data set rather than relative to the previously retrieved or stored data. See also *addressed-direct access*.

direct access device space management (DADSM)
A DFP component used to control space allocation and deallocation on DASD.

direct data set
A data set whose records are in random order on a direct access volume. Each record is stored or retrieved according to its actual address or its address according to the beginning of the data set. Normally accessed via BDAM.

directly-allocated printer
A printer that is allocated to the application program.

dynamic buffering
A user-specified option that requests that the system handle acquisition, assignment, and release of buffers.

entry-sequenced data set
A data set whose records are loaded without respect to their contents, and whose relative byte addresses (RBAs) cannot change. Records are retrieved and stored by addressed access, and new records are added at the end of the data set.

ESA See *Enterprise Systems Architecture*.

exclusive control
A way of preventing multiple write-add BDAM requests from updating the same dummy record or writing over the same available space on a track. When specified by the user, the exclusive control lock requests that the system prevent the data block that is about to be read from being modified by other requests; it is specified in a read macro and released in a write or relex macro. When a write-add request is about to be processed, the system

automatically gets exclusive control of either the data set or the track.

extended format
The format of a data set that has a data set name type (DSNTYPE) of EXTENDED, for example, extended format and extended key-sequenced data sets. Data sets in extended format can be striped or compressed. Data in an extended format VSAM KSDS can be compressed.

extended format data set
A sequential data set that is structured logically the same as a physical sequential data set but that is stored in a different physical format. Extended format data sets consist of one or more stripes and can take advantage of the sequential data striping access technique. See also *striping* and *stripe*.

extent A continuous space on a DASD volume occupied by a data set or portion of a data set.

field In a record or control block, a specified area used for a particular category of data or control information.

file-owning region (FOR)
A data-owning region, a CICS address space whose primary purpose is to manage files and databases.

file system
In the z/OS UNIX hierarchical file system (HFS) environment, the collection of files and file management structures on a physical or logical mass storage device, such as a diskette or minidisk. See also *hierarchical file system (HFS) data set*.

FOR See *file-owning region*.

forgotten
The state of a unit of recovery that occurs when the unit of recovery has completed and RRS has deleted its log records.

format-D
ASCII variable-length records.

format-DB
ASCII variable-length, blocked records.

format-DBS
ASCII variable-length, blocked spanned records.

format-DS
ASCII variable-length, spanned records.

format-F
Fixed-length records.

format-FB
Fixed-length, blocked records.

format-FBS
Fixed-length, blocked, standard records.

format-FS
Fixed-length, standard records.

format-U
Undefined-length records.

format-V
Variable-length records.

format-VB
Variable-length, blocked records.

format-VBS
Variable-length, blocked, spanned records.

format-VS
Variable-length, spanned records.

Forward recoverable data set
A data set that was defined with the LOG(ALL) attribute option.

forward recovery
A process used to recover a lost data set. The data is recovered from a backup copy and all the changes that were made after the backup copy was taken are applied. The forward recovery process requires a log of the changes made to a data set, together with a date and time stamp. The log of changes is called the forward recovery log.

forward recovery log
A log that contains copies of records after they were changed. The forward recovery log records are used by forward recovery programs and products such as CICS VSAM Recovery (CICSVR) to reconstruct the data set in the event of hardware or software damage to the data set.

free space
Space reserved within the control intervals of a key-sequenced data set for inserting new records into the data set in key sequence or for lengthening records already there; also, whole control intervals reserved in a control area for the same purpose.

gigabyte
1 073 741 824 bytes.

global shared resources (GSR)
. Indicates use of a global resource pool.

GSR See *global shared resources*.

header label
An internal label, immediately preceding the first record of a file, that identifies the file and contains data used in file control.

The label or data set label that precedes the data records on a unit of recording medium.

HFS See *hierarchical file system*

hierarchical file system (HFS) data set
A data set that contains a POSIX-compliant file system, which is a collection of files and directories organized in a hierarchical structure, that can be accessed using z/OS UNIX System Services. See also *file system*.

hierarchical file system (HFS)
A POSIX-compliant file system, which is a collection of files and directories organized in a hierarchical structure, that can be accessed using z/OS UNIX System Services. HFS enables an application written in a high-level language to create, store, retrieve, and manipulate data on a storage device. The view of the data to the end user is a hierarchical directory structure similar to IBM DOS. See also *file system*.

index record
A collection of data-record pointers retrieved and stored together. Contrast with *index record*

instance
The code and control blocks that represent access to VSAM data sets through DFSMStvs. An instance of DFSMStvs starts when DFSMStvs is initialized as part of SMSVSAM address space initialization or enabled by operator command. The instance ends when DFSMStvs enters a quiesced or disabled state or when the SMSVSAM address space ends.

A peer recovery instance of DFSMStvs serves to recover from the failure of some other DFSMStvs instance that ran on another system within a sysplex when the system on which the other DFSMStvs instance was running failed. The peer recovery instance shares the SMSVSAM address space, and certain control blocks, with a "native" DFSMStvs instance. Automatic restart manager (ARM) can start a peer recovery instance automatically when a system in a sysplex fails. A peer recovery instance can also be started manually by operator command. The instance ends when it completes its peer recovery process, when it is stopped by operator command and enters a quiesced state, or when the SMSVSAM address space ends.

in-backout
The state of a unit of recovery when one or more resource managers reply negatively to a commit request. The syncpoint manager tells each resource manager to back out the changes. The resources are returned to the values they had before the unit of recovery was processed. When all the resource managers have backed out the changes, the syncpoint manager notifies the application.

in-commit
The state of a unit of recovery when all resource managers reply positively to a commit request. The syncpoint manager tells each resource manager to make its changes permanent. When all resource managers have made the changes, the syncpoint manager notifies the application.

in-completion
The state of a unit of recovery when any enabled completion exit routines run. After this phase completes, RRS passes a return code to the application indicating that the changes have been committed or backed out.

in-doubt
For a distributed request, the state of a unit of recovery on the originating system from the end of the prepare phase of the two-phase commit until the distributed syncpoint resource manager (DSRM) returns a commit or backout request.

in-end The state of a unit of recovery when the resource managers have responded to the syncpoint manager that commit or backout is complete. The unit of recovery is logically complete.

in-flight
The state of a unit of recovery when an application accesses protected resources. The resource managers express interest in the unit of recovery.

in-forget
The state of a unit of recovery for a distributed request. The unit of recovery has completed, but RRS is waiting for the server distributed syncpoint resource manager (SDSRM) to indicate how to process the log records for the unit of recovery.

in-only-agent
The state of a unit of recovery when only one resource manager has expressed an interest in the unit of recovery. RRS invokes the ONLY_AGENT exit routine to tell the resource manager to process the commit immediately.

in-prepare
The state of a unit of recovery when the application has issued a commit request and the syncpoint manager tells each resource manager to prepare its resources for commit or backout.

in-reset
The state of a unit of recovery before an application program has used any protected resources.

in-state-check
The state of a unit of recovery when the application has issued a commit request and the resource managers check if their resources are in the correct state.

key-sequenced data set (KSDS)
A VSAM data set whose records are loaded in ascending key sequence and controlled by an index. Records are retrieved and stored by keyed access or by addressed access, and new records are inserted in key sequence because of free space allocated in the data set. Relative byte addresses of records can change because of control interval or control area splits.

keyed-sequential access
In VSAM, the retrieval or storage of a data record in its key or relative-record sequence, relative to the previously retrieved or stored record as defined by the sequence set of an index.

kilobyte
1024 bytes.

library
Synonym for partitioned data set. See *partitioned data set*.

linear data set (LDS)
A VSAM data set that contains data but no control information. A linear data set can be accessed as a byte-addressable string in virtual storage.

load module
The output of the linkage editor; a program in a format ready to load into virtual storage for execution. Contrast with program object.

local shared resources (LSR)
Resources in the local resource pool.

locate mode
A transmittal mode in which a pointer to a record is provided instead of moving the record. Contrast with *move mode*.

log of logs
A log that DFSMStvs and CICS write to provide information to forward recovery programs such as CICS VSAM Recovery (CICSVR). The log of logs is a form of user journal that contains copies of the tie-up records that DFSMStvs or CICS has written to forward recovery logs. This log provides a summary of which recoverable VSAM data sets that DFSMStvs or CICS has used, when they were used, and to which log stream the forward recovery log records were written.

If you have a forward recovery product that can utilize the log of logs, ensure that all CICS regions that share the recoverable data sets write to the same log-of-logs log stream.

log stream
A log stream is a collection of data in log blocks that reside in the coupling facility or on DASD.

log tail
In DFSMStvs, the oldest log record of interest. Log tail deletion is the process of deleting unneeded records that are older than the oldest record of interest to DFSMStvs.

log trimming
Removal of records that are no longer required from the DFSMStvs primary system log or secondary system log.

LSR See *local shared resources*.

management class
A collection of management attributes, defined by the storage administrator, used to control the release of allocated but unused space; to control the retention, migration, and back up of data sets; to control the retention and back up of aggregate groups, and to control the retention, back up, and class transition of objects.

member
A partition of a partitioned data set or PDSE.

move mode
A transmittal mode in which the record to be processed is moved into a user work area.

MVS Multiple Virtual Storage.

native context
The automatically occurring context of a work request. A native context is associated with a single task. This context always exists.

non-VSAM data set
A data set allocated and accessed using one of the following methods: BDAM, BPAM, BISAM, BSAM, QSAM, QISAM.

nonrecoverable data set
A data set for which no changes are logged because its LOG parameter is either undefined or set to NONE. Neither backout nor forward recovery is provided for a nonrecoverable data set.

nonshared resources
A data set that does not use shared resources.

NSR See *nonshared resources*.

object A named byte stream having no specific format or record orientation.

z/OS UNIX System Services (z/OS UNIX)
The set of functions provided by the SHELL and UTILITIES, kernel, debugger, file system, C/C++ Run-Time Library, Language Environment, and other elements of the z/OS operating system that allow users to write and run application programs that conform to UNIX standards.

operand
Information entered with a command name to define the data on which a command operates and to control the execution of the command.

optimum block size
For non-VSAM data sets, optimum block size represents the block size that would result in the smallest amount of space utilization on a device, taking into consideration record length and device characteristics.

Parallel Sysplex
A sysplex that uses one or more coupling facilities.

partitioned data set (PDS)
A data set on direct access storage that is divided into partitions, called members, each of which can contain a program, part of a program, or data.

partitioned data set extended (PDSE)
An system-managed data set that contains an indexed directory and members that are similar to the directory and members of partitioned data sets. A PDSE can be used instead of a partitioned data set.

path A named, logical entity composed of one or more clusters (an alternate index and its base cluster, for example).

PDS directory
A set of records in a partitioned data set (PDS) used to relate member names to their locations on a DASD volume.

peer recovery
A recovery process that occurs when an

application fails. Peer users can perform recovery and clean up resources.

peer recovery instance
See *instance*.

pointer
An address or other indication of location. For example, an RBA is a pointer that gives the relative location of a data record or a control interval in the data set to which it belongs.

primary key
One or more characters within a data record used to identify the data record or control its use. A primary key must be unique.

primary space allocation
Amount of space requested by a user for a data set when it is created. Contrast with *secondary space allocation*.

primary system log
See *undo log*.

privately managed context
A context created and owned by a resource manager. The resource manager can switch a privately managed context from one task to another. Privately managed contexts are usually used by a resource manager that is also a work manager, like IMS. This sort of work manager can accept and manage transactions, or other kinds of work, from outside the system.

program library
A type of PDSE which contains program objects only. A PDSE from which programs are loaded into memory for execution by the operating system.

program object
All or part of a computer program in a form suitable for loading into virtual storage for execution. Program objects are stored in PDSE program libraries and have fewer restrictions than load modules. Program objects are produced by the binder.

protected resource
A local or distributed resource that can be changed in a synchronized manner during processing coordinated by a syncpoint manager, such as RRS. Databases, conversations between two communications managers, or product-specific resources can all be protected resources. A protected resource is also often called a *recoverable resource*.

random access
See *direct access*.

record definition field (RDF)
A field stored as part of a stored record segment; it contains the control information required to manage stored record segments within a control interval.

record-level sharing
See *VSAM record-level sharing (VSAM RLS)*.

recoverable data set
A data set that can be recovered using backout or forward recovery processing, defined with the LOG parameter set to `UNDO` or `ALL`. See also *protected resource*.

recoverable resource
A data set that can be recovered using commit, backout, or forward recovery processing because its LOG parameter is set to `UNDO` or `ALL`.

register
An internal computer component capable of storing a specified amount of data and accepting or transferring this data rapidly.

relative byte address (RBA)
The displacement of a data record or a control interval from the beginning of the data set to which it belongs; independent of the manner in which the data set is stored.

relative record data set (RRDS)
A type of VSAM data set whose records have fixed or variable lengths, and are accessed by relative record number.

residence mode (RMODE)
The attribute of a load module that identifies where in virtual storage the program will reside (above or below 16 megabytes).

resource
A database, a conversation between two systems, or a product-specific item. A resource can be local (residing on the current system) or distributed (residing on another system). A resource is

protected when it can be changed in a synchronized manner.

resource manager (RM)
A subsystem or component, such as CICS, IMS, or DB2, or DFSMStvs, that manages resources that can be involved in transactions. There are three types of resource managers: work managers, data resource managers, and communication resource managers.

reusable data set
A VSAM data set that can be reused as a work data set, regardless of its old contents. It must not be a base cluster of an alternate index.

RLS See *VSAM record-level sharing (VSAM RLS)*.

scheduling
The ability to request that a task set should be started at a particular interval or on occurrence of a specified program interrupt.

secondary space allocation
Amount of additional space requested by the user for a data set when primary space is full. Contrast with *primary space allocation*.

secondary system log
See *shunt log*.

security
See *data security*.

sequence checking
The process of verifying the order of a set of records relative to some field's collating sequence.

sequential access
The retrieval or storage of a data record in: its entry sequence, its key sequence, or its relative record number sequence, relative to the previously retrieved or stored record. See also *addressed-sequential access* and *keyed-sequential access*.

sequential data set
A data set whose records are organized on the basis of their successive physical positions, such as on magnetic tape. Contrast with *direct data set*.

service request block (SRB)
A system control block used for dispatching tasks.

shared lock
A lock that several tasks can hold.

shared resources
A set of functions that permit the sharing of a pool of I/O-related control blocks, channel programs, and buffers among several VSAM data sets open at the same time.

shunt The process of moving failed work or long-running work from the primary system log to the secondary system log. If a unit of work fails, it is removed (shunted) from the primary system log to the secondary system log, pending recovery from the failure.

shunt log
The secondary system log, which contains entries that were shunted to the log when DFSMStvs was unable to finish processing sync points. If a *unit of work* fails, it is removed (shunted) from the primary system log to the secondary system log, pending recovery from the failure.

shunted
The state of a unit of recovery when it has been moved from the primary system log to the secondary system log because of a failed or long-running unit of work.

slot For a fixed-length relative record data set, the data area addressed by a relative record number, which might contain a record or be empty.

single point of failure
An environment in which one failure can result in simultaneous loss of both the coupling-facility list structure for a log stream and the local storage-buffer copy.

skip-sequential access
Keyed-sequential retrieval or storage of records here and there throughout a data set, skipping automatically to the desired record or collating position for insertion: VSAM scans the sequence set to find a record or a collating position. Valid for processing in ascending sequences only.

SMF See *System Management Facilities.*

SMS class
A list of attributes that SMS applies to

data sets having similar allocation (data class), performance (storage class), or backup and retention (management class) needs.

SMS configuration
A configuration base, Storage Management Subsystem class, group, library, and drive definitions, and ACS routines that the Storage Management Subsystem uses to manage storage.

SMS-managed data set
A data set that has been assigned a storage class.

spanned record
For VSAM, a logical record whose length exceeds control interval length, and as a result, crosses, or spans one or more control interval boundaries within a single control area. For non-VSAM, a spanned record that occupies part or all of more than one block.

staging data set
Staging data sets are allocated by the system logger to safeguard log data when there is an error that leaves the only copy of log data in a volatile configuration.

storage class
A collection of storage attributes that identify performance goals and availability requirements, defined by the storage administrator, used to select a device that can meet those goals and requirements.

storage control
The component in a storage subsystem that handles interaction between processor channel and storage devices, runs channel commands, and controls storage devices.

storage group
A collection of storage volumes and attributes, defined by the storage administrator. The collections can be a group of DASD volumes or tape volumes, or a group of DASD, optical, or tape volumes treated as a single object storage hierarchy.

Storage Management Subsystem (SMS)
A DFSMS facility used to automate and to centralize the management of storage. Using SMS, a storage administrator describes data allocation characteristics, performance and availability goals, backup and retention requirements, and storage requirements to the system through data class, storage class, management class, storage group, and ACS routine definitions.

store-through caching
A process used by the store-through user in which changed data is written to the cache structure and to permanent storage at the same time and under the same serialization so that at any time, the data in the cache structure matches the data in permanent storage.

stripe The portion of a striped data set (for example, an extended format data set) that resides on one volume. The records in that portion are not necessarily logically consecutive. The system distributes records among the stripes such that the volumes can be read or written simultaneously to gain better performance.

striping
A software implementation of a disk array that distributes data sets across multiple volumes to improve performance.

system-managed storage
Storage managed by the Storage Management Subsystem. SMS attempts to deliver required services for availability, performance, and space to applications. See also *DFSMS environment*.

system management facilities (SMF)
A component of z/OS that collects input/output (I/O) statistics, provided at the data set and storage class levels, which help you monitor the performance of the direct access storage subsystem.

sync point
An end point during processing of a transaction. A sync point occurs when an update or modification to one or more of the transaction's protected resources is logically complete. A sync point can be either a commit or a backout.

syncpoint manager
A syncpoint manager is a function that coordinates the two-phase commit process for protected resources, so that all changes to data are either committed or

backed out. In z/OS, RRS can act as the system level syncpoint manager.

tie-up record
A record that associates each data set after-image record in the log with a file name. You can associate a data set with more than one file with the same data set. When a file is opened, DFSMStvs records the association between the file and the data set as a tie-up record in the forward recovery log. This information is also written to the log of logs. For non-BWO backups, the forward recovery utility uses this tie-up record to apply the log records to the correct data sets.

transaction
A unit of application data processing initiated by a single request. A transaction might involve multiple application programs and might require the initiation of one or more jobs for its execution. In DFSMStvs, a transaction is a unit of work, which consists of one or more logical units of recovery.

transaction ID (TRANSID)
A number associated with each of several request parameter lists that define requests belonging to the same data transaction.

TRANSID
See *transaction ID*.

trimming
See *log trimming*.

two-phase commit
The process used by syncpoint managers and resource managers to coordinate changes in an ACID transaction.

In the first phase of the process, resource managers prepare a set of coordinated changes, but the changes are uncommitted pending the agreement of all the resource managers involved in the transaction. In the second phase, those changes are all committed if the resource managers all agreed to them; or, the changes are all backed out if any of the resource managers failed or disagreed.

Using the two-phase commit process, multiple changes across multiple resource managers can be treated as a single ACID transaction.

undo log
The primary system log, which contains images of changed records as they existed prior to being changed. Backout processing uses the undo log to back out the changes that a transaction made to resources.

unit address
The last two hexadecimal digits of a device address. This identifies the storage control and DAS string, controller, and device to the channel subsystem. Often used interchangeably with control unit address and device address in System/370 mode.

unit of recovery (UR)
A set of changes on one node that is committed or backed out as part of an ACID transaction.

A UR is implicitly started the first time a resource manager touches a protected resource on a node. A UR ends when the two-phase commit process for the ACID transaction changing it completes.

unit of recovery identifier (URID)
Persistent tokens used by RRS to identify a transaction.

unit of work
In DFSMStvs, one or more logical units of recovery that are committed or backed out together as a transaction.

universal character set (UCS)
A printer feature that permits the use of a variety of character arrays. Character sets used for these printers are called UCS images.

update number
For a VSAM spanned record, a binary number in the second RDF of a record segment that indicates how many times the segments of a spanned record should be equal. An inequality indicates a possible error.

user buffering
The use of a work area in the processing program's address space for an I/O

buffer; VSAM transmits the contents of a control interval between the work area and direct access storage without intermediary buffering.

virtual storage access method (VSAM)
An access method for direct or sequential processing of fixed and variable-length records on direct access storage devices. You can organize the records in a VSAM data set in logical sequence by a key field (key sequence), in the physical sequence in which they are written to the data set (entry sequence), or by relative record numbers.

VSAM
See *virtual storage access method.*

VSAM record-level sharing (VSAM RLS)
An extension to VSAM that provides direct record-level sharing of VSAM data sets from multiple address spaces across multiple systems. Record-level sharing uses the z/OS coupling facility to provide cross-system locking, local buffer invalidation, and cross-system data caching.

VSAM volume data set (VVDS)
A data set that describes the characteristics of VSAM and system-managed data sets residing on a given DASD volume; part of a catalog.

z/OS A network computing-ready, integrated operating system consisting of more than 50 base elements and integrated optional features delivered as a configured, tested system.

Index

Numerics

A

B

C

D

E

F

G

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Product Number: 5650-ZOS

Printed in USA

GC52-1388-00